AUTHOR QUERY FORM

ELSEVIER	**Journal: MICPRO** **Article Number: 1849**	**Please e-mail or fax your responses and any corrections to:** **E-mail: corrections.esch@elsevier.sps.co.in** **Fax: +31 2048 52799**

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list.

For correction or revision of any artwork, please consult http://www.elsevier.com/artworkinstructions.

No queries have arisen during the processing of your article.

Thank you for your assistance.

Microprocessors and Microsystems xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A hardwired NoC infrastructure for embedded systems on FPGAs

Muhammad E.S. Elrabaa *, Abdelhafidh Bouhraoua

Computer Engineering department, King Fahd University for Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Multi-core embedded systems
Field Programmable Gate Arrays (FPGAs)
Networks-on-chip
Systems-on-chips

ABSTRACT

A hardwired network-on-chip based on a modified Fat Tree (MFT) topology is proposed as a communication infrastructure for future FPGAs. With extremely simple routing, such an infra structure would greatly enhance the ongoing trend of embedded systems implementation using multi-cores on FPGAs. An efficient H-tree based floor plan that naturally follows the MFT construction methodology was developed. Several instances of the proposed NoC were implemented with various inter-routers links progression schemes combined with very simple router architecture and efficient client network interface (CNI). The performance of all these implementations was evaluated using a cycle-accurate simulator for various combinations of NoC sizes and traffic models. Also a new data transfer circuit for transferring data between clients and NoC operating at different (unrelated) clock frequencies has been developed. Allowing data transfer at one data per cycle, the operation of this circuit has been verified using gate-level simulations for several ratios of NoC/client clock frequencies.

1. Introduction

Field Programmable Gate Arrays (FPGAs) have gained considerable acceptance among VLSI designers not only as prototyping platforms but also as system implementation platforms for applications with short time to market constraints. State of the art FPGAs have also become very attractive for system-on-chip (SOC) and embedded systems designs due to their ease of use, flexibility, large gate count, abundance of I/Os and efficient hardware macros, embedded processors (both hard and soft), re-configurability, extensive tool support and improved speed [1,2]. FPGAs can be configured (and re-configured) to implement any digital processor or group of processors (called cores or Intellectual Property blocks or IPs). Several cores could be implemented simultaneously that can communicate among themselves and the outer world. This matches the current trend of multi-core implementation of embedded systems [1]. This lead to a new trend of implementing embedded systems on FPGAs. The inter-core communications in these implementations, however, are still implemented as dedicated point-to-point or using shared buses. The first method though provides good speed, consumes relatively large portion of the FPGA's logic/interconnect resources reducing the total number of cores that can be implemented. The second method is more resource efficient but suffers from performance penalties and does not scale well. Having a scalable interconnection network (i.e. a network-on-chip) on future FPGAs has the highest potential for satisfying the communication needs of future FPGA-based embedded systems [1].

Conventional FPGAs suffered from relatively high configuration time (time needed to load all the configuration bits into the FPGA) due to the limited number of external configuration ports and the serial nature of the configuration process itself. Also, configuration interconnects were completely separated from data/functional interconnects. Since adding more external configuration ports would be at the expense of regular data I/Os, FPGA vendors added internal reconfiguration ports [3] that can be accessed by various cores implemented on the FPGA itself. Vendors also have divided their FPGAs into areas that can be configured separately with each area having its own local interconnect with functional and configuration interconnects remaining separate. This alleviated some of the problems associated with the serial nature of reconfiguration making dynamic reconfiguration (re-configuring the FPGA during use) more practical. Still, having separate configuration and functional interconnects is not only wasteful of space but means that special circuitry are required to connect the regular data port of an IP to the configuration interconnects and then to the internal configuration port of the region being re-configured. This requires having a dedicated configuration management circuit on the FPGA, yet more wasting of valuable logic resources. Hence it makes perfect sense to share the physical interconnects by functional (regular) data and configuration data. This would not only free valuable FPGA space but would also make available more intercon-

* Corresponding author. Tel.: +966 3 860 1496; fax: +966 3 860 3059.
E-mail addresses: elrabaa@kfupm.edu.sa (M.E.S. Elrabaa), abouh@kfupm.edu.sa (A. Bouhraoua).

0141-9331/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.09.008

nect resources available for reconfiguration, enabling fast dynamic run-time reconfiguration. This in turn would open the door for new and innovative computing paradigms.

Another major problem with FPGA interconnects is their relatively large delays. This is due to the fact that in order to make them configurable switch boxes (made of MUXs or Pass gates) are inserted at regular lengths along the interconnect wires. These wire delays constitute a challenge for circuit designers to meet their timing constraints (i.e. timing closure of the design). The same problem (timing closure) has faced designers of conventional SoCs where a chip is assembled by integrating different IPs, possibly from different vendors, with various operating frequencies and communication patterns. Networks-on-chip were proposed to solve these problems by providing a scalable shared interconnect medium that can meet the different communication requirements of IPs [4–6]. As a result, there has been a significant amount of effort made in the area of NoCs, and the focus has mostly been on proposing new topologies, and routing strategies. However, recently the trend has shifted towards engineering solutions and providing design tools that are more adapted to reality. For example, power analysis of NoC circuitry has intensively been studied [7,8], more realistic traffic models have been proposed [9], and more adapted hardware synthesis methodologies have been developed.

High throughput architectures, however, haven't been addressed enough in the literature. Besides the efforts related to the Nostrum [10] and the Æthreal [11], most of the other efforts were based on a regular mesh topology with throughputs (expressed as a fraction of the wire speed) not exceeding 30% [11].

In [12,13] a NoC topology based on a modified Fat Tree (MFT) was proposed to address the throughput issue. MFT is a subclass of Fat Trees (FT) multi-stage-interconnection networks [14]. The conventional Fat Tree topology was modified by adding enough links such that contention was completely eliminated thus achieving a throughput of nearly 100% [12] while eliminating any buffering requirement in the routers. Also, simplicity of the routing function, typical of Trees, meant that the router architecture is greatly simplified. The approach was extended to satisfy heterogeneous bandwidth requirements of different clients [15]. The new approach allowed NoC designers to satisfy the bandwidth requirement of each client without the need to "overdesign" the NoC, reducing power consumption without impacting the performance of applications running on the NoC.

Many researchers have proposed *soft* NoCs for FPGAs that utilize the FPGA's configurable resources to implement the NoC's various components (e.g. [16–19]). Though many of these soft NoCs have demonstrated various degrees of performance improvement, they consumed significant portion of the FPGA's resources (logic/interconnects/memories). Most critical of these are memory blocks (for implementing I/O buffers in the NoC interface) which leave designers with insufficient RAM blocks to implement their applications.

As a result of the limitations of soft NoCs, several researchers have recently proposed introducing NoCs as FPGA macros (i.e. as hardwired, full-custom designed circuitry) [20–26]. In [20] a reconfigurable router that can be configured to have various numbers of ports was first proposed. Users would need to implement the routing algorithm they desire and configure the NoC accordingly. In [21] a configurable Mesh-based NoC is proposed. This NoC is only modeled in SystemC using channels to implement the routing protocols with no actual implementation provided. Also the NoC itself require significant reconfiguration (routing tables, logical to physical addresses mapping, virtual point-to-point connections, etc.). Although Mesh topology would fit conveniently with the tile-based FPGA re-configuration regions, it has sever performance limitations in terms of achievable throughput (<30% of the wire speed) [10,11]. Another Mesh-based NoC is proposed in [22]. It combines hardwired resources (links and routers) with soft resources (configurable network interface). This provided extra flexibility at the cost of using up the FPGA's resources and reducing the speed since soft parts will inevitably have lower speed than the hardwired parts. Elmiligi et al. [23] proposes a centralized programmable switch fabric to connect routing nodes and coarse grain configurable processing elements. This strategy won't work with regular fine-grained FPGAs because the size of the central switch will grow exponentially with the number of I/O ports. A similar approach is proposed in [24]. Based on the Æthereal NoC [11], routers and links are implemented as hardwired blocks while the network interfaces are split between hard and soft parts. The NoC requires significant configuration to implement the Æthereal's different QoS services. This in turn requires the implementation of a dedicated NoC configuration manager as part of a system manager [25], using up yet more FPGA resources or area. Configurable router that can support several NoC topologies is proposed in [26]. Routers contain the required logic circuitry to implement the supported topologies' routing algorithms, something that is wasteful of FPGA chip area. Furthermore, Routers and clients are connected using the FPGA's configurable interconnects and using asynchronous handshaking (Req-Acknowledge). This is not only wasteful of further FPGA resources, but also limiting to the over all NoC performance. As was demonstrated in [33], full-custom hardware components can achieve much higher performance at a very small fraction of the area of configurable logic.

In this work a new FPGA hardwired NoC is proposed. The proposed NoC would provide an efficient infrastructure for functional inter-module communications as well as reconfiguration of the FPGA's regions. A brief overview of the proposed approach is first introduced in the next section followed by a brief description of how the NoC is constructed (in terms of links and routers) and the proposed physical floor plan. In Section 3 the detailed NoC design is presented including router's architecture, links progression, and client interface design. This is followed by performance evaluation (in terms of throughput and packet delays) of the proposed NoC implementations options under various conditions of size and traffic types. This includes a detailed gate count for these options and the area impact of adding a hardwired NoC to an FPGA. Finally conclusions are provided in Section 5.

2. Approach

To circumvent the limitations of the previously proposed FPGA NoCs, the target of this work was to develop an FPGA NoC infrastructure with minimal reconfiguration requirements. This would minimize the use of soft parts in the NoC implementation, increasing the speed and reducing the overall area footprint of the NoC. Also to achieve maximum data rates in the NoC a full synchronous design of the NoC is adopted. Since the NoC is designed as a full-custom circuit with a relatively small area, it is not difficult to design it as a synchronous circuit with its own clock tree. In fact an efficient H-tree floor plan was developed for the proposed NoC that would match the clock tree's own topology (usually laid out as an H-tree) simplifying the timing closure of the NoC. Since IPs will have their own clocks, specialized circuitry was developed to transfer data between the NoC and IPs at the maximum possible data rate. Also the developed NoC would have minimal intrusion on existing FPGA architectures for seamless integration into existing FPGA design flows. It is meant to be an infra structure for providing a fast medium for delivering functional/configuration data to all parts of the FPGA. In other word it provides the lower layers (up to routing) of the networking protocol while the higher layers are left for the system designer to choose and implement (or

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

choose from a library of IPs). The fixed NoC infrastructure is based on the MFT architecture [12,13]. The router architecture is further simplified by eliminating adaptive routing altogether. This would not only reduce the router area, but it will also insure in-order reception of packets, thus simplifying the client network interface design significantly. Also to reduce the NoC area further and to simplify the clients' network interface, the original MFT architecture was modified by reducing the number of links significantly. Different strategies for link reduction were evaluated using simulations to determine the best strategy in terms of throughput and gate count.

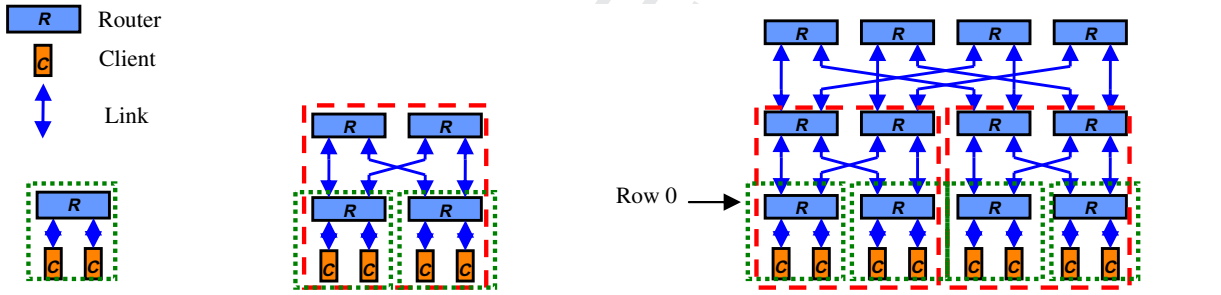
3. NoC construction and floor plan

3.1. Modified Fat Tree (MFT) NoC construction

Fig. 1 shows the recursive construction of a FT network. The starting basic unit, called a group of order 1, is composed of a single router with two clients attached to it. This unit is replicated and a new row of routers is added. Higher order groups are then formed in the same way from lower order groups, Fig. 1. Hence an n-client network will have $(n/2)^*$ $\mathrm{LOG}_2(n)$ routers organized as $n/2$ columns by $\mathrm{LOG}_2(n)$ rows. In this work, rows are numbered from 0 to $LOG_2(n) - 1$, with row 0 being the first level of routers which are connected to the clients. The MFT is constructed in a similar manner with the addition of additional links, Fig. 2. In the original MFT the downward output ports (links) are double the number of upper links for each router. This link doubling (referred to in this work as geometrical link progression) would proceed from the top row to row 0. Hence at the client/NoC interface routers will have $2^{n+1} - 1$ downward links (i.e. $2^n - 1$ input links per client). Each of these I/P links featured a FIFO buffer as called for by the original MFT architecture [12]. In this work two new modifications were adopted for the MFT; (1) adaptive upward routing was eliminated to further simplify the router's design and guaranteeing in-order packet arrival, thus simplifying the client interface greatly, Fig. 2) Different schemes for increasing the number of downward output links (i.e. link progression) were adopted. The later modification was adopted to reduce the number of links and FIFOs at the edge of the NoC at the client interface. This was a result of observations from numerous simulations with different traffic models that most of the extra links were inactive for most of the time. The router design was modified to support the different link progression schemes evaluated.

3.2. Routing in MFT

Packets are routed up till they reach a summit router where there is a direct path to the destination. So when an upward packet is received by a router at row r, $LOG_2(n) - r - 1$ bits of the destination address are simply compared to a stored pattern of equal length that indicates all the addresses reachable from that router. If they match the packet is routed down the opposite side, if not, the packet is routed up on the same side till it reach the summit. Downward packets are routed right or left based on the value of one of the destination address bits. The exact bit locations within the destination address that are used for upward and downward routing depends on the router location and are hardwired into



(a) The Basic unit (Group of order 1) (b) 4 clients group made of 2 two clients groups (order 2) (c) 8 clients group made of 2 four clients groups (order 3)

Fig. 1. The recursive construction process of a FT.

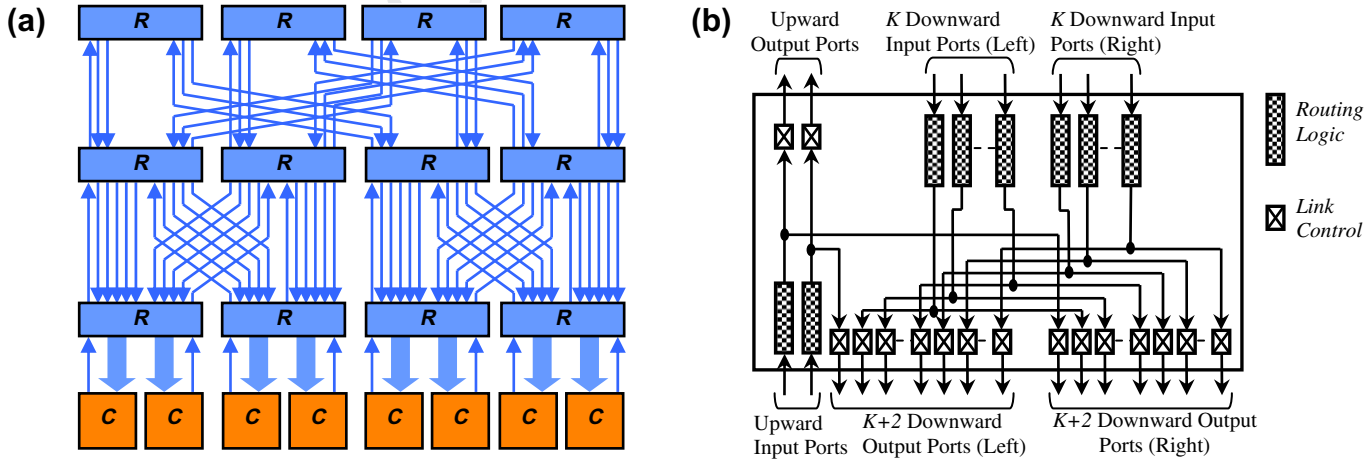


Fig. 2. (a) The original MFT topology and (b) router architecture with downward link doubling (geometrical link progression).

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

the router during design. This greatly simplifies the routers design. Also due to the deterministic routing and the buffer-less nature of the network packet order is maintained.

3.3. NoC floor plan

An efficient floor plan is proposed for the NoC based on the H-tree topology, Fig. 3. Each tile in the floor plan represents a configuration region (CR) and it will be considered as a single client from the NoC design point of view. This is just for the purpose of configuration (i.e. each of these tiles represents the minimum re-configuration region for the purpose of partial configuration). An actual client may occupy any number of CRs and can utilize any (or all) of the NoC ports in these CRs. Also two (or more) clients can occupy the same CR tile, but the designer needs to implement his/her own interface switch to time multiplex the NoC port in that CR. It should be noted that this floor plan is just illustrative and not an actual layout (i.e. it is not to scale). In reality, the routers and links area would be very small compared to CR tiles. Fig. 3 also lists the maximum length of links in terms of CR length.

As Fig. 3 shows, the routers and links layout follow the borders of the CRs to minimize the impact on conventional FPGA structure. With this floor plan, the longest links are the most upper links that span one-half the chip length. They are, however, fewer in numbers, making the routing of these links' wires easier. Also, proper buffer insertion in these links must be applied to maintain the required clock rate.

4. NoC design

4.1. Link progression schemes in the MFT NoC

Three types of link progression in the MFT NoC have been evaluated in this work; (1) Arithmetic progression, (2) Mixed Arithmetic/Geometrical progression and (3) Limited Geometrical progression. The three types are briefly explained below.

4.1.1. Arithmetic link progression

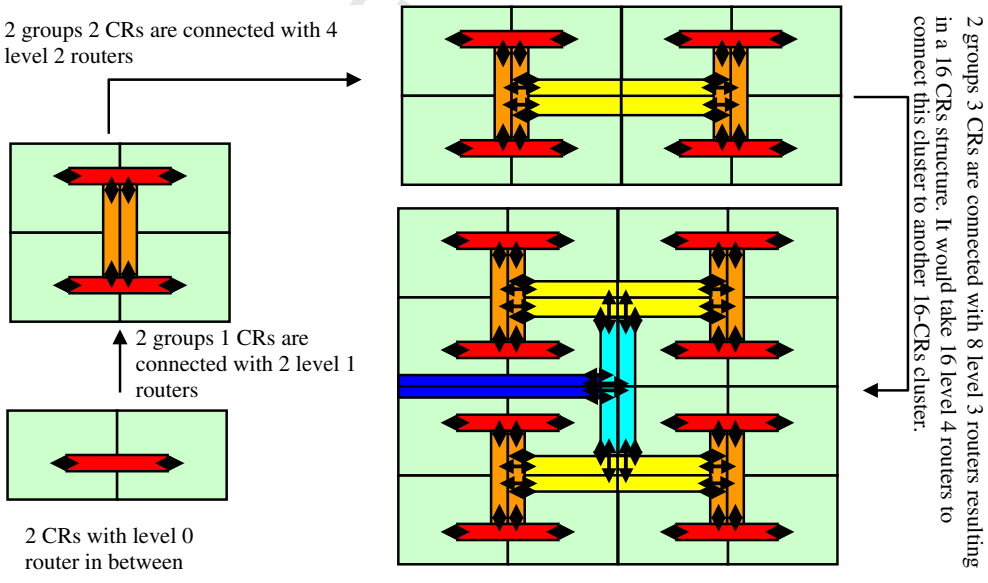
The Arithmetic progression MFT, abbreviated as AMFT, is based on partial reduction of contention through the addition of a fixed number of links on the downward path of each router. Hence, instead of the doubling called for by the original MFT architecture, the number of output ports of each router is equal to the number of input ports plus a fixed number called the *increment* (I_A). An increment of 2, means adding an extra link to both the left and right groups of output links. Another parameter, *stop level* (S_L) is introduced to increase the variations of the network structure exploration. Starting from the top row, S_L represents the row at which incrementing the output links will stop leaving the routers in lower levels with an identical number of inputs and outputs. Table 1 below illustrates the arithmetic progression for a NoC with 64 CRs for various combinations of I_A and S_L. It shows the number of output links per side (left or right) per router for each row in the NoC. The last column shows the corresponding MFT number of output links when doubling of the links is used. It is very clear that this scheme results in a significant decrease in the number of links at most levels especially the last one (row 0).

4.1.2. Mixed arithmetic/geometrical link progression

This scheme as its name indicates is a mixture of arithmetic and geometric progression. Starting from the top row, output ports are

Table 1
Number of routers' output links per side for a network of 64 CRs, various (I_A, S_L) combinations and using the arithmetic link progression scheme.

Row	I_A	2	2	2	2	4	4	4	4	6	6	6	6	MFT
	S_L	0	1	2	3	0	1	2	3	0	1	2	3	
5		1	1	1	1	1	1	1	1	1	1	1	1	1
4		2	2	2	2	3	3	3	3	4	4	4	4	3
3		3	3	3	**3**	5	5	5	5	7	7	7	7	7
2		4	4	4	3	7	7	7	5	10	10	10	7	15
1		5	5	4	3	9	9	7	5	13	13	10	7	31
0		6	5	4	3	11	9	7	5	16	13	10	7	63



NoC Size (CRs)	4	8	16	32	64	128
Maximum link length (in terms of CR length)	1	2	2	4	4	8

Fig. 3. The H-tree construction of the NoC floor plan (not to scale). The client/NoC interface (CNI) is at the edge of the NoC at the client's interface to the 1st level of routers (row 0).

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

Table 2
Number of routers' output links per side for a network of 64 CRs, various (I_A, S_L) combinations and using the mixed arithmetic/geometrical link progression scheme.

Row	I_A / S_L	2 1	2 2	2 3	4 1	4 2	4 3	6 1	6 2	6 3	MFT
5		1	1	1	1	1	1	1	1	1	1
4		2	2	2	3	3	3	4	4	4	3
3		3	3	3	5	5	5	7	7	7	7
2		4	4	**7**	7	7	11	10	10	15	15
1		5	9	15	9	15	23	13	21	31	31
0		11	19	31	19	31	47	27	43	63	63

incremented by I_A until the row S_L is reached. After that, doubling of the links is implemented resulting in number of links in between that of the arithmetic progression and the original MFT. This scheme is meant to increase the number of links at the lower rows because in a typical placement, clients are placed adjacent to other clients that they communicate with the most. Hence most of the traffic will be routed through lower levels. Table 2 shows this link progression scheme for a NoC with 64 CRs for various combinations of I_A and S_L.

4.1.3. Controlled geometrical progression

Based on the assumption that more links are needed at lower rows, this scheme keeps the number of links constant (with no progression at all) for upper rows and start doubling them at level S_L. This results in a much lower number of links and the overall gate count.

4.2. Modified router architecture

As was explained before, the routers in the original (see Fig. 2) were designed with no possibility of contention due to the geometrical link progression. With the above progression schemes contention may arise and hence requires modification of the routers' architecture. Since several input ports are going to be eventually competing for fewer output ports crossbar structures have to be inserted on both the left and the right paths to manage the allocation of the output ports. Fig. 4 shows the router architecture modified to handle contention. Two crossbars have been added to arbiter contentious allocation of the output ports. The two crossbars are independent from one another, simplifying the design of the arbitration control. Routing, computed in the input port logic, determines to which crossbar the request for output port allocation is sent. Unlike regular crossbars, all output ports are considered as one single path and only a status bit (allocated/unallocated) is associated with output ports. Requests are queued in a first come first serve basis using simple logic that maintains the order of arrival of the requests. A request queue maintains this priority. It is a simple FIFO with a size equal to the difference between the number of input ports and output ports of the crossbar. Each entry is designed to hold the index of the corresponding input port that issued the request. Because all output ports are identical resources, there is no need to specify which output port is requested.

4.3. The client network interface (CNI)

The MFT NoC is characterized by a relatively high number of links at the edge of network (where it interface to the clients). Buffering has also been pushed to these interfaces. In order to overcome these limitations a new client interface architecture is proposed. This interface aims at reducing the number of parallel FIFOs (from the incoming links) into a single centralized FIFO before interfacing this FIFO to the client. Fig. 5 below shows the block diagram of the CNI. It is made of three parts; an upper part consisting of several bus-widener structures that will be named *parallelizers* from this point forward, a single centralized FIFO memory and a data transfer interface circuit. The outputs of the parallelizers are connected to the central FIFO via a single many-to-one multiplexer. The data transfer interface circuit is required to transfer the data between the two clock domains; the NoC's and the client's. The design of each of these parts is described below.

4.4. The parallelizers

Each one of the parallelizers is made of two layers. The first layer is a collection of registers connected in parallel to the incoming data bus from one of the incoming links (ports). Packet data is received into one of these registers one word at a time. When this layer is full, an entire line made by concatenating all the registers of the first layer is transferred to a second set of registers (the second layer in the parallelizer) in a single clock cycle. The ratio between the width of the parallel bus and the width of a single word is called *the parallelization factor*. Portions of packets from the same source are always received by the same parallelizer in order. Packets from different sources are received on different paral-

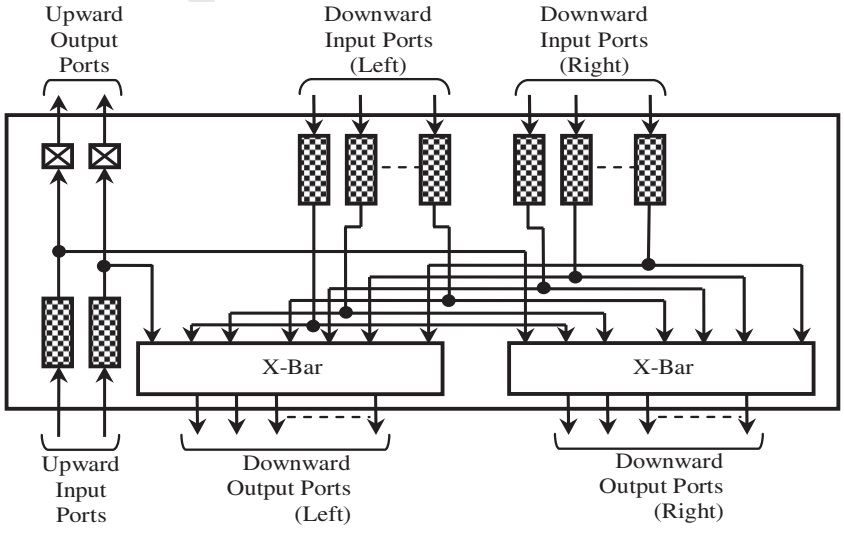


Fig. 4. Router architecture with contention handling.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

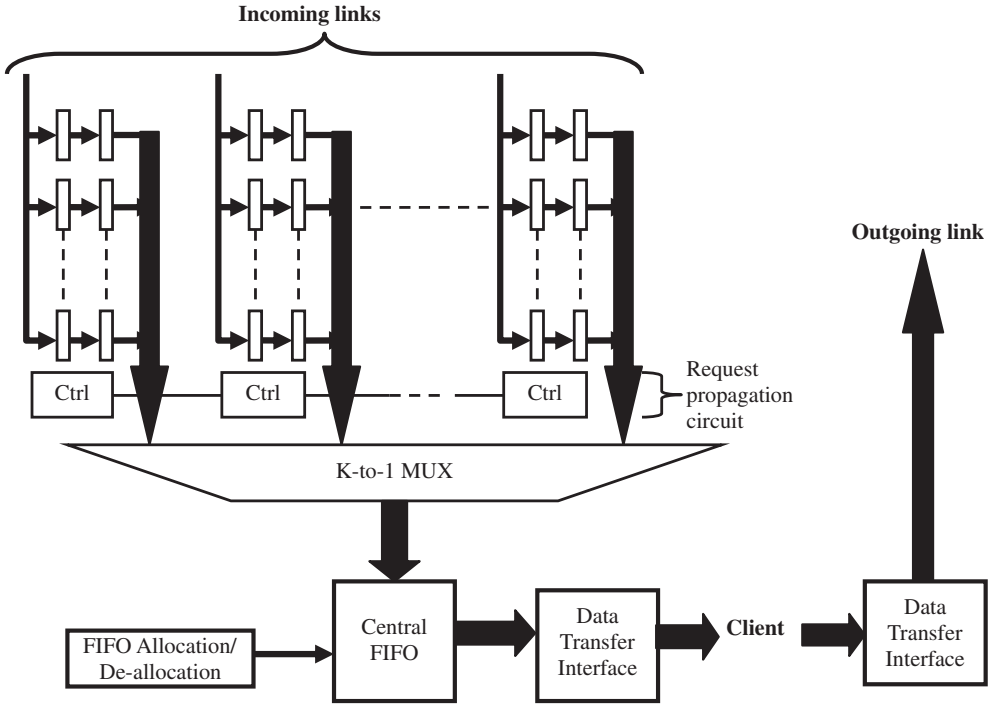


Fig. 5. Block diagram of the client network interface (CNI).

lelizers simultaneously and independently. When the first portion of a packet is received and transferred to the second layer of the parallelizer, a flag is set to request transfer of a new packet to the FIFO. The control logic responsible for these transfers will first attempt to reserve space in the FIFO corresponding to one packet. The condition here for this architecture to produce efficient results is the adoption of a fixed packet size. This condition simplifies the space allocation in the FIFO and alleviates the control logic from any space or fragmentation management due to variable size allocation and disposal. In the case the FIFO is full and no space could be reserved, the request is rejected and the backpressure mechanism is triggered on that requesting port.

The control logic continuously transfers the received packet words to the FIFO. Every time a packet portion enters the second layer of registers in one of the parallelizers a flag is set to indicate the presence of data. Those parallelizers which are currently receiving packets are said to be active. Only active parallelizers are continuously polled to check the presence of data. The polling follows a round-robin policy. A single clock cycle is used to process the currently selected parallelizer. Polling the active parallelizers only supposes some mechanism to "skip" all the non-active parallelizers between two active ones. In order to avoid wasting clock cycles crossing those non-active parallelizers, a special request propagation circuit has been designed. Fig. 6 shows the schematic of this circuit. The upper set of flip-flops correspond to the status flag indicating whether a parallelizer is active or not while the lower one is used to indicate which parallelizer is selected to transfer its data during a given clock cycle. The multiplexers are used to instantly skip the non-active parallelizers.

As a result of this fast polling scheme packets arriving simultaneously on different parallelizers may be received in different order. Packet order from the same source is still guaranteed though because of the buffer-less nature of the network. The absence of buffers means that two consecutive packets cannot be sourced from the same client simultaneously. So if these two packets are destined to the same sink, the second cannot reach the destination before the first one is entirely received. So, even if the first packet is delayed, the second one cannot be sourced. Moreover, the routing scheme is deterministic so two packets originating from the same source that are destined to the same client take the same path. In the extreme case where the packet size is so small that it can fit in the FFs within few routers along the path freeing the source output link (to the first router) and allowing the second packet to enter the network, contention will occur between the two packets for the same output port of a router along the path and the second packet is held back.

4.5. The central FIFO buffer

The central FIFO is organized into a pool of packet slots that can be allocated/freed individually. Each slot has the size of one packet. Different parallelizers, simultaneously active, polled in a round robin fashion using the fast polling circuit result in an out-of-order writing of the different packet portions into the central FIFO. The definition of FIFO is violated here as some packets complete their writing before packets that started being written earlier in time. This requires special handling. Two queues are used to keep track of the slot indexes; (1) the Allocated Queue (AQ) and (2) the Free Queue (FQ). The AQ contains the indices of the allocated slots containing complete packets while the FQ keeps track of the empty slots. A temporary address table, called Writing Table (WT), is used to store the addresses of the packets currently being written into the FIFO. The size of this table is equal to the number of parallelizers and each parallelizer has a corresponding entry in the table. Fig. 7 shows the structure of the central FIFO.

When a new packet portion is received at one of the parallelizers, a request is sent to the central FIFO control logic. If the FQ is empty, backpressure is initiated to stop the network from sending more data into the parallelizer. If the FQ is not empty, the top element in the FQ is popped and sent to the entry corresponding to the requesting parallelizer in the WT. The entry, stored in a register, is divided into two fields. The upper field, called an Index (IDX), is used to store the index slot allocated to the current packet corresponding to the parallelizer it is coming from. The lower field points to the current location in the slot to which the next packet word is to be written. This field, called the Counter (CNT) is initial-

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

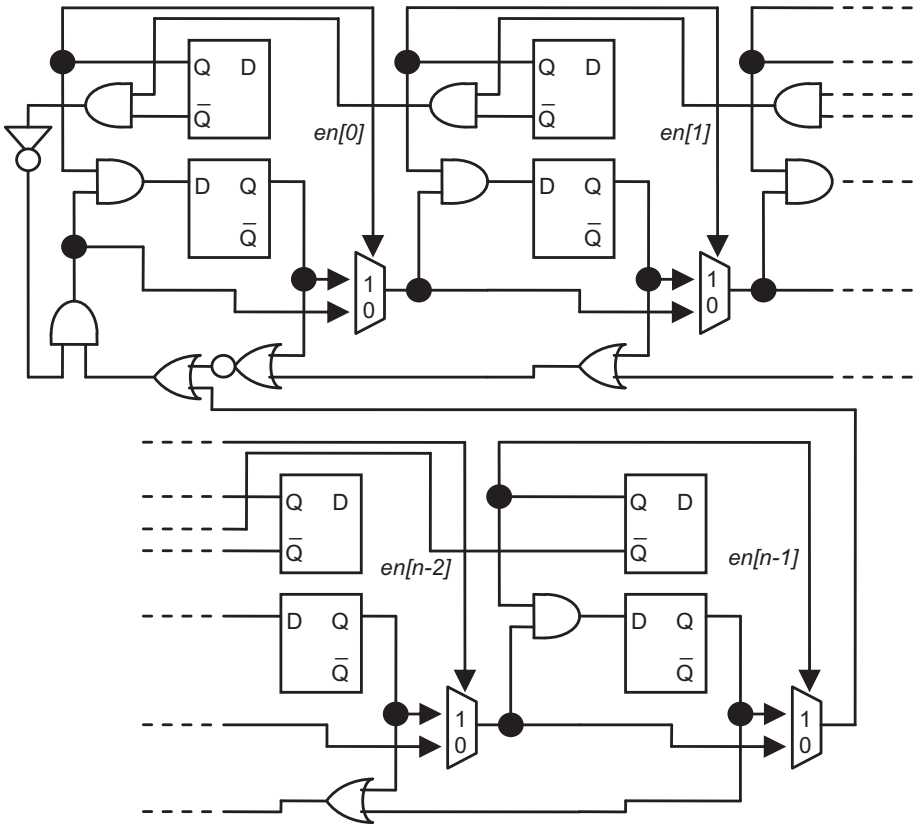


Fig. 6. Schematic of the request propagation circuit at the heart of the intelligent polling scheme.

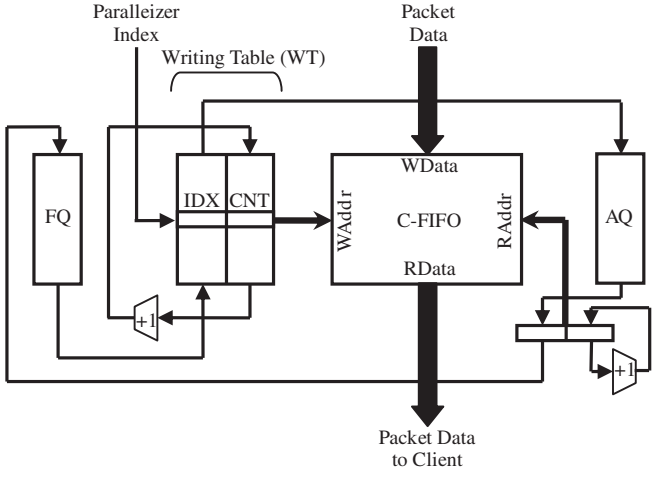


Fig. 7. The structure of the central FIFO.

ized to zero when the slot is allocated. When a new portion of a packet for which a slot has already been allocated reaches the central FIFO, it is written in the location addressed by the corresponding entry in the WT. The CNT field is then incremented by one. When the last portion of the packet is written into the central FIFO, the corresponding entry in the WT is invalidated and the IDX field is pushed at the tail of the AQ.

The client side of the central FIFO logic continuously monitors the contents of the AQ. If at least one entry is present, which means one packet is ready to be sent out, the entry is popped out from the AQ. The IDX field is used to read out the different packet words from the central FIFO. After the packet is completely read out of the FIFO, the IDX field is pushed into the FQ to be reallocated to another packet.

4.6. The data transfer interface (DTI) circuit

There are several ways to transfer data between two clock domains in GALS systems (Globally Asynchronous Locally Synchronous) [27]. With out any assumptions about the two clocks, the asynchronous style represents the least impact on the communicating blocks operation and performance. Due to handshaking and synchronization between the two domains, a datum transfer would take at least three clock cycles of the slower of the two clocks [27]. In [28] throughput was increased to one datum per clock cycle (of the slower of the two clocks) by pipelining the synchronization itself alongside the data. This simple approach of implementing the FIFOs as pipelines greatly reduced the probability of failure due to Metastability and eliminated the need for detecting full/empty conditions. However it increased the latency of the interface since the pipeline has to be filled first before data can come out of it. It also imposed the constraint that the sender and receiver had to operate at the same data rate. A better approach for data transfer between different clock domains based on a general FIFO was proposed in [29]. It allows the sender and receivers to put (or send) and get (or receive) data at their own clock rates simultaneously. In addition for elaborate circuitry for detecting empty/full FIFO conditions, more circuits were added to detect when the FIFO is nearly full or empty. These signals are necessary to maintain the data transfer rates while synchronizing the conventional empty/full signals.

In this work a novel FIFO design for transferring data between two different clock domains was developed. The new FIFO design is simpler than the one in [29] yet it allows independent data send-

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

ing and receiving at different rates (equal to the sender and receiver clock rates, respectively) with no need for empty/full detection. At the heart of this FIFO is a new circuit for transferring data between two clock domains. This basic component is first described then the construction of the FIFO will be shown.

4.7. The basic DTI FIFO stage

Fig. 8 below illustrates the basic circuit and the signaling protocol required to transfer data between a client and the NoC. Data transfer is illustrated for equal client and NoC clock frequencies (CLK_C and CLK_{NoC}, respectively), the worst case condition in terms of the number of cycles required to complete a datum transfer. Two data latches are utilized, one on the client side and another on the NoC side. EN_C and EN_{NoC} are the latch enable signals for putting the data and getting the data. A simple four-phase signaling protocol is used to simplify the circuit design. A conservative two Flip Flop synchronizer design is employed [30]. The client initiates the transfer by setting up the data and raising the PUT signal. The client-side controller would strobe the latch ($EN_C \uparrow$) and initiates a request signal (Req_{Out}). This signal would reach the NoC-side controller as PUT_{Req} after two clock cycle (the synchronization delay). This controller would then strobe the NoC data latch and initiates an acknowledgement signal ($TAKE_{ACK}$). This signal would reach the client-side controller after two more clock cycles which in turn respond by deactivating the request signal. The NoC controller would then deactivate the acknowledge signal, completing the transfer in 8 cycles. If the client or the NoC have higher clock frequency than the other, the transfer would take less number of cycles (the minimum is four). The use of two latches (instead of a single latch or FF as in most FIFOs) per cell achieves two objectives; (1) it greatly simplifies the design by decoupling the PUT (writing to the client's side latch) and GET (reading the NoC's side latch output) operations, (2) it effectively provides a two-stage pipeline per FIFO stage, reducing the impact of clock frequency difference on the PUT/GET rates.

The number of required transfer cycles could have been reduced by overlapping data transfers but this would have resulted in more complex circuitry that would be slower to operate. As will be shown later, a maximum throughput of one datum transfer per cy-

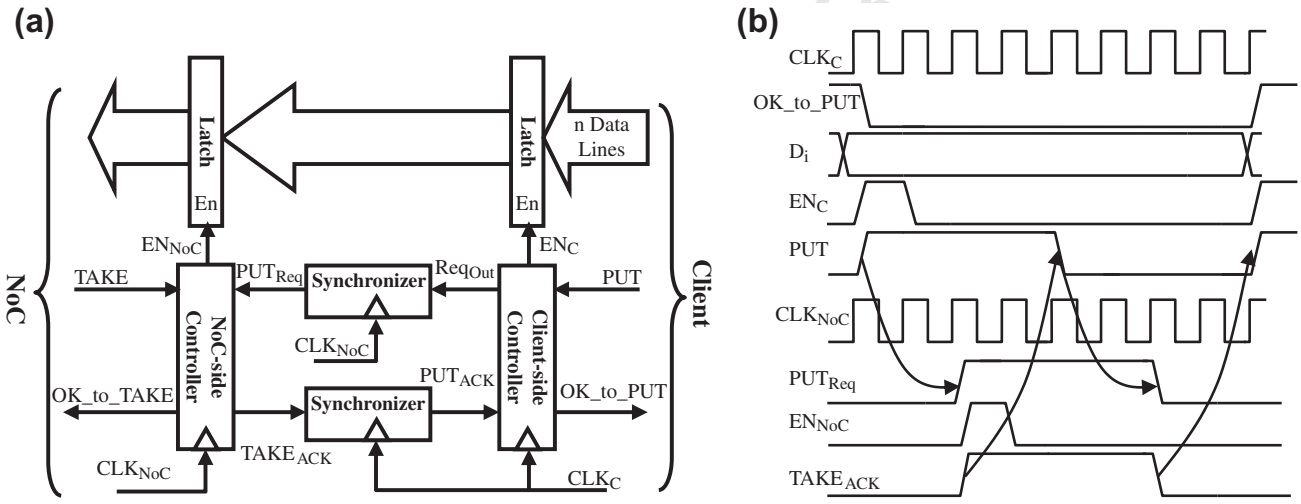
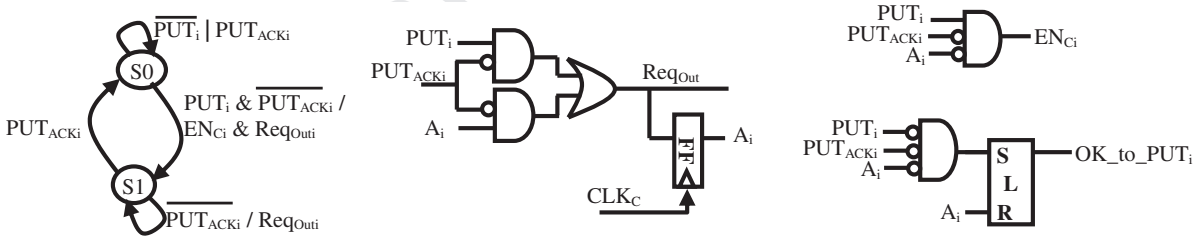
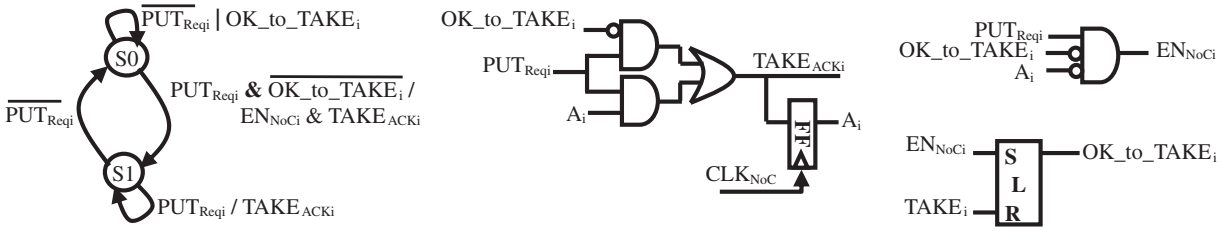


Fig. 8. The proposed circuitry for data transfer between two clock domains; (a) the basic block diagram, (b) the signaling protocol for equal clock frequencies.



(a) The design of the client-side controller.



(b) The design of the NoC-side controller.

Fig. 9. The design of the control circuits of the basic DTI FIFO stage.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

cle with a latency of less than one cycle is still achieved by the proposed FIFO.

Fig. 9 below shows the design of the client/NoC controllers of one of the FIFO's stages (stage *i*). Each controller has a simple two-state FSM implemented with a single FF and simple logic. The client-side controller basically would latch the data into the client-side latch and assert the Req_{Out} signal if it receives a PUT request while the PUT_{ACK} signal is low. The Req_{Out} signal is kept high till the PUT_{ACK} signal goes high. The OK_to_PUT signal is then set when the PUT_{ACK} signal goes back low. This signal is reset when the controller latch new data. It would be clear why an SR-latch is used to set and reset this signal when the FIFO construction is described. The NoC-side controller would latch-in the data when it receives a PUT request while the previous data has been consumed (i.e. when OK_to_TAKE is low). It also asserts the $TAKE_{ACK}$ signal and keeps it high till the PUT request signal goes low. After latching the data the OK_to_TAKE signal is set high. The NoC consumes the data by asserting the TAKE signal which in turn resets the OK_to_TAKE signal.

4.8. DTI FIFO construction

Fig. 10 shows the block diagram of an *n* stage FIFO constructed from the basic data transfer circuit described above. Two counters are used as pointers to the tail of the PUT queue and the head of the GET queue. A ring-connected shift register could be used as a pointer for FIFOs with large number of stages. Input data lines (D_{IN}) are connected to the inputs of all stages on the client side. When a PUT request is received from the client an internal PUT signal (PUT_i) is asserted and routed to the stage selected by the PUT pointer if its OK_to_PUT signal is high. The pointer is also incremented. The SR-latch in the FIFO stage used for the OK_to_PUT signal of the selected stage would reset after one clock cycle, hence the internal PUT signal would not *evaporate* before the end of the cycle. This allows the client to put a data item every cycle (of its own clock) as long as the FIFO is not full (indicated by the OK_to_PUT signal of the tail cell).

On the NoC side a datum is removed every clock cycle from the head of the queue selected by the TAKE pointer if the corresponding OK_to_TAKE signal is high. The NoC can assert the back pressure signal (BP) to stop taking the data. When a data item is taken, the TAKE pointer is incremented to point to the next cell. As explained earlier, it would take 8 cycles to complete a datum transfer within a cell. Data transfer within all cells proceed in parallel, hence using 8-stage FIFO ensures achieving the maximum PUT/TAKE data rates of one datum per clock cycle for any client/NoC clock frequencies.

5. Performance evaluation

To evaluate the performance of the proposed NoC a custom cycle-accurate simulator has been developed. Numerous simulations have been carried out to determine the optimum parallelization factor, central FIFO organization (width and size), and link progres-

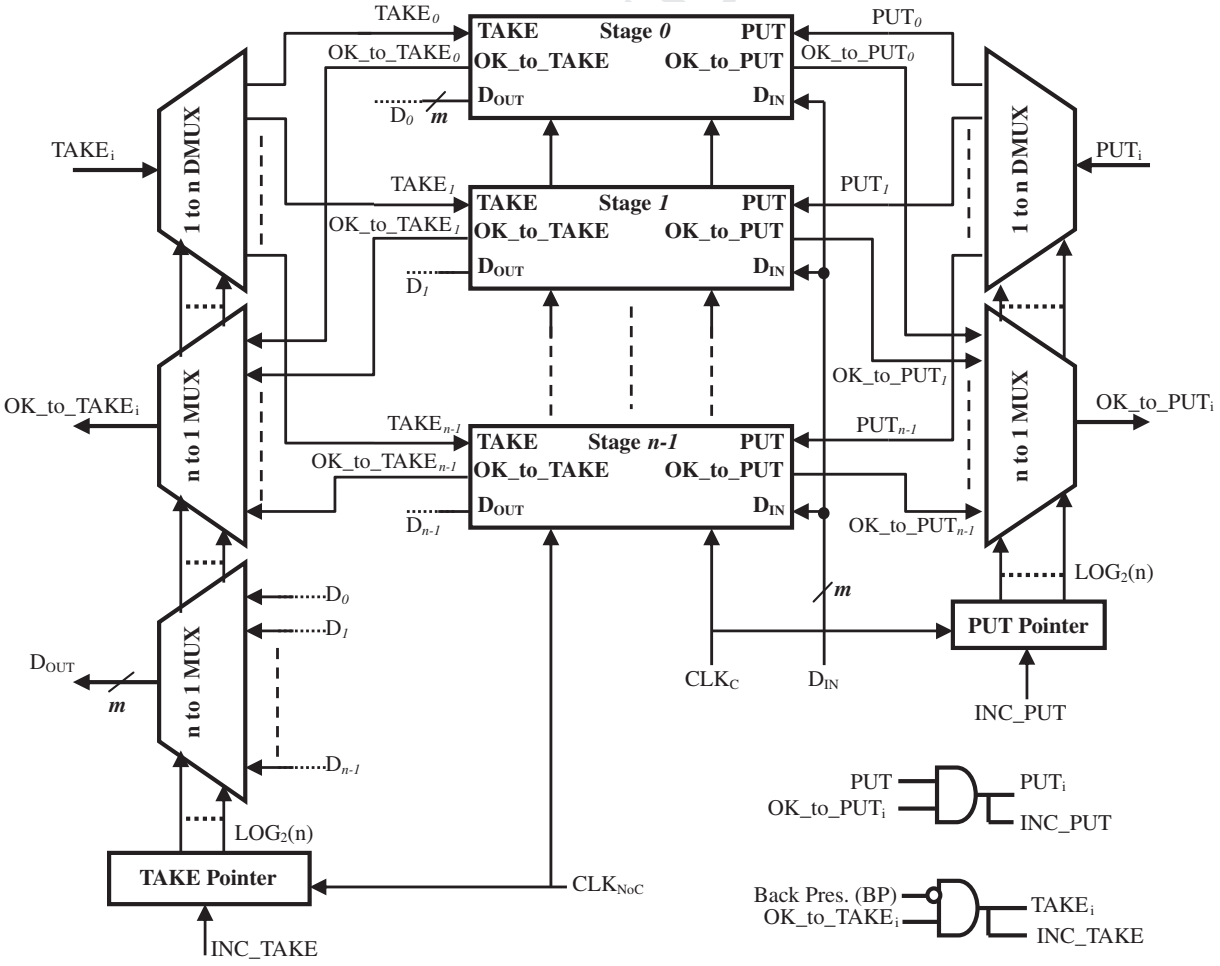


Fig. 10. A Block diagram of an *n* stage FIFO constructed from the basic cells.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

sion scheme for several NoC sizes. Two simple synthetic traffic models have been used; traffic with uniform address distribution and traffic with non-uniform distribution. The latter model assumes that clients that communicate with one another the most are placed adjacent to one another. Hence the probability of a certain destination address is assumed to decrease logarithmically with the 'distance' between the destination address and the source address. The distance between two clients, d, is the order of the minimum group that contains both clients. So the probability of a certain destination address P_A is simply calculated as $P_A = (0.5)^d$ with the imposed constraint that the sum for all destination probabilities equal to 1. A packet's delay is defined as the number of clock cycles from the time a packet is injected into the network until it starts being read out by the receiving client. Throughput is the amount of useful data versus idle time actually delivered by the network per unit of time. It is expressed as a fraction of the maximum bandwidth or wire speed (which is 1 word/cycle/client). In all simulations, a very high injection rate (by all clients simultaneously) was used to stress out the NoC.

5.1. Evaluation of parallelizers' width and central FIFO organization on performance

Several simulations have been carried out to determine the effects of the parallelization factor and central FIFO organization (size and width) on the NoC performance for various NoC sizes for the two traffic models (uniform and non-uniform). Also two packet sizes were considered; 32 bytes and 64 bytes. Figs. 11 and 12 show throughput and maximum delay for various combinations of central FIFO width (or emptying rate, R), size and parallelizer's width (called P on the figure). So an R4P4 value on the x-axis means a FIFO emptying rate of 4 bytes/cycle and a parallelizer's width of 4. For each NoC three FIFO sizes were considered with the two traffic models and two packet sizes (32 and 64 bytes). So a NonU F16 legend means non-uniform traffic and FIFO size of 16 packets. Similarly a U F16 legend means uniform traffic and a FIFO size of 16 packets. These results were obtained using geometrical link progression to maximally stress out the client interface circuitry. It is very clear that a FIFO size of 16 packets, emptying rate of 8 and parallelizer width of 8 is sufficient to obtain the maximum performance under all conditions. Also the larger packet size yields a slightly better throughput. Hence these values ($R = 8$, $P = 8$, FIFO size of 16 packets and packet size of 64 bytes) were used in all subsequent simulations. It should also be noted that for the same FIFO size and emptying rate a larger parallelizer width may actually lead to a slightly reduced throughput (e.g. going from R4P8 to R4P16). This is due to the fact that it would take more cycles to assemble packet fragments in the parallelizers. So when the FIFO is full and backpressure is triggered, it will take longer before the parallelizers get filled and transfer the data to the FIFO.

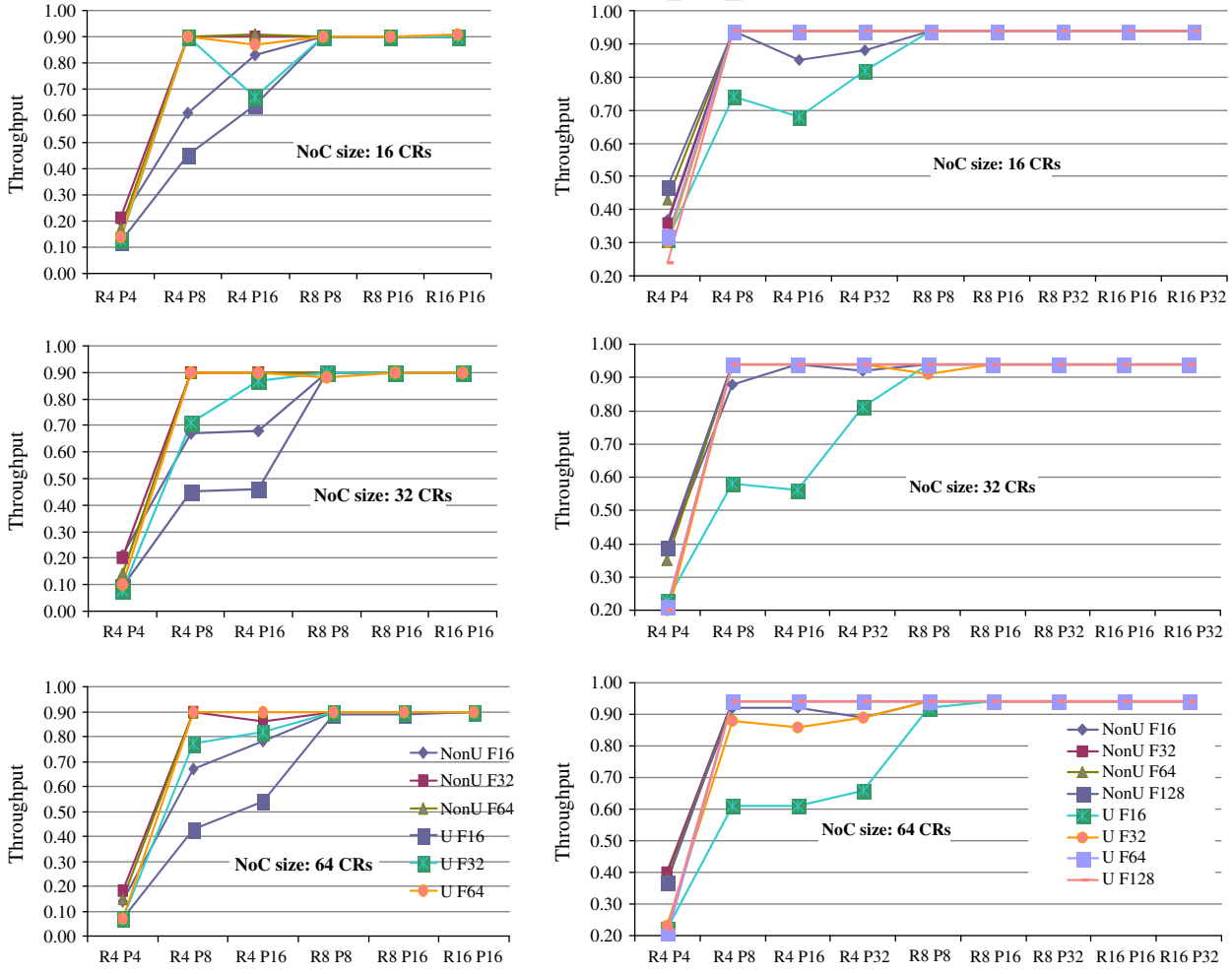


Fig. 11. Measured throughput versus FIFO emptying rate (R) and parallelizer's width (P) for 3 NoC sizes, several FIFO sizes, uniform and non-uniform traffic models. Results on the left are for a packet size of 32 bytes while results on the right are for a packet size of 64 bytes.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

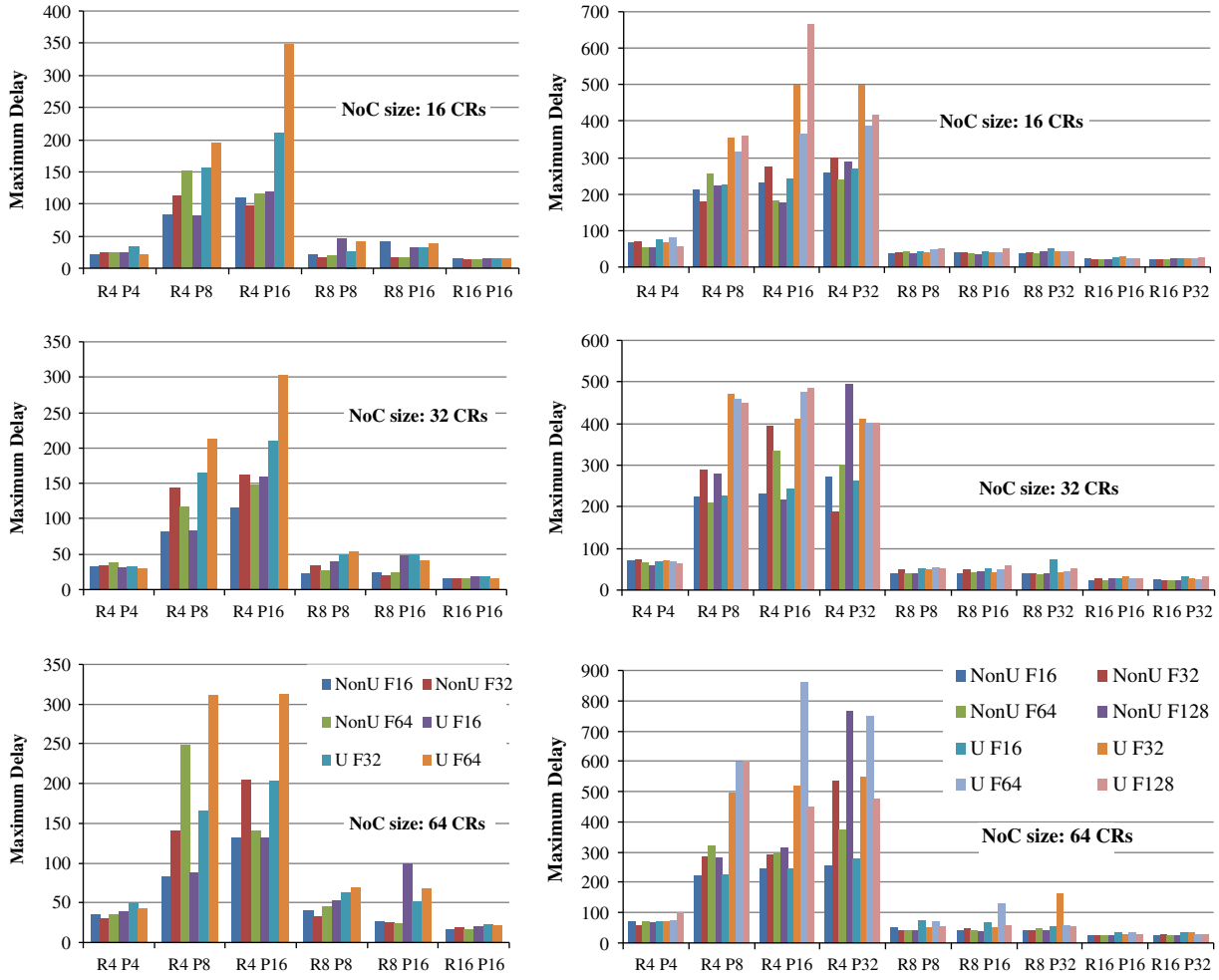


Fig. 12. Measured maximum packet delay versus FIFO emptying rate (R) and parallelizer's width (P) for 3 NoC sizes, several FIFO sizes, uniform and non-uniform traffic models. Results on the left are for a packet size of 32 bytes while results on the right are for a packet size of 64 bytes.

5.2. Evaluation of the different link progression schemes

Using the optimum parallelizer and central FIFO parameters obtained above, numerous simulations were carried out to evaluate the performance of the different link progression schemes described in Section 3. Fig. 13 below shows the throughput for arithmetic and mixed arithmetic/geometrical link progression for a NoC size of 64 CRs for uniform and non-uniform traffic. This Figure clearly shows that for non-uniform traffic where most of the traffic is local, for both schemes, the minimum number of extra links (i.e. $I = 2$ and $S = 4$) would boost the throughput to the maximum possible value. Any other values for I and S would be an over design. As for uniform traffic (where most of the traffic traverse upper rows of the NoC), a minimum increment of 2 combined with slightly lower stopping level of 3 would yield a throughput very close to the maximum while providing huge savings on hardware resources. The average packet delays for these two progression schemes were found to be identical and independent of the link progression

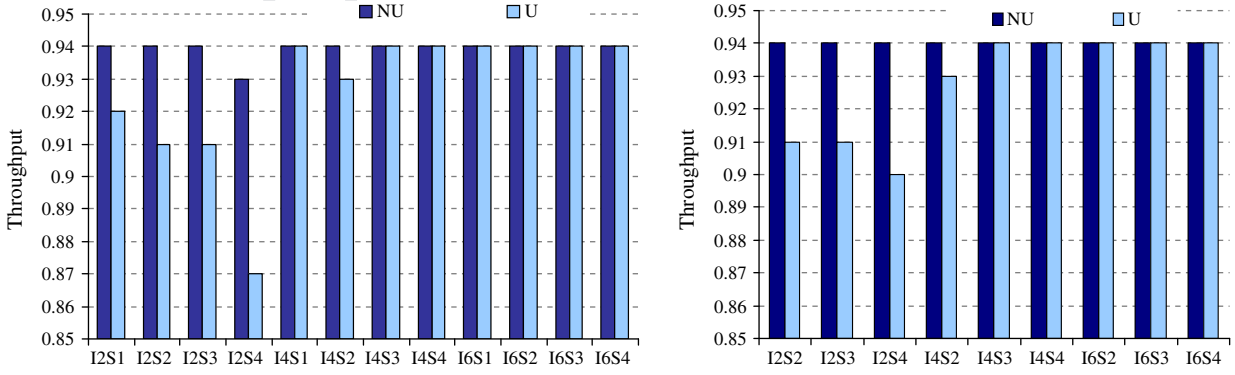


Fig. 13. Throughput as a function of link progression parameters (increment I and stop level S) for arithmetic progression (left) and mixed arithmetic/geometrical progression (right) for uniform (U) and non-uniform (NU) traffic.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

Table 3
Average packet delay for both arithmetic and mixed arithmetic/geometrical link progression schemes and the original geometrical link progression scheme for the two traffic models.

NoC type	Arithmetic or mixed link progression		Original MFT with pure geometrical link progression	
NoC size	Uniform	Non-uniform	Uniform	Non-uniform
16-CRs	15	11.5	14.5	11.5
32-CRS	16.5	12	16	12
64-CRs	18.5	12	18	12

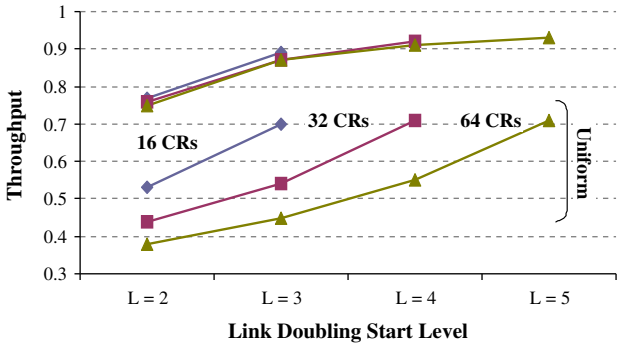


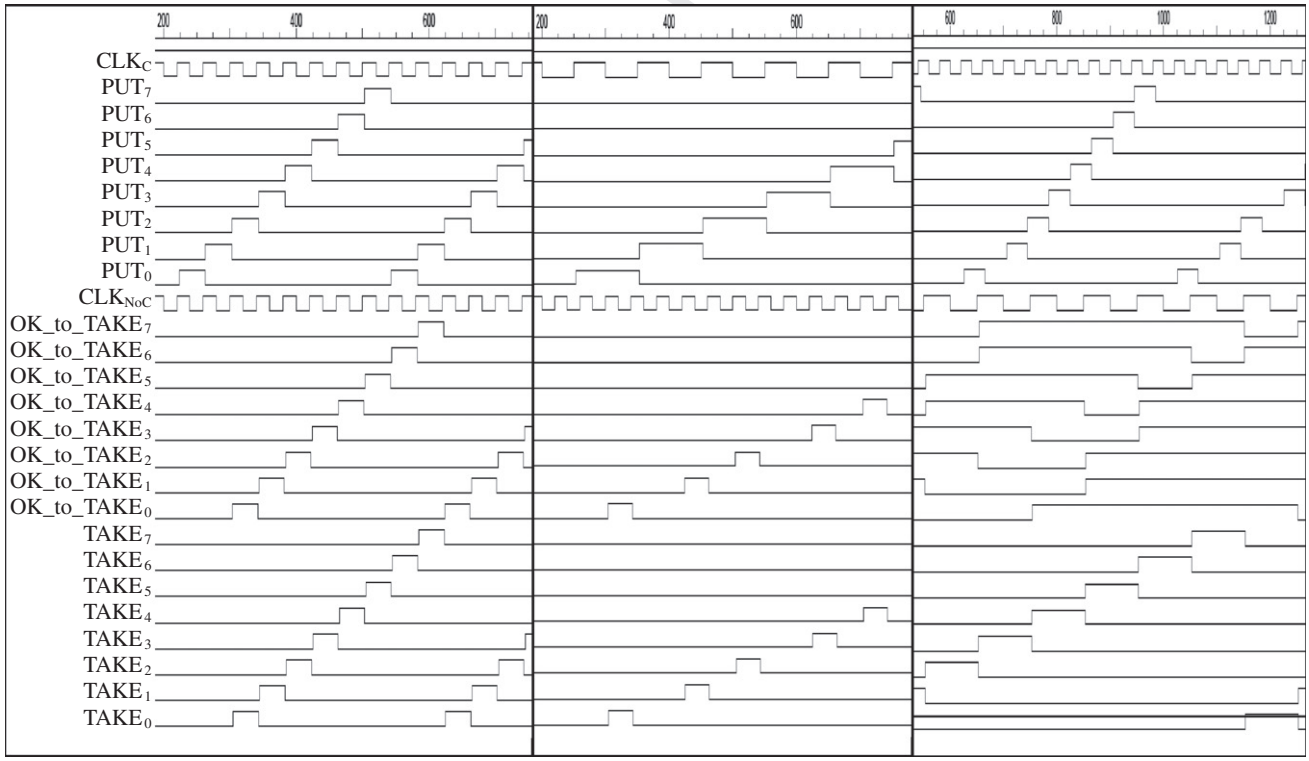
Fig. 14. Throughput versus link doubling start level for the controlled geometrical link progression scheme for three NoC sizes and for uniform (bottom group of curves) and non-uniform traffic.

parameters (increment and stop level). Table 3 shows the average packet delays for these link progression schemes compared to the original geometrical link progression for the two traffic models for several NoC sizes. Also the delays of these schemes are very close to the geometrical progression scheme.

Fig. 14 shows throughput versus link doubling start level for the controlled geometrical link progression scheme for three NoC sizes for the two traffic models. As expected this scheme yielded the lowest throughput for uniform traffic since it has the fewest links in upper rows where most of the traffic is. For the more realistic non-uniform traffic, this progression scheme achieves high throughput (>90%) even for relatively low level of link doubling. As will be shown next this allows the designer to reduce the number of links at the edge of the NoC (and consequently the NoC gate count) significantly while retaining an excellent throughput. Again the average packet delays for this scheme were identical to the other link progression schemes.

5.3. DTI simulation results

The ability of the DTI circuit to transfer data between a client and the NoC at the maximum rate of one datum/cycle is very crucial for the operation of the NoC. To verify the operation of the DTI circuit a gate-level implementation of an 8-stage FIFO was simulated (with unit gate delays) with three Client to NoC clock frequency ratios; 1:1, 1:2.5 and 2.5:1. Fig. 15 shows the simulation results for the three clock ratios alongside one another. The results show that for equal frequencies, both client and NoC are able to put/get a datum per clock cycle. When the NoC's clock frequency is 2.5× that of the client, the client is still able to put data every cycle but the data removal rate by the NoC is automatically reduced by a factor of 2.5 of the NoC clock frequency. When the client's clock frequency is 2.5× that of the NoC, initially when the FIFO is empty, the client is able to put data at the maximum rate. The rate gradually goes down till it reaches 1/2.5 of the client's clock rate. The gradual reduction of the rate is due to two factors; (1) for this



(a) Equal clock frequencies. (b) NoC's clock frequency is 2.5X the client's. (c) Client's clock frequency is 2.5X the NoC's.

Fig. 15. Simulation results of the 8-stage FIFO with three client/NoC clock frequency ratios.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

clock ratio, it takes 4 NoC clock cycles to transfer a datum between the client-side latch to the NoC-side latch. Since the FIFO size is 8 there will be enough time for several cells to complete their data transfers. (2) The inherent pipelining with the cell due to the use of two latches.

5.4. Comparison with a mesh-based NoC

In order to compare the MFT's performance to a classical Mesh NoC, another simulator was developed for simulating Mesh NoCs using simple non-bursty traffic with uniform address generation. Fig. 16 below shows the average latency results for several Mesh sizes. It is very clear that unlike the MFT, these networks start to saturate at very low injection rate (~30%). This is consistent with reported results in the literature for this popular type of NoCs.

5.5. NoC gate count

The total gate count of various sized NoCs for the three link progression schemes have been evaluated using a word size of 8-bits. A special C-code is used to generate the RTL-level Verilog description of routers, parallelizers and central FIFOs under different link progression schemes. These Verilog codes were then automatically synthesized. The special polling circuitry and the DTI were manually designed using basic logic components (gates, MUXs and FFs) to optimize their performance. A break down of gate count is shown below:

1. The router is made of:
   - Upward and downward input ports which have been synthesized using 250 and 260 gates, respectively (mostly FFs)
   - Crossbars that are only present in routers with a number of output ports that are not enough to eliminate contention. These crossbars are considered as series of wide many-to-one multiplexers implemented using pass-gates. They are modeled as 3 2-input NAND-equivalent gates per bit. One multiplexer per output port is needed. The number of input ports depends on the link progression scheme. For example, a crossbar of 6 inputs and 8 outputs with an 8-bits datapath is equivalent to approximately $6 * 8 * 8 * 3 = 1152$ gates
   - Output ports contain minimum amount of logic (mainly for the backpressure signal)
2. Parallelizers are implemented using simple FFs at a gate-equivalency of 4/FF/bit. For example, a 16-word-wide, 2-stage parallelizer of 8 links will amount to $16 \times 2 \times 8 \times 8 \times 4 = 8$ K gates.
3. The central FIFO is implemented using embedded memories where the area taken by a 16 packets memory with a packet size of 64 bytes (8 K-bits) is 0.008 mm^2 in a 0.13 μm technology. This is roughly equivalent to 1.4 K gate in the same technology. The control logic of the central FIFO amounts to about 1 K gates.
4. The total gate count for DTI circuit depends on the width of the central FIFO's data width. For an 8-byte data width the DTI contains 7 K gates.

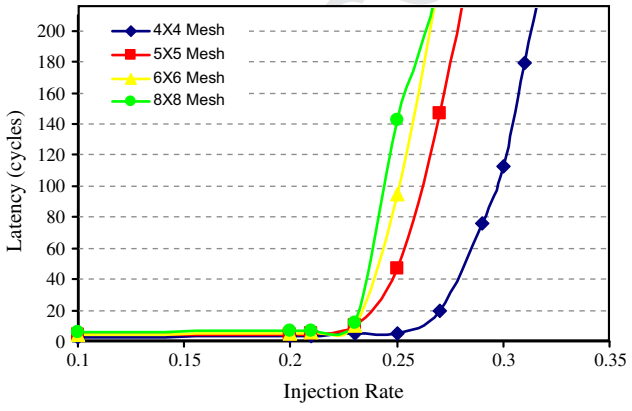


Fig. 16. Average latency versus injection rate for mesh-based network of various sizes.

Table 4 below shows the gate count for arithmetic link progression for various combinations of arithmetic increment I_A and stop level S_L. The gate count for the original MFT is also shown in the table (last column) for comparison). The gate count for the mixed arithmetic/geometrical and the controlled geometrical link progression schemes are shown in Tables 5 and 6, respectively. For all progression schemes, the total RAM size used for central FIFOs is 128 Kb, 256 Kb and 512 Kb for NoC sizes of 16, 32 and 64 CRs, respectively.

Fig. 17 shows a bar chart of the throughput of all link progression schemes versus stop (start) levels for the arithmetic and mixed arithmetic/geometrical (controlled geometrical) link progression schemes for the two traffic models for a 64 CRs NoC. Also shown on the same Figure are the total gate counts for the three schemes (as lines) and the throughput and gate count of the original MFT NoC for reference. As this figure shows, the controlled geometrical progression with a low starting level yields the lowest gate count at an adequate throughput for non-uniform traffic. On the other hand the arithmetic link progression with a stop level of 3 achieved maximum throughput at a ~50% gate count for both types of traffic. It is very clear that the original geometrical and mixed arithmetic/geometrical link progression schemes represent an over design; large gate count at a throughput not more than the arithmetic progression. Based on these results FPGA designers can have different products with different link progression schemes for different applications; controlled geometrical progression (starting at level 2) for applications with localized non-uniform traffic and arithmetic progression targeting applications with more uniform traffic.

5.6. Impact on FPGA resources

To put the numbers of gate count into perspective the impact on FPGA resources has been estimated for two cases of a 64 CRs NoC with central FIFO size of 8 Kb; controlled geometrical progression with starting level of 2 and arithmetic link progression with an arithmetic progression of 2 and stop level of 4. The 1st design provide 75% throughput for non-uniform traffic with minimum gate count of 477 K-gates of logic and 512 Kb of RAM while the second design provide throughputs of 93% and 87% for non-uniform and uniform traffic, respectively at a gate count of 558 K-gates of logic and 512 Kb of RAM. To evaluate the impact of the extra logic and RAM resources required by the NoC on the FPGA resources (slices) the following has been done:

1. To evaluate the area penalty of adding custom logic two Xilinx Virtex-6 FPGAs with identical resources except for the number of custom DSP slices have been considered. The XC6VLX240T and XC6VLX365T [31] FPGAs are identical in every aspect (I/Os, MACs, Transceivers, Block RAMs, etc.) except for the number of custom DSP slices (called DSP48E1 slices) and the number of logic slices. The XC6VLX240T has 192 more DSP48E1 slices than the XC6VLX365T. Accordingly the XC6VLX365T has 19,200 more logic slices than the XC6VLX240T. Hence one can fairly assume that each additional DSP48E1 slice costs 100 logic slices. So if the number of logic gates per DSP slice is estimated, one can estimate the general cost of custom logic. This is a crude method but can still give a good estimate of the cost of

Table 4
Gate count (in K gates) for several NoC implementations with various parameters for arithmetic link progression.

NoC type	MFT with arithmetic link progression											MFT
NoC size	$I_A = 2, S_L = 1$	$I_A = 2, S_L = 2$	$I_A = 2, S_L = 3$	$I_A = 2, S_L = 4$	$I_A = 4, S_L = 1$	$I_A = 4, S_L = 2$	$I_A = 4, S_L = 3$	$I_A = 4, S_L = 4$	$I_A = 6, S_L = 1$	$I_A = 6, S_L = 2$	$I_A = 6, S_L = 3$	$I_A = 6, S_L = 4$	
16-CRs	106	93	73	–	165	139	102	–	200	245	131	–	194
32-CRS	299	274	230	–	547	488	381	–	856	756	561	–	781
64-CRs	833	782	686	558	1657	1529	1275	946	2726	2497	2025	1420	3147

adding custom logic. According to the data sheet of these FPGAs [31], each DSP48E1 slice contains the following; 25 × 18-bit two's complement multiplier, a 48-bit accumulator, multiplier bypass logic, a 48-bit add/subtract arithmetic unit and a logic unit that can generate any one of 10 different logic functions of the two 48-bit operands. The DSP48E1 includes an additional pre-adder, and the multiplier can perform logic functions (AND, OR) and barrel shifting. Assuming that the multiplier is using a modified Booth algorithm that saves area and enhance speed [32] it would require a minimum of 2614 gates (30 gates for encoding and sign extension, 2200 gates for partial products generation and adder array, and 384 gates for the final 48-bit adder). The pre-adder and bypass logic would require at least 624 gates, the logic unit would require a minimum of 528 gates and the accumulator needs about 480 gates. The total (ignoring any logic required to do the barrel shifting by the multiplier) amounts to 4246 gates. This means that adding 42.46 gates of custom logic will require the elimination of a single FPGA logic slice. Hence, to add one of the above NoC implementations to a Virtex-6 FPGA would require the elimination of about 11,200 and 13,100 logic slices for the controlled geometrical progression and arithmetic progression NoCs, respectively. For an XC6VLX760 Virtex-6 FPGA with 118,500 slices, this amounts to 9.45% and 11% reduction of logic resources.

2. Considering the same LX760 Virtex-6 FPGA, the central FIFOs would consume 512 Kb of the FPGA's 25,920 Kb block RAMs, which is less than 2%.

Hence, according to these area approximations, to add a fairly large sized NoC (64-clients) as a hardware macro to a state-of-the-art FPGA around 10% of the logic resources and 2% for the block RAMs will have to be sacrificed.

5.7. CR size selection

Selecting the appropriate size of a CR in terms of logic slices represents a major design trade-off. As the number of CRs increases (hence each CR will have smaller number of logic slices), the CR fragmentation is reduced but the NoC area will grow and more logic slices have to be eliminated. Hence the number of CRs should be determined by the class of applications targeted by the FPGA. For example if the average number of logic slices required by clients is 2000 with up to 20 clients per implementation, then a 32-CRs NoC with 2000 logic slice per CR could be used resulting in an 64 K slice FPGA. Assuming arithmetic link progression with increment of 2 and stop level of 2, the NoC would occupy an area equivalent to 5600 logic slices and consume 256 Kb of the block RAMs. So an existing FPGA design with about 120,000 slices (e.g. Xilinx XC6VLX760 Virtex-6 FPGA) could be re-designed to achieve the above. This is the trend followed by FPGA vendors for some time now; providing FPGAs with various flavors for different classes of applications (general logic, DSP applications, embedded processors or the so called platform FPGAs, etc.). The NoC will be just another variation on top of these basic options with FPGAs ranging from having no NoC at all to having NoCs with various sizes.

5.8. FPGA implementation of the MFT NoC

Two NoCs with full geometrical progression (i.e. link doubling from one level to the next) have been mapped to two FPGAs; a 32-client NoC on a Xilinx Virtex-6 XC6LX760 and a 16-client NoC on a Virtex-4 XC4LX200 to show the area/speed of the NoC if implemented as soft macros. Highest speed grades were used for both implementations. The first NoC has a total of 80 routers, 2240 links, parallelizers (31 per client) and 32 central FIFOs. The second NoC has a total of 32 routers, 608 links, 240 parallelizers

Table 5
Gate count (in K gates) for the mixed arithmetic/geometrical link progression scheme.

NoC type	MFT with mixed arithmetic/geometrical link progression									MFT
NoC size	$I_A = 2, S_L = 2$	$I_A = 2, S_L = 3$	$I_A = 2, S_L = 4$	$I_A = 4, S_L = 2$	$I_A = 4, S_L = 3$	$I_A = 4, S_L = 4$	$I_A = 6, S_L = 2$	$I_A = 6, S_L = 3$	$I_A = 6, S_L = 4$	
16-CRs	117	152	–	173	199	–	231	246	–	194
32-CRS	327	427	–	537	638	–	770	854	–	781
64-CRs	890	1146	1636	1585	1862	2464	2403	2628	3296	3147

Table 6
Gate count (in K gates) for the limited geometrical link progression scheme.

NoC type	MFT with controlled geometrical link progression				MFT
NoC size	$S_L = 1$	$S_L = 2$	$S_L = 3$	$S_L = 4$	
16-CRs	64	104	–	–	194
32-CRS	142	222	401	–	781
64-CRs	317	477	836	1593	3147

(15 per client) and 16 central FIFOs. These are fairly large NoCs (due to geometrical progression) that would be implemented using 781 K and 194 K Gates of custom logic and 256 Kb and 128 Kb of RAM. The central FIFOs were implemented using the FPGAs' BRAMs which are 18 Kb each. Each NoC was implemented twice; once with the parallelizers implemented using slice FFs and another with BRAMs. Table 7 shows the area and speed results for these two NoCs alongside several NoC implementations on FPGAs.

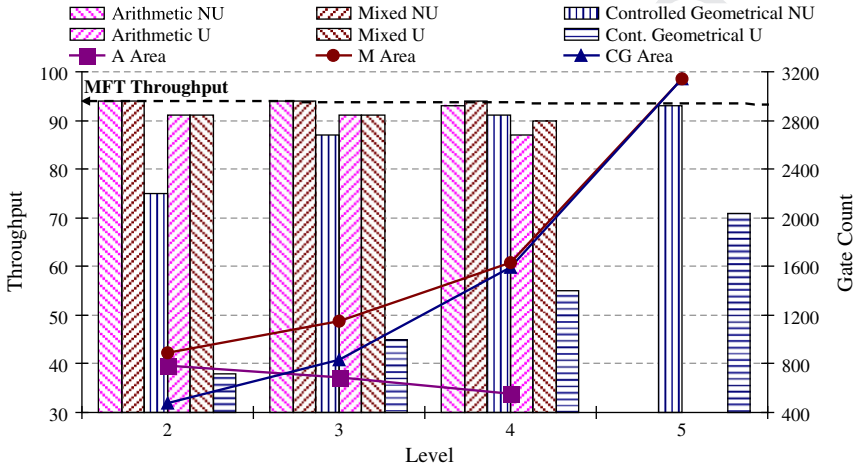


Fig. 17. Throughput and total gate count for a 64 CRs NoC versus stop level of the arithmetic and mixed arithmetic/geometrical progression (start level for the controlled geometrical) link progression for uniform (U) and non-uniform (NU) traffic.

Table 7
Area and speed results for the soft implementation of the MFT NoC. Area is reported as number of LUTs used and their percentage of available. The same is done with BRAM. Resources have been normalized for 8-bit data width.

NoC	Resources	Total LUTs		FFs only		BRAM		Freq.
		Complete NoC	Per client	Complete NoC	Per client	Complete NoC	Per client	
32-client MFT on Virtex-6 (LX760) with 5-i/p LUTs	With FF parallelizers	139,409 (29.4%)	4356 (0.9%)	84,928 (17.9%)	2654 (0.6%)	576 Kb (2.2%)	18 Kb (0.07%)	349 MHz
	With BRAM parallelizers	82,497 (17.4%)	2578 (0.5%)	26,032 (5.5%)	814 (0.2%)	18,432 Kb (71%)	576 Kb (2.2%)	341 MHz
16-client MFT on Virtex-4 (LX200) with 4-i/p LUTs	With FF parallelizers	21,133 (12%)	1321 (0.8%)	25,743 (14.4%)	1609 (0.9%)	288 Kb (4.8%)	18 Kb (0.3%)	211 MHz
	With BRAM parallelizers	20,623 (11.5%)	1289 (0.7%)	9007 (5%)	563 (0.3%)	4608 Kb (76%)	288 Kb (4.8%)	210 MHz
5-clients on Virtex-4 (LX200ff) with 4-i/p LUTs [24]	With FF FIFOs	7716 (4.33%)	1544 (0.9%)	–		–		118 MHz
	With BRAM FIFOs	4278 (2.4%)	856 (0.5%)	–		–		122 MHz
5-ports router on Altera Stratix EP1S80 [26][a]	Only the router	550 4-i/p LUTs		–		–		123 MHz
4-clients on Virtex-2 XCV800 with 4-i/p LUTs [16][a]	With BRAMs in routers	3227 (34.8%)	807 (8.7%)	–		–		40 MHz
5-ports router on Virtex-2 Pro (XC2VP30) with 4-i/p LUTs [17][a]	With BRAMs in router	772 (2.57%)		–		180 Kb		32.25 MHz
9-clients on Virtex-2 Pro 40 with 4-i/p LUTs [20][a]	With BRAMs in routers	6110 (14%)	679 (1.6%)	–		594 Kb (17%)	66 Kb (1.9%)	50 MHz

[a] Clients network interfaces are not included.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008

Table 8
Area (gate count) and speed results for the HW implementation of a 16-client MFT NoC compared to other HW NoCs. Resources have been normalized for 8-bit data width.

	Total Gate count		RAM		Freq.
NoC	Complete NoC	Per client	Complete NoC	Per client	
16-client MFT using 90-nm technology[b]	194 K gates 216 K gates	12.1 K gates 13.5 K gates	128 Kb Included in equivalent gates	8 Kb	800 MHz
5-clients using 90-nm technology [24][c]	77 K Gates	15.4 K gates	Included in equivalent gates		500 MHz
5-ports router using 180-nm technology [26][a]	1500		Not including buffer RAM and its control		340 MHz
9-clients using 130-nm technology [20][a,b]	1.1 M gates	122 K gates	Included in equivalent gates		200 MHz

[a] Clients network interfaces are not included.
[b] Speed estimated as 4× that of the FPGA implementation as in [33].
[c] Gate count estimated by dividing total area in mm^2 by the area of a minimum size 2-ip NAND gate.

Resources are reported as total used and per client since different implementations implemented different NoC sizes. Also, for the published soft NoCs the number of LUTs were normalized for a data width of 8-bits by simple division. Many of the reported implementations do not report the detailed usage of BRAMs (like [16] where BRAMs are used in the routers). Many researchers just implement the routers without the clients' network interfaces. Resources are reported as total numbers and percentage of available resources. Although these percentages do not mean much because they depend on the FPGA used, they are reported since they were reported in the original papers.

Although the MFT NoC was not intended for soft implementation it is still comparable to other soft NoCs in resource utilization and superior in speed as the results in Table 7 show.

5.9. Speed and area comparison with other HW NoCs

Table 8 shows the area (gate count) and speed estimation for the 16-client MFT NoC with geometrical link progression alongside several HW NoCs from the literature. The performance of the MFT NoC is estimated based on the Virtex-4 FPGA implementation to be at least between 800 and 1000 MHz [33] for a 90-nm technology (the Virtex-4 technology). The gate count is given for two instances; one with central FIFOs expressed in RAM Kbs and another with the RAM converted to their gate equivalent for comparison with other HW NoCs. This table shows that the MFT NoC is very efficient in area and superior in performance even if the other HW NoC speed is assumed to increase by 50% from one technology generation to the next.

6. Conclusions

A new hardwired NoC based on a modified Fat Tree topology for future FPGAs has been developed along with an efficient H-tree floor plan that naturally follows the construction methodology. The performance of the proposed NoC has been evaluated for three link progression schemes for various NoC sizes and traffic models. Simulations result showed that for uniform traffic, arithmetic link progression would yield best performance with moderate gate count. For more localized traffic (non-uniform), the controlled geometrical progression can provide adequate performance at the lowest gate count. As for the client interface, a new efficient design have been developed that requires a single central FIFO buffer combined with link-termination circuitry (parallelizers) that provide temporary storage for incoming data words. An intelligent round robin circuit was also developed to transfer packet data from parallelizers to the central FIFO. The design of these interface circuitry was optimized through extensive simulations. These simulations showed that a parallelization factor of 8 combined with a FIFO size of 16 packets and emptying rate (i.e. width) of 8 words are enough to achieve maximum throughput and minimum packet delay. Simulations also showed that performance is fairly independent of packet size. Finally a new interface circuit that transfer data between the NoC and clients operating at different (and unrelated) clock frequencies has been developed. Unlike the asynchronous interfaces used by other NoCs, this circuit allows the NoC and clients to communicate synchronously at the maximum rate of 1 datum/cycle no matter what is the clock frequency ratio between them. Due to its asynchronous nature, the operation of this circuit was verified with gate-level simulations. Both soft and hard wired implementations of the MFT NoC show superior performance over reported NoCs at an area cost that is comparable or better than other NoCs.

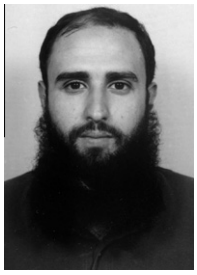
Acknowledgement

This work was supported by King Fahd University of Petroleum and Minerals (KFUPM) Grant # IN070367.

References

[1] T. Dorta, J. Jiménez, J.L. Martín, U. Bidarte, A. Astarloa, Overview of FPGA-based multiprocessor systems, in: Proc. 2009 International Conference on Reconfigurable Computing and FPGAs, 2009, pp. 273–278.
[2] W. Webb, FPGAs Reshape Embedded Design, EDN Magazine, March 2009. <www.edn.com/article/CA6643363.html>.
[3] Xilinx Virtex-5 FPGA Configuration Guide, UG191 (v3.8), 2008.
[4] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, D. Lindqvist, Network on chip: an architecture for billion transistor era, in: Proc. IEEE NorChip Conf., 2000.
[5] W.J. Dally, B. Towles, Route packets, not wires: on chip interconnection networks, in: Proc. 38th Design Automation Con., 2001, pp. 684–689.
[6] J. Henkel, W. Wolf, S. Chakradhar, On-chip networks: a scalable, communication-centric embedded system design paradigm, in: Proc. 17th Int. Conf. VLSI Design, 2004, pp. 845–851.
[7] E. Nilsson and J. Öberg, Reducing power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking, in: Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS'04), 2004, pp. 176–181.
[8] E. Nilsson and J. Öberg, Trading off power versus latency using GPLS clocking in 2D-mesh NoCs, in: Proc. Int. Sym. On Signals, Circuits and Systems (ISSCS), 2005, pp. 51–54.
[9] V. Soteriou, H. Wang, L. Peh, A statistical traffic model for on-chip interconnection networks, in: Proc. of the 14th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06), 2006, pp. 104–116.
[10] E. Nilsson, Design and Implementation of a Hot-potato Switch in a Network on Chip, Master's Thesis, Royal Institute of Technology, IMIT/LECS 2002–2011, Sweden, June 2002.
[11] K. Goossens, J. Dielissen, A. Radulescu, Æthereal network on chip: concepts, architectures, and implementations, IEEE Design and Test of Computers 22 (5) (2005) 414–421.
[12] A. Bouhraoua and M. Elrabaa, A high-throughput network-on-chip architecture for systems-on-chip interconnect, in: Proc. of the Int. Symp. on System-on-Chip (SOC06), 2006, pp. 1–4.
[13] A. Bouhraoua, M. Elrabaa, An efficient network-on-chip architecture based on the fat tree (FT) topology, Arabian Journal of Science and Engineering (AJSE) 32 (2C) (2007) 13–26 (special issue on microelectronics).
[14] C. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers C-34 (10) (1985) 892–901.

[15] A. Bohraoua and M. Elrabaa, Addressing heterogeneous bandwidth requirements in modified fat-tree networks-on-chips, in: Proc. of the 4th IEEE Int. Sym. on Electronic Design, Test & Applications (DELTA'2008), 2008, pp. 486–490.
[16] T. Marescaux, et al., Networks on chip as hardware components of an OS for reconfigurable systems, in: Proc. 13th Int. Con. on Field Programmable Logic and Applications (FPL), 2003, pp. 595–605.
[17] B. Sethuraman, P. Bhattacharya, J. Khan, R. Vemuri, LiPaR: A light weight parallel router for FPGA based networks on chip, in: Proc. 15th ACM Great Lakes Symp. on VLSI, 2005, pp. 452–457.
[18] N. Kapre, et al., Packet switched vs. time multiplexed FPGA overlay networks, in: Proc. 14th Annual Int. IEEE Sym. on Field-Programmable Custom Computing Machines (FCCM), 2006, pp. 205–216.
[19] T. Pionteck, et al., Applying partial reconfiguration to networks-on-chips, in: Proc. 16th Int. Con. on Field Programmable Logic and Applications (FPL), 2006, pp. 1–6.
[20] T. Bartic, J.-Y. Mignolet, T. Marescaux, D. Verkest, S. Vernalde, R. Lauwereins, Highly scalable network-on-chip for reconfigurable systems, in: Proc. 2003 IEEE Int. Symp. On System-on-Chip (SOC03), 2003, pp.79–82.
[21] R. Hecht, S. Kubisch, A. Herrholtz, D. Timmermann, Dynamic reconfiguration with hardwired networks-on-chip on future FPGAs, in: Proc.14th Int. Conf. on Field Programmable Logic and Applications (FPL), 2005, pp. 527–530.
[22] R. Gindin, I. Cidon, I. Keidar, NoC-based FPGA: architecture and routing, in: Proc. 1st ACM/IEEE Int. Sym. on Networks-on-Chip (NOCs'07), 2007, pp. 253–264.
[23] H. Elmiligi, M. El-Kharashi, F. Gebali, Introducing OperaNP: a reconfigurable NoC-based platform, in: Proc. 2007 IEEE Canadian Conf. on Electrical and Computer Engineering, 2007, pp. 940–943.
[24] K. Goossens, M. Bennebroek, J.-Y. Hur, M. Wahlah, Hardwired networks on chip in FPGAs to unify functional and configuration interconnects, in: Proc. 2nd ACM/IEEE Int. Sym. on Networks-on-Chip (NOCs'08), 2008, pp. 45–54.
[25] M. Wahlah, K. Goossens, 3-Tier reconfiguration model for FPGAs using hardwired network on chip, in: Proc. Int. Con. on Field Programmable Technology (FPT), 2009, pp. 504–509.
[26] R. Pau, N. Manjikianm, Implementation of a configurable router for embedded network-on-chip support in FPGAs, in: Proc.6th NEWCAS-TAISA Con., 2008, pp. 25–28.
[27] P. Teehan, M. Greenstreet, G. Lemieux, A survey and taxonomy of GALS design styles, IEEE Design & Test of Computers 24 (5) (2007) 418–428.
[28] J. Seizovic, Pipeline synchronization, in: Proc. Int. Symp. On Advanced Research in Asynchronous Circuits and Systems (ASYNC 94), 1994, pp. 87–96.
[29] T. Chelcea, S. Nowick, Robust interfaces for mixed timing systems, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 12 (8) (2004) 857–873.
[30] R. Ginosar, Fourteen ways to fool your synchronizer, in: Proc. of 9th Int. Symp. On Asynchronous Circuits and Systems (ASYNC'03), 2003, pp. 1–8.
[31] http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf.
[32] M. Elrabaa, I. Abu-Khater, M. Elmasry, Advanced Low-Power Digital Circuit Techniques, Kluwer Academic Publishers, 1997.
[33] I. Kuon and J. Rose, Measuring the gap between FPGAs and ASICs, in: Proc. of 15th Int. Symp. IEEE Field Programmable Gate Arrays (FPL), 2006, pp. 21–30.



Muhammad E. S. Elrabaa received M.A.Sc. and PhD degrees in Electrical & Computer Engineering from the University of Waterloo, Waterloo, Canada, in 1991 and 1995, respectively. From 1995 till 1998, he worked as a senior component designer with Intel Corp., Portland, Oregon, USA. He designed and developed low power digital circuits for Microprocessors. He is currently an associate professor with the computer Engineering department, KFUPM University. His current research interests include Networks-on-chip, defect tolerant circuit techniques and reconfigurable computing. He authored and co-authored numerous papers, a book and holds two US patents.

Abdelhafidh Bouhraoua is an Assistant Professor at the Computer Engineering Dept., KFUPM, Saudi Arabia. Dr. Bouhraoua completed his Bs. in Computer Science and Engineering from the National Institute of Informatics in Algiers, Algeria in 1989. After completing his Bs; he joined the Ministry of Scientific Research, Algeria, where he contributed in the design and VLSI integration of a RISC microprocessor. He, then, joined the VLSI/System Architecture group (ASIM) at the University Of Paris, France, where he completed his Ms. and PhD degrees, respectively in 1993 and 1998. During his PhD, Dr. Bouhraoua contributed to the design and performance evaluation of the interconnection network that was the core of a PC-based network of workstations. Prior to joining KFUPM, Dr. Bouhraoua was working as a Senior ASIC Designer for numerous North American Fabless ASIC design companies like Applied Micro Circuits Corporation and Zarlink Semiconductor where he was designing ASICs for various telecom systems. Currently, Dr. Bouhraoua research interests are in SoC design methodologies, NoCs design; reconfigurable architectures; FPGA systems; embedded systems, robotics and mechatronics.

Please cite this article in press as: M.E.S. Elrabaa, A. Bouhraoua, A hardwired NoC infrastructure for embedded systems on FPGAs, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.09.008