Poznań University of Technology

Faculty of Electronics and Telecommunications

Chair of Multimedia Telecommunications
and Microelectronics

Doctoral Dissertation

Architecture and protocols for networks-on-chip implemented in FPGA devices

(Architektura i protokoły dla sieci w mikroukładzie realizowanej w bezpośrednio programowalnych macierzach bramek)

Jakub Siast

Supervisor: Prof. dr hab. inż. Marek Domański

Poznań, 2019

I owe many thanks to my colleague, Adam Łuczak who has offered me his time and inspired me with many good ideas.

I would like to thank professor Marek Domański, for his time and help that have guided me towards completing this dissertation.

I am grateful to my colleagues from the Chair of Multimedia Telecommunications and Microelectronics for all the inspiring years we have spent together. Furthermore, I need to thank Krzysztof Klimaszewski, Dawid Mieloch, and Adrian Dziembowski, for careful reading of the manuscript.

This dissertation is dedicated to my beloved parents: Elżbieta and Tadeusz, who have supported me unconditionally.

Moje podziękowania należą się Adamowi Łuczakowi, który poświęcił mi swój czas i zainspirował mnie dobrymi pomysłami.

Chciałbym również podziękować panu profesorowi Markowi Domańskiemu, za jego czas i pomoc, która doprowadziła mnie do ukończenia tej rozprawy.

Jestem wdzięczny kolegom i koleżankom z Katedry Telekomunikacji Multimedialnej i Mikroelektroniki za inspirujące, wspólnie spędzone lata. Ponadto dziękuję Krzysztofowi Klimaszewskiemu, Dawidowi Mielochowi i Adrianowi Dziembowskiemu za wnikliwe przeczytanie tekstu rozprawy.

Rozprawę tę dedykuję moim ukochanym rodzicom: Elżbiecie i Tadeuszowi, którzy zawsze mnie wspierali.

Table of contents

Abstract

In the dissertation, the problems related to the design of networks-on-chip (NoCs) on field-programmable gate arrays (FPGAs) are considered in the context of typical features of FPGAs manufactured by leading vendors. As a result of these considerations, the RingNet architecture and communication protocol are proposed with the aim to exploit the specific potential of FPGA devices as much as possible. Although a few FPGA-oriented NoCs have been proposed so far, the RingNet design is likely the most determinedly adapted to typical FPGA resources and their architectures. The specific features of RingNet include: communication exclusively through system memory (large SDRAM or block RAM), control over traffic load executed by the processing elements, FPGA-optimized 3-port switches organized into the tree-of-rings topology, distributed memory (LUTRAM) used as small buffers in the switches, and virtual cut-through switching.

In the dissertation, it is demonstrated that RingNet offers guaranteed throughput, predictable latency, and fair network access. The synthesis results demonstrate that RingNet implementations are efficient in terms of maximum clock frequency and resource consumption for flagship FPGA devices from major manufacturers. As compared to the widely-accepted state-of-the-art interconnection architecture AXI4 Interconnect, RingNet implementations demonstrate higher maximum clock frequency and lower resource consumption. Therefore, the author believes that the RingNet NoC architecture and protocol may be widely adopted in FPGA-based SoC designs, especially in high-volume data processing applications, such as video processing.

Streszczenie

W rozprawie poruszono zagadnienie sieci w mikroukładzie (ang. network-on-chip, NoC) w kontekście właściwości typowych bezpośrednio programowalnych macierzy bramek (ang. *field-programmable gate array*, *FPGA*) oferowanych przez czołowych producentów. Wynikiem rozważań jest architektura RingNet oraz odpowiedni protokół komunikacyjny, opracowane z myślą o możliwie największym wykorzystaniu potencjału układów FPGA. Dotychczas, kilka sieci NoC dedykowanych układom FPGA zostało zaproponowanych w literaturze, jednak to sieć RingNet jest prawdopodobnie najbardziej dopasowana do architektury i zasobów typowego układu FPGA. Cechami charakterystycznymi RingNet są: komunikacja prowadzona wyłącznie poprzez pamięć systemową (podłączoną do układu FPGA pamięć SDRAM lub wbudowaną pamięć blokową), kontrola nad ruchem w sieci sterowana przez urządzenia podłączone do sieci (ang. *processing elements*), zoptymalizowane pod układy FPGA trójbramowe przełączniki sieciowe zorganizowane w topologię drzewa pierścieni, pamięć rozproszona (LUTRAM) użyta do implementacji małych buforów w przełącznikach, oraz zastosowanie przełączania *virtual cut-through*.

W rozprawie zademonstrowano takie właściwości RingNet jak: gwarantowaną przepustowość, dające się przewidzieć opóźnienie i sprawiedliwy dostęp do sieci. Analiza implementacji sieci RingNet zademonstrowała jej wysoką wydajność rozumianą jako wysoką częstotliwość sygnału taktującego i niskie zużycie zasobów dla flagowych układów FPGA głównych producentów. Implementacja została porównana z powszechnie stosowaną architekturą komunikacyjną AXI4, co wykazało wyższą maksymalną częstotliwość taktowania i mniejsze zużycie zasobów na korzyść RingNet. Z tego powodu autor wierzy, że architektura i protokół sieć RingNet mogą być powszechnie wykorzystane w systemach opartych o układy programowalne, w szczególności w aplikacjach przetwarzania dużej ilości danych, takich jak przetwarzanie sekwencji wizyjnych.

List of terms and abbreviations

ALUT	*Adaptive Look Up Table* In Intel devices, a structure that consists of a LUT associated with a two-bit adder.
ALM	*Adaptive Logic Module* In Intel devices, a logic block that consists of ALUTs and FFs.
AMBA	*Advanced Microcontroller Bus Architecture* On-chip interconnect specification from ARM Ltd.
ASIC	*Application-Specific Integrated Circuit*
AVC	*Advanced Video Coding* A digital video-compression codec, also known as H.264.
AXI4	*Advanced eXtensible Interface 4* A version of the AMBA interface recommended for FPGAs by ARM Ltd.
BEN	*Byte Enable Bits* Bits that indicate the validity of associated data.
BRAM	*Block Random-Access Memory* In FPGA devices, a block of memory with capacity from a few kb to tens of Mb.
DDR3	*Double Data Rate type 3* Type of SDRAM.
DSP	*Digital Signal Processing block*
EDA	*Electronic Design Automation*
FF	*Flip-Flop* In FPGA devices, it is configurable as D-type or latch.
Fifo	*First-In-First-Out buffer*
Flit	A portion of data usually transferred at one clock cycle between connected switches.
FPGA	*Field-Programmable Gate Array*

HEVC	*High Efficiency Video Coding* A digital video-compression codec.
IEEE	*Institute of Electrical and Electronics Engineers*
L2R	*Leaf-to-Root* In RingNet, one of two physical channels.
LAB	*Logic Array Block* In Intel devices, a structure that consists of 10 Adaptive Logic Modules (ALMs).
LC	*Logic Cell* Also known as logic element (LE). It is equivalent to a LUT4 paired with an FF. The capacity of an FPGA is usually described in terms of LCs, even if LUT4s are not used in a given FPGA.
LE	*Logic Element* Also known as a logic cell (cf. LC).
LI	*Leaf Interface* In RingNet, one of two types of a network interface.
Logic block	Basic structure in an FPGA device that contains LUTs, FFs, and additional elements like line multiplexers, and carry logic. It is connected to other logic blocks and to an FPGA network of programmable signal pathways.
LUT	*Look Up Table* In FPGAs, it implements a logic function with a limited number of logic inputs.
LUT4	4-input LUT.
LUT6	6-input LUT.
LUTRAM	*LUT-based RAM* In FPGA devices, a block of LUT-based memory with a capacity from tens to hundreds of bytes, also known as distributed RAM.
NoC	*Network-On-Chip*
PE	*Processing Element* A functional module of an SoC, e.g., processor, controller, grabber.
PFU	*Programmable Functional Unit* In Lattice devices, a structure that consists of four logic blocks.

PG	*Packet Generator* Simulation model of PE.
PLL	*Phase-Locked Loop*
QDR SRAM	*Quad Data Rate Static RAM*
R2L	*Root-to-Leaf* In RingNet, one of two physical channels.
RAM	*Random-Access Memory*
Reflector	In RingNet, one of the system buffers dedicated for control messages.
RI	*Root Interface* In RingNet, one of two types of a network interface.
RLDRAM	*Reduced Latency Dynamic Random-Access Memory* DRAM device featured with SRAM-like interface.
RTL	*Register-Transfer Level* Level of digital circuit description at which the flow of signals between hardware registers are defined.
SDRAM	*Synchronous Dynamic RAM* E.g., DDR3 SDRAM.
SG	*Slot Generator* In RingNet, a module used for the purpose of generating slots for flits and packets.
Slice	In Xilinx and Lattice devices, a structure that consists of LUTs and FFs. In Xilinx devices, it contains four logic blocks with two LUT6s and two FFs each. In Lattice devices it contains one logic block with two LUT4 and up to two FFs. A corresponding structure in FPGAs from Intel is called ALM.
SLICEL	In Xilinx devices, a type of slice that does not contain RAM-capable LUTs.
SLICEM	In Xilinx devices, a type of slice that contains RAM-capable LUTs.
SoC	*System-On-Chip*
VC	*Virtual Channel*
VHDL	*Very High Speed Integrated Circuits Hardware Description Language*

VLSI *Very-Large-Scale Integration*

Chapter 1. Introduction

1.1. Scope of the dissertation

In 1958, a first semiconductor integrated circuit (IC) was presented, composed of a single transistor and other passive components [Kil00]. Ever since, the density of transistors in an integrated circuit has been increased, which led to the development of circuits of very-large-scale integration (VLSI) with more than 20,000 transistors in the 1970s [Mea80] and many more today. Those circuits integrate many processing elements (PEs) into a system-on-chip (SoC). PEs in SoCs require interconnections. Like many other aspects of VLSI design, the interconnections constitute an important research topic. At the beginning, on-chip interconnections successfully utilized design-specific point-to-point wiring. Nevertheless, as the complexity of the circuits increased, developing design-specific wiring became difficult and suffered from long connections with poor physical parameters, e.g., high and unpredictable cross-talks, and wiring delays longer than a clock cycle. The design-specific point-to-point approach towards on-chip interconnection design had to change.

As early as in 1999, it was demonstrated that for multiprocessors, interconnections organized as a regular network can be more efficient than a collection of point-to-point dedicated wires [Dal99]. Starting from 2001, the idea of utilizing the network approach to the interconnection problem in SoCs attracted a lot of interest. Early works [Dal01], [Ben02], [Ben06] proved important advantages of network-on-chip (NoC) when compared with design-specific wiring. The most important advantages are the following:

- Controlled physical parameters, i.e., a network is divided into links of controlled length, featured with low and predictable cross-talks and short wiring delays.

- Modularity, i.e., a standard interface is defined that can be reused. The interface can be optimized, since the development effort can be amortized across many SoCs.

- Effort to test, upgrade, and extend modular SoC that uses a standard interface can be reduced.

- Utilization of links can be higher in networks than in point-to-point design specific interconnects. In an SoC, many PEs are interconnected and while part of them are temporally idle, others can utilize links in a network.

So far, most of the research on NoCs has been related to application-specific integrated circuits (ASICs) [Hel15] but the rapidly growing field-programmable gate array (FPGA) capacity also yields a growing interest in NoCs implemented in complex FPGAs. Using NoCs as interconnections for FPGAs is not a new idea [Łucz11], [Pap12], [She14], [Ret14], [Mai15], [Pap15], [Kap15], [Was17], [Kap17a], [Kap17b], [Kap17c], [Vip17], [Mai17], [Sid18], [Ahm18], [Red19]. Although FPGAs differ from their ASIC counterparts, most of the known FPGA NoCs were adopted from ASICs without considering FPGA-specific features. Therefore, the potential of NoCs has not been fully exploited for FPGAs. In this situation, older interconnecting techniques like crossbars (e.g., AXI4 Interconnect) are still in use, regardless of their poor scalability [Mai15]. The development of new NoC architectures, better suited to FPGA, is still a challenging problem.

The growing FPGA capacity and parallel processing capabilities make FPGAs a useful platform for multimedia processing. Moreover, FPGAs give engineers the ability to develop their systems much faster than in ASICs, which is desired under strong time-to-market pressure. Therefore, a lot of multimedia processing is done in FPGAs. A few examples of multimedia processing considered suitable for FPGAs are: medical imaging filtering [Lic18], MVC (Multiview Video Coding) coder [Stę10b], HEVC coding [Buk17], disparity estimation for stereo vision [Dom15], [Tto16], features detection in videos [Cha15], channel coding for multimedia transmissions [Bre17], audio beamforming and audio wave field synthesis [The11]. Many more can be found in literature. All those applications may benefit from the manner of interconnecting the processing elements in a way that is well-suited to the properties of a specific FPGA structure seen in modern integrated circuits of this kind.

In this dissertation, the author proposes a novel NoC architecture called RingNet that is well-suited to the features of contemporary FPGAs and can be adopted in multimedia processing systems.

1.2. Motivation

The idea for the research described in the dissertation comes from the observed lack of a universal FPGA-oriented NoC suitable for multimedia applications. The observation is a result of the years spent developing applications for multimedia processing and FPGA

implementations during the work at the Chair of Multimedia Telecommunications and Microelectronics.

The implementation of multimedia processing in FPGAs has a long history in the Chair of Multimedia Telecommunications and Microelectronics. Works aiming at the implementation of components for low bit rate, integrated video encoders and decoders on FPGA platform took place from 2004 to 2007 and were led by Prof. Marek Domański [Stę06a], [Stę06b], [Gar06], [Stanki07]. The results of this project were sold to the industry. An NoC for MPEG-4 AVC / H.264 hardware decoder was implemented at the Chair of Multimedia Telecommunications and Microelectronics in 2008 [Łucz08]. In 2009, an NoC-based processing platform was developed [Łucz09] during the work led by Dr. Adam Łuczak.

The author joined the Chair in 2010 and took part in further development of NoC for multi-chip systems [Stę10b], [Łucz11], and NoC with embedded debugging [Stę10a]. In 2010, the author proposed a communication interface for FPGAs [Łucz10]. In her 2013 PhD dissertation, Marta Stępniewska, a member of the Chair of Multimedia Telecommunications and Microelectronics, discussed the problem of defining the connection architecture suitable for video codecs implemented in FPGAs.

The author has rich personal experience in multimedia processing. In 2010-2011, the author took part in HEVC 3D codec implementation [Dom11a], [Dom11b]. The developed codec has been submitted to the "Call for Proposals on 3D Video Coding Technology" issued by the Motion Picture Experts Group (MPEG) of ISO/ITU in 2011 [MP11]. This proposal has been rated very high among other proposals and was found to be one of the best performing proposals in the HEVC category. The excellent results were described in IEEE Transactions on Image Processing [Dom13a] and in [Dom13b]. In addition to the mentioned work submitted to the call for proposals, the author took part in other works undertaken by MPEG [Weg12a]–[Weg12f], [Stanki12a]–[Stanki12c], [Dom13a], [Dom14b], and proposed a coding improvement in 3D-HTM test codec [Sia12], accepted by the group. The author has contributed to many papers focused on multimedia processing, including papers describing multiview and 3D coding tools [Stanko12], [Dom12a]–[Dom12c], analysis of encoded video streams [Stanko14], [Stanko15], video tracking algorithms [Łucz14], and free-viewpoint video acquisition [Dom14a]. The author is the main contributor to the watermarking algorithm [Sia13], as well as to HEVC coder improvement technique [Sia16]. The author was granted two patents by USPTO, the first of

which, published in 2014, describes an image coding method [Dom14c], and the second, published in 2017, describes a system and a method for tracking objects in video sequences [Łucz17]. The author is the main contributor to a method and a system for video signal encoding and decoding with motion estimation, for which EPO granted a patent in 2015 [Sia14].

The experience in multimedia processing and in FPGA implementations resulted in the development of an original wireless multi-camera system in the Chair of Multimedia Telecommunications and Microelectronics [Dom14a], that uses FPGAs as the main processing platform. Video streaming and processing for the system required a high throughput interconnect for FPGAs. A dedicated interconnect has been implemented for the project. The author was one of the main engineers implementing the interconnect. The multi-camera system has been used for capturing multiview sequences. Despite the success of the project and the utilized dedicated interconnect, the project showed a lack of a universal FPGA-oriented NoC suitable for multimedia applications.

Based on extensive experience in multimedia processing and previous experience in FPGA NoC implementations [Stę10a], [Stę10b], [Łucz11] the author, under the supervision of Dr. Adam Łuczak, started the development toward a universal FPGA-oriented NoC suitable for multimedia applications. The dissertation summarizes the achievements of the work. The achievements have been previously presented by the author, Dr. Adam Łuczak, and Professor Marek Domański in the paper "RingNet: A Memory-Oriented Network-On-Chip Designed for FPGA," published in the Institute of Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integrated (VLSI) Systems in 2019 [Sia19].

1.3. Goals and theses of the dissertation

The main goal of the dissertation is to propose a universal NoC architecture suitable for various FPGAs.

With the aim of proposing such an NoC architecture, other goals need to be achieved beforehand. First, the features of FPGAs need to be investigated. Especially, the common features of available FPGAs, useful for NoC implementation, need to be identified. A universal FPGA-oriented NoC should perform similarly for all considered FPGA architectures. The performance of an FPGA-based design can be defined by the number of utilized FPGA resources and the maximum frequency it can be clocked at. The frequency performance can be

examined in the context of the maximum clocking frequency for hardware modules available in FPGAs, e.g., digital signal processing (DSP) blocks and random-access memory (RAM) blocks. Exploiting the identified FPGA common features should help keep the performance of an NoC at a similar, highest possible level for various FPGA architectures. Second, the requirements for NoC need to be explored and interconnections already proposed and known from the literature need to be investigated. A useful NoC should provide features like controlled throughput, network access fairness, etc. Various techniques aimed at providing those features are proposed in the literature and they should be considered in the context of FPGA.

Theses of the dissertation are as follows:

T1) It is possible to develop a network-on-chip architecture and protocol featured with controlled throughput, network access fairness, and a maximum clock frequency higher than 90% of the maximum clock frequency of FPGA hardware resources, across FPGAs of leading vendors.

T2) It is possible to develop a network-on-chip architecture and protocol with controlled throughput and network access fairness for FPGAs which would use less resources and would be featured with a higher maximum clock frequency than the state-of-the-art crossbar (AXI4 Interconnect).

1.4. Overview of the dissertation

In this dissertation, the author proposes a novel NoC architecture called RingNet and its protocol that are well-suited to the features of contemporary FPGAs. Among other NoC architectures proposed for FPGAs, RingNet stands out with communication through a central memory and traffic load controlled by the recipient. The dissertation starts with a discussion on the common features of FPGAs that are important for NoC architectures (Chapter 2). In Chapter 3, general requirements for NoCs are discussed. Next, in Chapter 4, the author summarizes the state-of-the-art in NoC designs for FPGA and points out NoC design constraints that match the FPGA features considered in Chapter 3. In Chapter 5, a novel RingNet architecture of NoC is proposed with its protocol described in Chapter 6. Through Chapters 7 – 9, experimental results are presented, RingNet synthesis for different FPGAs are compared, and RingNet implementation is compared with the widely-used crossbar interconnection called AXI4

Interconnect. In Chapter 10, the author summarizes the dissertation and discuss applications of RingNet.

Chapter 2. Study of FPGA devices from the point of view of intra-device communication

In this section, an architecture of a modern FPGA device is presented. A simplified scheme of a typical FPGA layout is depicted in Fig. II.1.

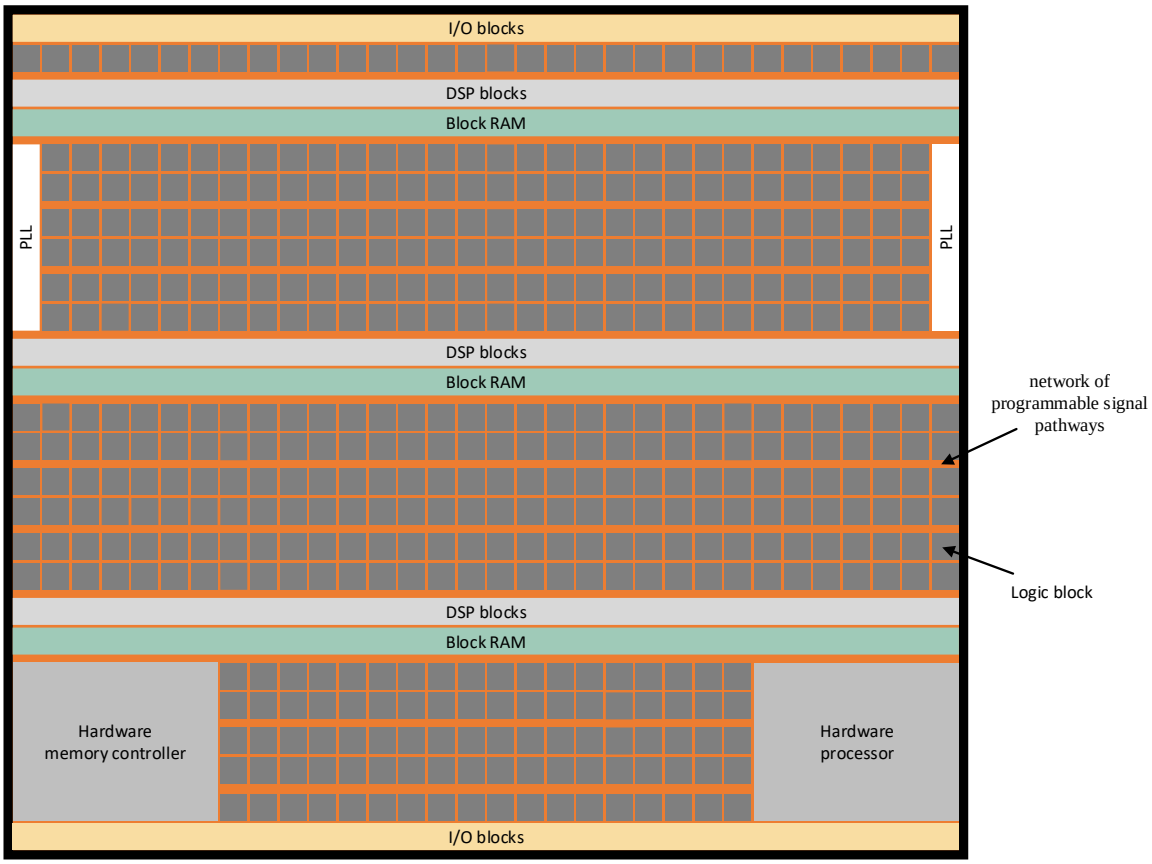


Fig. II.1. Simplified scheme of a typical FPGA layout.

The primary resource of an FPGA device is an array of logic blocks surrounded by or interleaved with input and output (I/O) blocks [Luu16]. The logic blocks may be interleaved with additional types of resources, e.g., digital signal processing (DSP) blocks, blocks of random-access memory (BRAM), phase-locked loop (PLL) blocks, hardware processors, hardware memory controllers, etc. The backbone of every FPGA architecture is a predefined network of programmable signal pathways, which interconnects all the blocks [Mar16]. Clock distribution tree, not depicted in Fig. II.1, delivers low-skew, high fanout clock signals to all synchronous elements in the FPGA fabric.

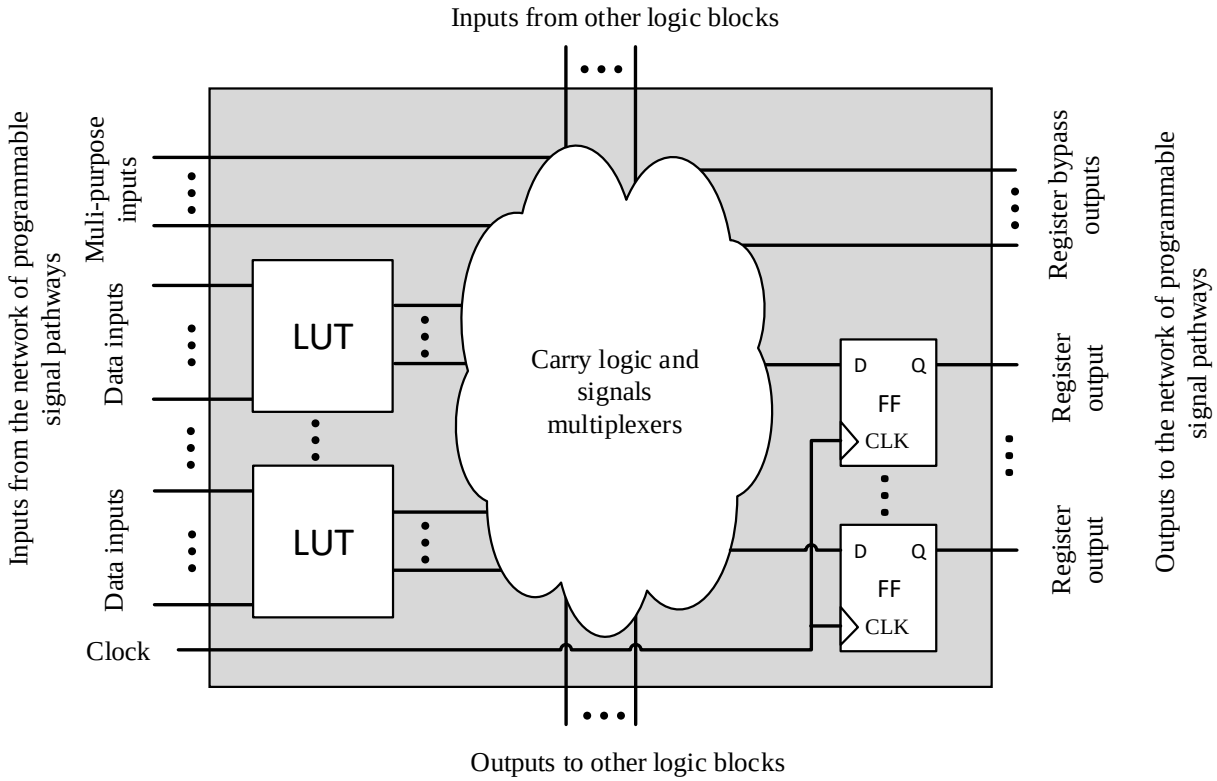


Fig. II.2. Simplified scheme of connections in a logic block.

As depicted in Fig. II.2, the logic block contains look-up tables (LUTs) and flip-flops (FFs). LUT implements a logic function defined by a user, whereas FF is used to store the logic function result. LUT has a limited number of logic inputs, four or six being a common value for a modern FPGA architecture. In order to increase the functionality of their logic blocks, vendors usually enrich them with additional hardware, e.g., input lines that bypass LUT, or output lines that bypass FFs, input and output lines that connect neighboring logic blocks, carry logic useful for implementing arithmetic functions, and programmable multiplexers for signal switching. The multiplexer is used to combine outputs from many LUTs, which is used when a logic function with more inputs that are supported by a single LUT is implemented. In some architectures, a single LUT can implement a function with multi-bit result, therefore LUT may have more than one output line.

Different FPGA architectures use logic blocks with different numbers of LUTs and FFs and may be called differently, e.g., a logic block in FPGAs manufactured by Microsemi has a single LUT and single FF and is called a logic element [Mic18a], [Mic18b], a logic block in FPGA manufactured by Lattice is equipped with two LUTs and two FFs and is called a slice [Lat13]–

[Lat16], whereas Intel uses logic blocks called Adaptive Logic Modules (ALMs), which contain two LUTs and four FFs [Alt11]–[Alt16], [Int16], [Int17]. Nevertheless, a general description of those structures matches the logic block description, i.e., it is a structure that contains LUTs, FFs, and additional resources, and is connected to a predefined network of programmable signal pathways.

In modern FPGAs, LUT not only can be used to implement a logic function but also may be utilized as a memory block. Memory-capable LUT has limited hardware added for supporting on-the-fly LUT reprogramming. LUT-based random-access memory (LUTRAM), is often called distributed RAM, stemming from the fact that LUTs are evenly distributed across FPGA.

The configuration for FPGA blocks that reflects user design is kept in an external or internal memory, depending on FPGA architecture. The configuration is applied to the FPGA blocks during power-up, or during run-time partial reconfiguration [Sed06], [Cle16], [Ahm18]. The configuration for FPGA blocks is produced by algorithms that can implement the user digital design in FPGA architecture resources [Luu16]. The digital design, which is the input to the implementing algorithm, is prepared in textual format using the hardware description language (HDL). Verilog [Pal03] and Very High Speed Integrated Circuits Hardware Description Language (VHDL) [Nay97] are just two popular examples of HDLs. Important steps of the implementing algorithm include the logic synthesis by the decomposition of user-defined functions into atoms, which are applicable to FPGA resources (LUTs, logic blocks, DSP blocks, etc.) [Łub95], [Hry07], [Wyr13], placement of the atoms in resources of FPGA, and routing of the placed atoms using a predefined network of programmable signal pathways. The implementing algorithm is one of many Electronic Design Automation (EDA) tools that can process a design described using HDL. Other EDA tools are available for designers, e.g., a simulation tool that provides functional verification at an early stage of digital design development, and an optimization tool that can balance different aspects of the design, such as utilization of resources, maximum clock frequency, or energy consumption.

2.1. Restriction on considered FPGAs

In the dissertation, the basic common features of modern FPGAs are studied in the context of products offered by three of the leading FPGA vendors: Xilinx Inc., FPGA department of

Intel (formerly Altera), and Lattice Semiconductor Corp. Detailed descriptions of logic block architectures used by the vendors are presented in Appendices I – III.

The author limits his considerations to devices large enough to contain an SoC. The size of an FPGA is measured in logic cells (LCs) that are equivalent to a 4-input LUT (LUT4) paired with a flip-flop (FF). A memory controller, a common module of SoC, requires several thousands of LCs (e.g., Double Data Rate type 3 (DDR3) synchronous dynamic RAM (SDRAM) memory controller for Artix7 requires more than 6500 LCs). The whole SoC is expected to be substantially larger, therefore the author limits the considerations to a series of FPGAs with devices of more than 50,000 LCs.

2.2. Common features of considered FPGAs

In this section, common features of FPGA architectures are presented that should be considered in the development of NoCs for FPGA. The goal of this section is to demonstrate that key aspects of considered FPGAs are similar, which might render the possibility of a universal NoC architecture development.

The FPGA considerations are based on data sheets from the vendors [Alt11]–[Alt16], [Int16], [Int17], [Xil16a]–[Xil17], [Lat13]–[Lat16], and summed up in Table II.1. The above-mentioned three manufacturers offer products similar in various aspects, all based on LUTs and FFs, and with the capacity of up to millions of LCs. One can also see that memory controllers for high-capacity synchronous dynamic RAM (SDRAM) are supported as software IPs or hardware pre-engineered blocks. Each FPGA contains memory blocks (BRAMs with capacity from 9 kb to 45 Mb) distributed across its array. Each of the considered devices also includes distributed RAM.

TABLE II.1

COMPARISON OF FPGAS

Vendor	Device name	Year of announcement	Technology [nm]	Embedded ARM processor	Number of logic cells	FPGA basic cell feature	Supported external memory controller	Size of a block memory (BRAM)	The smallest LUT-based distributed memory (LUTRAM) configurations with separate read and write port	Portion [%] of LUTs usable as RAM
Intel (Altera)	Stratix 10	'13	14	yes	378k — 5.5M	6-input LUT + 2 FF [Alt11]	hard DDR4-2666	20kb and 45Mb	20×LUT as 20b×32-word deep or 10b×64-word deep RAM (32 bits per LUT)	25
	Arria 10	'13	20		160k — 1.15M			20kb		27 — 50
	Cyclone 10	'17	20	no	85k — 220k		hard DDR3-1866			21
	Stratix V	'10	28		236k — 952k		soft DDR3-1600			50
	Arria V	'11	28	yes	75k — 504k		hard DDR3-1066	20kb and 10kb		25 — 50
	Cyclone V	'11	28		25k — 301k		hard DDR3-800	10kb		24 — 35
Xilinx	Zynq UltraScale+	'15	16	yes	83k — 914k	6-input LUT + 2 FF	soft DDR4-2666	36kb or 2×18kb and 288kb	4×LUT as 3b×64-word deep or 6b×32-word deep RAM (48 bits per LUT)	30 — 50
	Virtex UltraScale+	'15		no	690k — 2.9M					
	Kintex UltraScale+	'15			205k — 915k					
	Virtex UltraScale	'14	20		783k — 5.5M		soft DDR4-2400	36kb or 2×18kb		14 — 22
	Kintex UltraScale	'14			318k — 1.5M					14 — 47
	Virtex7	'10	28		326k — 2M		soft DDR3-1866			28 — 43
	Kintex7	'10		yes (in Zynq)	66k — 478k					
	Artix7	'10			13k — 215k		soft DDR3-1066			
	Spartan7	'16		no	6k — 102k		soft DDR3-800			
	Spartan6	'09	45		4k — 147k		hard DDR3-800	18kb or 2×9kb		24 — 50
Lattice	ECP5	'14	40	no	12k — 84k	4 input LUT + 1 or 0 FF	soft DDR3-800	18kb	6×LUT as 4b×16-word deep RAM (10.7 bits per LUT)	50
	LatticeECP3	'09	65		17k — 149k					11 — 15
	LatticeECP2	'06	90		6k — 95k		soft DDR2-533			

In the following sections, the presented FPGA features are discussed from the point of view of NoC development.

2.2.1. Common features of distributed RAM

In the considered FPGAs, from 11% to 50% of LUTs can be used as RAM. The smallest available LUT-based memory (LUTRAM) configurations have the depth of 16 or 32 words, depending on the producer. Different numbers of LUTRAM bits are available per utilized LUT; on average, 32 bits per an LUT for Intel devices, 48 for Xilinx and 10.7 for devices from Lattice. As pointed out by a source related to Xilinx [Mai15], RAM-capable LUTs are well spread over an FPGA and their potential should be considered in NoC and SoC designs for Xilinx FPGAs. All the considered FPGAs include LUTRAM, so this conclusion should be extended to all of them.

2.2.2. Synchronous dynamic RAM support

Some of the considered FPGAs are equipped with hardware pre-engineered blocks of memory controllers of various memory types. For other FPGAs vendors provide memory controllers as software IP blocks. The most-supported memory type is synchronous dynamic random-access memory (SDRAM) in various versions (e.g., DDR, DDR2, DDR3, DDR3L, DDR4). The higher the version, the higher the maximum frequency of data transmission. The highest supported versions of SDRAM for each considered FPGA are listed in Table II.1. Nevertheless, not only SDRAM memory devices are supported. Examples of other supported types are reduced latency dynamic random-access memory (RLDRAM), and quad data rate static random-access memory (QDR SRAM).

External memory devices (SDRAM or other types) provide high-capacity storage for FPGA-based SoCs. 1Gb is a typical capacity of an SDRAM device, whereas the capacity of the internal memory of considered FPGAs (BRAM and LUTRAM) is one to three orders of magnitude lower.

2.2.3. Limitations on logic implementation in FPGA

FPGAs, unlike ASICs, have a predefined network of programmable signal pathways connecting LUTs [Mar16]. Each LUT in FPGA implements a logic function with a limited number of logic inputs. For Intel and Xilinx products, functions of up to 6 inputs can be

implemented in a single LUT. For Lattice products, the number of inputs is limited to 4. As the number of inputs for the required logic exceeds the number of single LUT inputs, it needs to be realized as multiple layers of LUTs connected with pathways. Additional layers of LUTs reduce the maximum clock frequency. In order to obtain the required design frequency, special care needs to be taken not to exceed the critical number of layers. It is not the case for ASIC designs, where the maximum clock frequency can be balanced with flexible lengths of pathways.

It can be concluded that the considered FPGAs differ from their ASICs counterparts. The key advantages of the considered FPGAs are highly available distributed RAM and support for high-capacity SDRAM, whereas the discussed frequency limitation is the main FPGA constraint. The identified advantages and constraints, which are common to all considered FPGAs, should be taken into account in the development of NoCs for FPGA.

Chapter 3. Requirements for network-on-chip and related previous works

The concept of NoC has been described in many books and articles, mostly in the context of ASICs [Ben06], [Che12], [Cot12], [Tat14], [Man14], [Dim15]. An NoC consists of switches[1] connected with links. Links physically interconnect switches, whereas the logic behind the communication is implemented in switches according to the adopted switching technique and designed communication protocol. Processing elements (PEs) are connected to NoC using network interfaces, which are the logic that adapts the PE communication protocol to the one used in NoC. Links and switches that are involved in the connection between two network interfaces create a path.

Switches can realize two switching techniques: connection-oriented (packet-switching) and connection-less (circuit-switching). In the connection-oriented approach, the whole path between the source and destination network interfaces is reserved before data is inserted into the network. On the other hand, in the case of the connection-less approach, consecutive segments of the path (buffers in consecutive switches and links between the switches) are reserved only for the time of data transmission on those segments (see also [LiuS13]). In the dissertation, only packet-switched networks are considered, which are characterized by high link utilization and are widely used for FPGA [Mai15], [Ret14]. In packet-switched NoC, a single message is divided into packets, and sent across the network. The packets are further divided into flow control units called flits. A flit is a portion of data usually transferred at one clock cycle between connected switches [Ber04].

NoC performance is characterized by some common metrics, i.e., throughput, latency, resource utilization. Throughput quantifies the rate at which data is transmitted through NoC. Latency has two components: first, the time it takes for a packet to pass through a number of synchronous elements (FFs and buffers in network elements), and second, the time the communication logic requires for taking switching decisions. Resources utilized for NoC may be of different kinds, e.g., silicon area in the case of ASIC, and the number of FFs and LUTs in the case of FPGAs. Electric power is another important resource for the applications in which

[1] The terms switch and router are often used as synonyms in context of NoC [Cot12].

the power source is constrained [Zyd11], as for mobile applications, or in which overheating may be a major concern.

A number of requirements for NoCs have already been identified [Pap15], [Mur07], [LiuS14], [Tat14]. The obvious requirements are:

a) Utilization of resources (power, silicon area, LUTs and FFs) should be minimized.

b) High data throughput should be offered.

c) Latency should be limited.

There are some other important requirements that are addressed in few references only:

d) Fairness of network access should be guaranteed, i.e., all network interfaces should experience throughput proportional to their relative request rates (throughput fairness) and the same latency (latency fairness) [Dal03], [LiuS14].

e) Network should be reliable, i.e., it should be deadlock-free, whereas the requirements of the minimum throughput and the maximum latency should be met [Mur07]. Another kind of reliability is fault tolerance required by some applications [Ben06], [Weh16], [Kan18], but fault tolerance is not considered in this dissertation.

3.1. Network reliability related aspects

Network reliability can be affected by congestions [Ber04]. Congestions lead to throughput and latency fluctuations, thus the average throughput is below the theoretical maximum value, and the average latency is increased, especially under high network load conditions.

According to [Hel15], [Pap15], [Tod14], [Zhu17], and [Abb14], the main aspects influencing throughput, latency and probability of congestions are the following: switching technique, topology (together with the routing algorithm), and the size of buffers. For each of those aspects, a number of techniques have been developed to meet the requirements for NoCs:

a) The packet switching technique. Three frequently used techniques are the store-and-forward, the wormhole and the virtual cut-through [Ber04]. Switching according to the store-and-forward technique requires collecting all flits of a packet before forwarding it to

the next switch or the network interface [Ker79]. This technique requires substantial buffer space and introduces extra packet delay at every switch. In the virtual cut-through switching scheme, packet forwarding can be started before the entire packet is buffered, therefore it reduces the delay, as compared to the store-and-forward scheme. Nevertheless, the virtual cut-through switching requires the same buffering space as the store-and-forward scheme, because the complete packet needs to be buffered in a switch until the next switch is ready to accept the packet. The advantage of the wormhole switching technique, as compared to the store-and-forward and virtual cut-through, is that it requires smaller buffers. In the virtual cut-through scheme and in the wormhole scheme, packet forwarding can be started before the entire packet is buffered but in contrast to the virtual cut-through, the wormhole switching scheme does not require buffering the whole packet in a switch if the next switch is not ready for the packet. The disadvantage of the wormhole scheme is that one packet may occupy several switches, block transmission through all of them, and lead to congestions. In general, the wormhole is congestion-sensitive and may result in low network utilization, but can provide low latency, and requires smaller buffers than the virtual cut-through. The latter requires larger buffers but provides low latency without limiting network utilization due to congestions.

b) The topology and the routing algorithm define paths in the network and influence the loads of individual links and switches. Uneven loads result in bottlenecks in the network that may cause congestions and unfair network access, leading to throughput and latency fluctuations [Dal03], [Ret14]. Topologies and routing algorithms that prevent congestions have already been investigated in the literature. In [Ret14], the congestions are limited by spreading traffic across NoC evenly by using adaptive routing. Another approach is to use multiple physical link or multiple physical network topologies [Mur07], [Yoo13]. Redundant physical links increase throughput, reduce bottlenecks and prevent congestions.

c) The size of buffers has been shown to have a major impact on throughput, latency and occurrence of congestions [Tod14]. The importance of this aspect is discussed in Section 3.2 together with techniques for adjusting the size of buffers known from the literature.

3.2. Buffer size adjusting techniques

The size of buffers used in a network affects its performance and cost. Determining the optimal buffer size is not trivial in the case of *a priori* unknown traffic load generated by

processing elements (PEs) and unknown ability of PEs to accept packets from the network buffers. In the references, several mechanisms were proposed to determine the buffer size. The simplest method is to set the size of buffers to hold as many packets as can be generated. In the case of *a priori* unknown traffic load, this worst-case approach results in unnecessarily large buffers [Son03].

More advanced methods of determining the buffer size exploit the statistics of the traffic load. Those statistics need to be explicitly provided [Abd16b], or a dedicated traffic load monitoring technique needs to be used [Tod14], [Zhu17], [Kam18], [AlF12].

In [Tod14], the buffer size is adjusted iteratively in consecutive SoC implementations. The monitoring module collects traffic statistics that are used to adjust the buffer size accordingly, and the estimated size is used in the next implementation. Multiple SoC implementations are time-consuming, therefore, in [Zhu17] simplified PEs are emulated and traffic statistics are collected faster. In [Kam18] an NoC simulator is proposed to estimate traffic statistics prior to NoC implementation. The above-mentioned mechanisms [Tod14], [Zhu17], [Kam18] need training data and are sensitive to any change in the traffic pattern.

In [AlF12], the total size of memory in a switch is constant, but based on the measured traffic load at each switch output, the memory is assigned between output buffers adaptively, during runtime. Nevertheless, even in [AlF12], it is pointed out that this mechanism is not dedicated for FPGA due to its complexity.

Commonly, networks distinguish the types of transmitted data and assign separate buffers to these different types. This technique is called virtual channels (VCs) [Yoo13], [Pos13], [Pap12], [Abb14]. In NoCs using VCs, it is common to guarantee throughput and latency just for critical types of data, like control messages. Therefore, the aforementioned buffer size determining techniques are only applied to buffers used by the critical data types. Buffers used by non-critical data can be optimized to reduce the memory cost.

In the NoC proposed in this dissertation, the buffer size does not depend on the traffic load. It is the opposite, and the traffic load is controlled to utilize the fixed-size network buffers without causing congestions. Details are provided in Chapter 5.

Chapter 4. State-of-the-art interconnections for FPGA

A number of NoCs, crossbars and bus architectures are proposed for ASICs in the references, and some of them are adopted for FPGAs. For the sake of brevity, the author focuses only on FPGA-oriented crossbars and NoCs.

4.1. AXI4 Interconnect

The Advanced eXtensible Interface 4 (AXI4) [Xil15] is a version of the Advanced Microcontroller Bus Architecture (AMBA) interface and protocol recommended by ARM Ltd for FPGAs. Originally, AXI4 was used to communicate with ARM cores embedded in many devices (see Table II.1). AXI4 modules, especially a crossbar called AXI4 Interconnect, are available as Intellectual Property (IP) cores in the Xilinx Vivado Design Suite and Intel Quartus. Many other IP cores with the AXI4 interface are available. Therefore, an FPGA-based SoC using AXI4 Interconnect and AXI4-compatible processing elements (PEs) can be developed rapidly. For this reason, AXI4 is widely used in FPGAs [Mai15], [Pap15].

AXI4 is a memory-mapped interface. The interface uses separate write and read channels and supports simultaneous, bidirectional transfers. Processing elements (PEs) with AXI4 interface are connected using AXI4 Interconnect, which is available as a standalone IP in the Xilinx IP Catalog. Many aspects of the AXI4 interface and AXI4 Interconnect IP can be adjusted to the requirements of a digital design. The interface supports packets of different sizes, up to 256 flits. The width of a flit can be configured in the range of 32 to 1024 bits, and additional bits of byte-enable (BEN) are transmitted, indicating which data bytes from a flit are valid. AXI4 Interconnect allows using address width up to 64-bits. The number of ports of the AXI4 Interconnect crossbar is configurable. Each port of AXI4 Interconnect can use LUTRAM or BRAM-based buffers. Optional pipeline flip-flops can be inserted at any interface to break a critical timing path at the cost of increased latency.

Despite its popularity, AXI4 Interconnect has drawbacks, e.g., in [Mai15] long connections were pointed out as the main drawback, and the application of NoC instead of AXI4 Interconnect was suggested for Xilinx FPGAs.

4.2. NoCs designed for FPGA

Two types of NoCs for FPGA have already been proposed in the literature: soft, with infrastructure implemented using general FPGA resources, and hard, using hardware switches embedded into the FPGA fabric as an additional resource [Abd14], [Abd16a], [Abd16c], [LiuT16], [Hud16], [Abd17]. Hard NoCs may limit the flexibility of FPGA and have not been implemented in chips that are available, and therefore only soft NoCs will be considered in this paper.

A few soft NoC architectures were proposed recently for FPGAs: Hoplite [Kap15], [Was17], [Vip17], [Kap17a], [Kap17b], [Kap17c], [Sid18], RAR-NoC [Ret14], CONNECT [Pap12], [Pap15], [Ahm18], [She14], [Mai15], LinkBlaze [Mai17], and OpenNoC [Red19].

The above-mentioned works use different approaches to the requirements for NoC (cf. Chapter 3) which is discussed in Sections 4.2.1 – 4.2.5.

4.2.1. Cost reduction

All the works consider the need to limit the usage of FPGA resources. Most of them dismiss adaptive routing techniques, and exploit static routing, willing for a smaller control logic [She14], [Pap15], [Kap15], [Mai15], [Red19]. Only in RAR-NoC [Ret14] is the usage of adaptive routing considered instead of static, and the increased throughput is reported at the cost of the switch size increase by 11%.

Buffers can consume a significant part of the network resources, therefore the authors of Hoplite [Kap15] and OpenNoC [Red19] propose an extreme approach and implement a toroid NoCs without buffers. Recently, Hoplite-based NoCs were proposed with the aim of lowering Hoplite average latency by improved routing [Was17], [Vip17], changing the topology from toroid to butterfly fat tree [Kap17c], or adding packet priorities [Sid18]. The small size of OpenNoC and Hoplite NoC is paid for by no guaranties for packet latency and lack of reliability. Using no buffer may not be justified for FPGAs that provide easily available distributed RAM (LUTRAM).

4.2.2. Distributed RAM utilization

In [Ret14] and [She14], it is proposed that the buffer size be limited by implementing wormhole switching that requires smaller buffers than virtual cut-through. Xilinx-related authors pointed out [Mai15] that LUTRAMs in Xilinx FPGAs have some generally defined minimum depths, so most of the LUTRAM capacity may be wasted by wormhole switching with shallow buffers. Virtual cut-through is proved [Mai15] to be an appropriate switching technique for a system with moderate size packets, i.e., shorter than the depth of LUTRAM-based buffers.

In the paper describing the CONNECT switch [Pap12], the LUTRAM potential is underlined, and the buffers are implemented strictly using LUTRAM. The configurability of the depth and width of the CONNECT switch is an advantage of the implementation, but it is also pointed out that even fixed but appropriate buffer depth can improve the predictability of resource utilization.

Based on [Mai15] and [Pap12], one can conclude that using buffers with depth equal to the depth of LUTRAM, using packets of length shorter than the buffer depth and employing virtual cut-through switching can result in resource-efficient NoC. Therefore, the author implements those ideas in RingNet NoC.

4.2.3. Inter-FPGA compatibility

The maximum operating frequency and the average throughput and latency are reported for all the above-mentioned networks in their source papers. The respective reports are provided for individual FPGA device types [Kap15], [Ret14], [Mai15], [She14], or for entire FPGA device lines [Pap15]. The lack of reports for FPGAs from different vendors may be an obstacle in determining the usability of NoCs.

Another NoC designed especially for FPGA is LinkBlaze [Mai17] that is dedicated for UltraScale+ devices from Xilinx. In particular, the switching logic was optimized for 6-input LUTs, and the switches were placed manually in the array with the aim of connecting them with global pathways. This device-aware network design results in a high-frequency NoC with its throughput higher than obtained for Hoplite and CONNECT when implemented on

UltraScale+ FPGA. LinkBlaze is an example of NoC optimized for one line of FPGAs, whereas the author of this dissertation looks for more universal NoC.

4.2.4. NoC reliability

A few of the presented NoC proposals focus on the requirement of NoC reliability. RAR-NoC [Ret14] uses a traffic monitor to control the routing algorithm implemented with the aim of reducing congestions. In [Pap15] and [Mai15], switches that support VCs are implemented. Still, even though fairness is an essential reliability parameter, it is out of the scope of most propositions. Only in [Pap15] did the authors point out the importance of fairness, but gave no numerical results for their NoC. In the dissertation, the author will provide a fairness analysis for RingNet, together with resource utilization and maximum operating frequency for various FPGAs to prove its usability.

4.2.5. SDRAM support exploitation

Out of the above-mentioned NoCs, only the authors of LinkBlaze [Mai17] consider FPGAs' support for synchronous dynamic random-access memory (SDRAM). SDRAM is a crucial component of many SoC projects [Abd16a] and utilizes a substantial part of NoC throughput. In [Ber04] an example of an SoC for an Advanced Video Coding (AVC) decoder is presented. In the system, the total communication traffic to and from SDRAM is much higher than the one required for communication between other processing elements (PEs). This type of memory-oriented SoCs are the target application for RingNet described in Chapter 5.

Chapter 5. RingNet architecture

Most NoC proposals focus on switch architecture only. In this dissertation, the author proposes a complete NoC architecture and a protocol for a memory-oriented SoC which is call RingNet. As stated earlier, the memory-oriented SoC is a specific type, where most of a traffic starts or ends in memory and minor traffic is send between processing elements. For such SoCs, RingNet excels in performance. In the development of RingNet, the conclusions from Chapters 2 – 4 are taken into account. The three basic ideas of the proposal are presented in Section 5.1. The secondary ideas are presented in Section 5.2.

5.1. Primary ideas of the proposal

5.1.1. Indirect communication

As described in Chapter 3, determining the buffer size is often complex due to the *a priori* unknown traffic load. In RingNet, the traffic load is controlled by a destination processing element (PE), which guarantees that packets injected into the network can be accepted by the destination PE without congestions. This way, fixed-size network buffers can be utilized.

With the aim of providing the traffic control mechanism, the author disallows direct communication between PEs. In RingNet, all traffic goes through one of the system buffers: System Memory (e.g., external SDRAM with a memory controller implemented in FPGA) or the Reflector. The System Memory is used as a data buffer, whereas the Reflector is a dedicated network buffer for control messages. The Reflector is introduced because it serves functions that are not supported by an ordinary memory controller. Among its functions, the Reflector informs a PE about data waiting to be read from the System Memory. For two PEs to communicate, the first PE writes data to the System Memory and writes information about the waiting data to the Reflector. Next, the Reflector informs the second PE about the data waiting to be read. Finally, the second PE reads the data from the System Memory to complete the communication. What is important, in RingNet, a PE requests data from system buffers when it is ready to accept it. Still, sending data to the system buffers is not a matter of any restriction and a sending PE can work at its own pace. The Reflector and the memory controller are the only devices in RingNet that need to be prepared for traffic with *a priori* unknown pattern. Details are given in Section 6.3.4.

RingNet has two distinctive properties among other NoCs proposed for FPGA. These are traffic load controlled by a destination processing element, and communication through system buffers.

5.1.2. Virtual cut-through switching

In Section 3.1, packet switching techniques are briefly presented. Next, in Section 4.2, it is summarized which of the techniques are used in known NoCs designed for FPGAs. For their property of low latency, the wormhole and the virtual cut-through switching are the two techniques adopted in FPGAs. The virtual cut-through requires substantial buffer space where the complete packet can wait until the next switch is ready to accept the packet. This buffer is not required in wormhole scheme; therefore, wormhole is chosen in some NoCs, which focus on limiting the FPGA resources usage [Ret14], [She14]. On the other hand, the wormhole scheme is congestion-sensitive and may result in low network utilization (cf. Section 3.1), which is not the case for the virtual cut-through technique. Therefore, the virtual cut-through technique should be used in order to obtain better NoC performance if the cost of required buffers is acceptable. In Section 2.2.1 it is discussed that distributed RAM (LUTRAM) is easily available in FPGA architectures. Therefore, the author proposes the virtual cut-through switching for RingNet aiming at efficient utilization of LUTRAM and for its congestion-insensitivity.

From Table II.1, one can see that 32-word deep is the shallowest LUTRAM that is supported by all the considered FPGAs, hence using the 32-flit deep buffers can result in resource-efficient NoC. For the virtual cut-through scheme, the packets should be small enough to fit into the buffer, therefore, packets in RingNet should be shorter than 32 flits.

5.1.3. Constraint of a switch size

For FPGA-based designs, the usage of logic functions with a low number of inputs can give a high clock frequency implementation (cf. Section 2.2.3). In [Pap15], the maximum clock frequency for a NoC implemented in FPGA is presented as a function of the number of ports that the network switches have. Reported frequencies are in the range 236 MHz – 241 MHz for 3-port switches, drops to 203 MHz for switches with 4 ports, degrades further to the 169 MHz – 181 MHz range if 5 ports are used, and finally drops to 121 MHz for 9-port switch. It shows that a low number of ports results in a NoC with high maximum clock frequency. In fact, a

3-port switch is the smallest switch usable in a network (see also [Mai17], and [Abd16a]). In order to maximize the clock frequency of the network, the author uses 3-port switches, called the Leaf Interface (LI) and Root Interface (RI).

5.2. Secondary ideas of the proposal

5.2.1. RingNet topology

A tree-of-rings topology is used (see Fig. V.1) with the System Memory and the Reflector connected to the ring at the root of the network tree, whereas PEs are connected to the rings at higher levels of the tree.

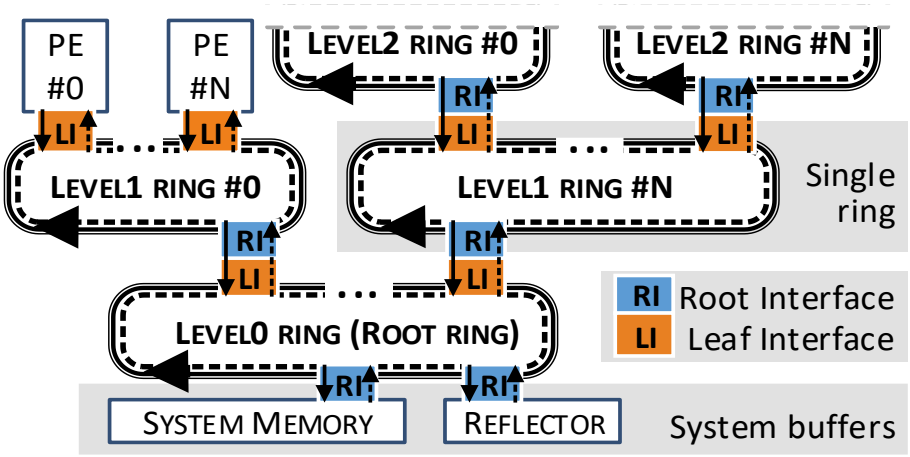


Fig. V.1. RingNet topology.

The ring topology is recommended for FPGAs by several authors [Abd16a], [Pap12], and it is one of just few topologies that can be constructed using 3-port switches. Moreover, the ring topology allows the network to spread over the whole FPGA area.

On the other hand, in NoCs with ring topology, latency increases proportionally to the number of PEs, and for a high number of PEs, this latency may be unacceptably high. Such latency can be reduced with the use of a mixed topology of smaller rings connected to a tree.

Both the tree and ring topologies are easy to scale in FPGAs without reducing the maximum clock frequency [Pap12]. For the tree-of-rings topology there is one path between the root and a leaf device, so static routing may be efficiently used, which simplifies the RingNet logic.

5.2.2. RingNet throughput control and multiple physical technique

The choice of topology affects the parameters of RingNet, especially it limits network maximum throughput. The maximum throughput of a ring is a function of the flit width and the clock frequency. The width of a flit in RingNet is constant, so the maximum ring throughput is determined by the maximum clock frequency for a given FPGA. To overcome this limitation, the multiple physical network technique is used. A single ring at any level of the tree can be replaced with multiple rings connected in parallel. The traffic is spread evenly between the parallel rings, and the maximum throughput is multiplied. The traffic in RingNet aggregates in a root ring and this level can also be multiplied to meet the throughput of the attached SDRAM memory. This approach to the problem of traffic aggregation in a root of a networks with tree topology is known from literature as fat-tree topology [Lei85], or variations of the topology, e.g., Fat H-tree [Mat09] or Z-fat tree [Add17].

The usage of the multiple physical network technique makes the throughput of RingNet controllable. In this situation, the System Memory throughput becomes an obvious limitation of the approach presented by the author. Nevertheless, the System Memory load can be reduced by connecting additional memory buffers at any ring. In Fig. V.2 an example of RingNet network is depicted with local buffer attached to the ring placed on the left. The local buffer (instead of the System Memory) can be used for the data exchange between PEs connected to a common ring. Nevertheless, PEs use the System Memory to exchange data with PEs connected to other rings of a network. Information about data waiting in the local buffer still needs to go through the Reflector as stated in Section 5.1.1. Each local buffer is connected to a ring through its own RI and can utilize the block RAMs available in all considered FPGAs.

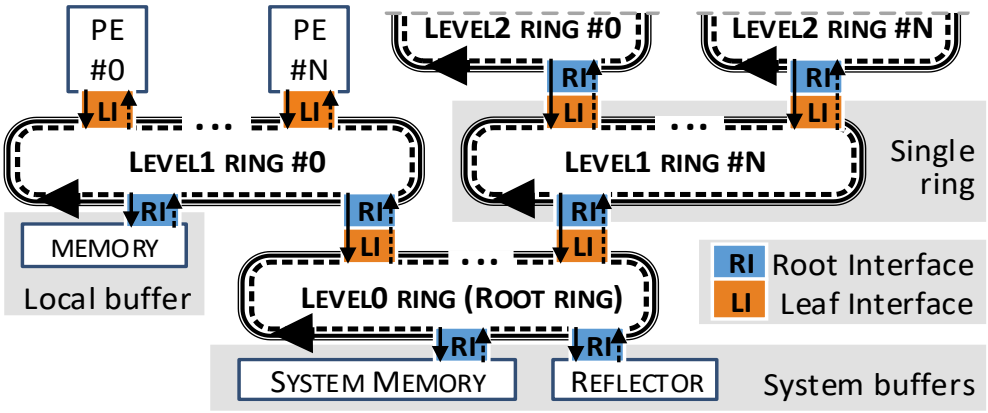


Fig. V.2. The RingNet network using the local buffer concept.

Reducing the System Memory load by using the local buffers requires a careful network configuration. In this scenario, the PEs that exchange data between each other need to be connected to a common ring with the local buffer attached.

5.2.3. Flit size

Flits in RingNet have 8 data bytes with additional 8 bits of byte-enable (BEN). Like in AXI4, BEN bits indicate which data bytes from a flit are valid. One of the goals of RingNet development is to provide maximum throughput equal to or higher than the throughput of SDRAMs supported by FPGAs. While using flits with 8 bytes of data, the maximum throughput of RingNet is compared with the throughput of the supported SDRAMs:

- The slowest SDRAM from Table II.1 is DDR2-533, which provides throughput of 4.3 Gbps for an 8-bit interface, whereas a single RingNet ring running at a moderate frequency of 75 MHz already exhibits the throughput of 7 Gbps.

- The most demanding DDR4-2666, supported in Xilinx UltraScale+ series and Intel 10 series, offers 192 Gbps for a 72-bit interface. For comparison, four parallel RingNet rings can transfer 196 Gbps when running at the clock frequency of 525 MHz that is easily achievable for the FPGAs.

5.2.4. Flow control

Considering the requirements for fair network access (Chapter 3), the author proposes a flow control mechanism for RingNet with the access controlled locally, at the level of each ring (cf. Chapter 6).

Chapter 6. Components and protocol of the RingNet network

This chapter provides an overview of the implementation of RingNet network components and protocol. The presented implementation is in compliance with the ideas presented in Chapter 5. The purpose of this implementation is to provide a functional design that can be processed using Electronic Design Automation (EDA) tools; especially simulation tools can be used to evaluate the performance of the RingNet architecture, whereas packing tools can be used to evaluate its resource utilization and the maximum clock frequency. For this reason, the author describes RingNet modules using Verilog hardware description language (HDL), which is one of the most popular HDLs, and is accepted by EDA tools. The results obtained using EDA tools are discussed in Chapters 7 – 9.

In Sections 6.1 – 6.3, details of the RingNet modules prepared using HDL are presented. The implementation presented in this chapter is just one realization of the ideas from Chapter 5. Other implementations are possible. Nevertheless, the author explored various configurations with different parameters in the search for optimal implementation. The author believes that the presented implementation at least does not limit the RingNet functionality presented in Chapter 5, and is therefore useful for evaluating the RingNet ideas.

6.1. Network physical channels

In this NoC, there are two physical channels. The first one transports packets form a processing element (PE) to a system buffer (System Memory or Reflector). The second one transports packets from a system buffer to a PE. PEs are connected at the leaves of the network tree, whereas the system buffers are connected at the root of the tree, therefore, the first channel is called Leaf-to-Root (L2R) and the second one is called Root-to-Leaf (R2L).

The main modules of RingNet are two types of network interfaces: Leaf Interface (LI) and Root Interface (RI). The LI is used to connect a PE to a network. The RI is used to connect a system buffer. As depicted in Fig. V.1, a combination of RI and LI is used to connect rings at different levels of a network tree. Both RI and LI are 3-port switches and they insert packets to a ring using a 2×1 multiplexer and accept packets from a ring using 1×2 demultiplexers (see Fig. VI.1).

Each ring in RingNet has one L2R channel and one R2L channel (see Fig. VI.1). A flow control mechanism for the L2R channel uses an additional L2R control channel (L2R CTRL).

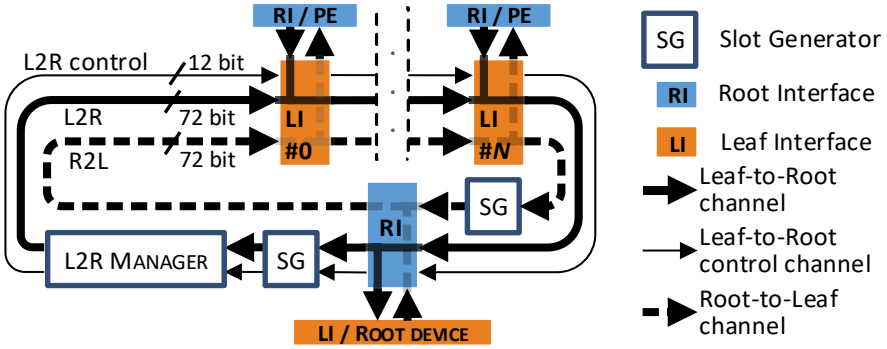


Fig. VI.1. A single RingNet ring.

6.2. Ring adapter

As already stated in Section 5.2.2, a single ring can be replaced with multiple rings connected in parallel in order to increase the overall network throughput, e.g., at certain critical levels of the tree. Usually, due to the traffic aggregation in the root of the RingNet tree, more multiplied rings will be used at levels closer to the network root. These multiplied rings in RingNet can be connected to other levels of the tree using a dedicated adapter. Fig. VI.2 depicts an adapter for the L2R channel. The adapter is placed between LIs and RIs of the rings when at least one ring is multiplied.

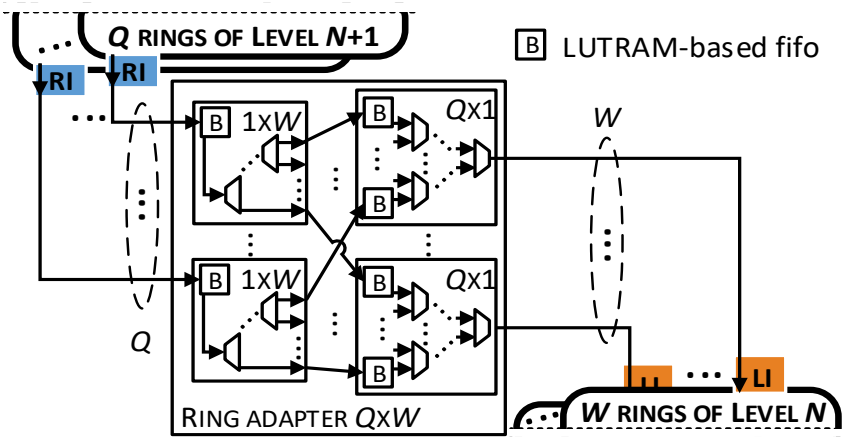


Fig. VI.2. Ring adapter for L2R channel.

An individual adapter is used for L2R and R2L channels. It is built of 2×1 multiplexers and 1×2 demultiplexers and LUTRAM-based 32-flit deep buffers.

6.3. RingNet protocol

For RingNet, a protocol is developed that provides communication between PEs, limits congestions, provides fair network access and supports packet priorities. The protocol covers 4 layers of the OSI model, from the data link layer to the session layer, described in Sections 6.3.1 – 6.3.4, respectively. Higher OSI layers are out of scope of the RingNet protocol.

6.3.1. Data link layer

The data link layer specifies the structure of flits and packets, and defines the ring access protocol.

As recommended in Section 5.1.2, packets should be shorter than the depth of the buffers. For the FPGAs from Table II.1, the most shallow LUTRAM is 32-word deep. Therefore, 32 and 64-flit deep buffers are used in RingNet. Packets with two possible lengths of 2 and 9 flits are chosen. The purpose for the two lengths is described in Section 6.3.3. The first flit of a packet is the header that encapsulates control information for each protocol layer (especially the routing information). The following 1 or 8 flits transport data. In RingNet flits, 64 bits are transmitted with additional 8 bits of byte-enable (BEN).

A constant pattern of time slots for header flits and data flits is passing through each Leaf Interface (LI) and Root Interface (RI) connected into a ring. The time slots constantly circulate around a ring. Slots are produced by the Slot Generator (SG). The access to the L2R channel slots is controlled by the L2R Manager.

6.3.1.1. Slot Generator

Fig. VI.3 depicts a pattern of time slots for header flits and data flits circulating through interfaces around a ring. Those flit slots are organized into long and short slots for long and short packets, respectively. Slots are produced by the Slot Generators (SGs) depicted in Fig.VI.1. The Leaf Interface (LI) and the Root Interface (RI) can populate these slots with packets.

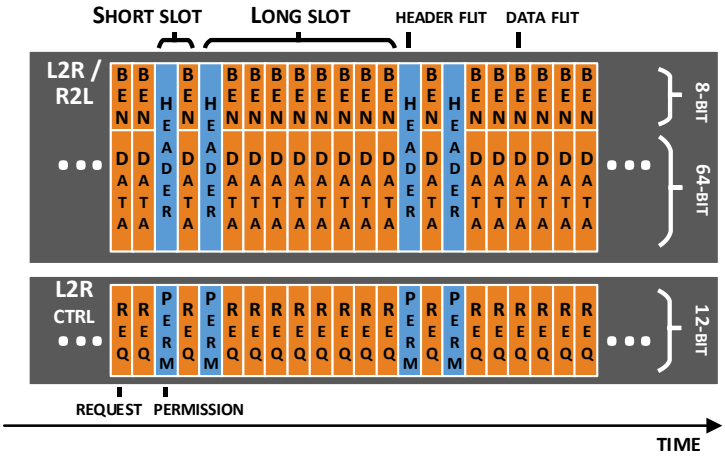


Fig. VI.3. Time slots at R2L, L2R and L2R control channels, passing through an interface.

The header flit encapsulates control information for each protocol layer. In Fig. VI.4 fields of the header flit are presented.

	OSI LAYER	FIELD NAME	NUMBER OF BITS	
HEADER	SESSION	Session ID (SID)	4	72
	TRANSPORT	Segment ID (SgID)	4	
		Memory Operation (MOP)	2	
	NETWORK	Mem Address	37	
		Level+4 Leaf Interface ID (LID_{L+4})	4	
		Level+3 Leaf Interface ID (LID_{L+3})	4	
		Level+2 Leaf Interface ID (LID_{L+2})	4	
		Level+1 Leaf Interface ID (LID_{L+1})	4	
		Current level Leaf Interface ID (LID_{CL})	4	
	DATA LINK	Packet Valid (V)	1	
		Packet Length (L)	1	
		Packet Priority (PP)	2	
		Packet Rejected (PR)	1	

Fig. VI.4. Packet Header format.

For the data layer, the header flit indicates the following information:

- if the slot is populated with a valid packet (Packet Valid field),
- if the packet is a short or a long one (Packet Length field),
- priority of the packet (Packet Priority field, see Section 6.3.1.2 for details),
- if the packet was rejected by the Root Interface (RI) due to insufficient buffer space available in the interface (Packet Rejected field, see Section 6.3.1.3 for details).

The network layer part of the header flit specifies the addresses used at the Leaf-to-Root channel (37 bits of Memory Address field) and the Root-to-Leaf channel (five fields for Leaf Interface ID, one for each level of a RingNet tree). More details on addressing used in the RingNet network are given in Section 6.3.2.

The transport layer part of the header instructs a system buffer what should happen to the data transported in the data flits (Memory Operation field, more details are given in Section 6.3.3). The Segment ID field indicates the order in which packets were generated.

The session layer part of the header flit marks a single stream of data divided between many packets (the Session ID field).

Fig. VI.5 depicts a block diagram of the Slot Generator module.

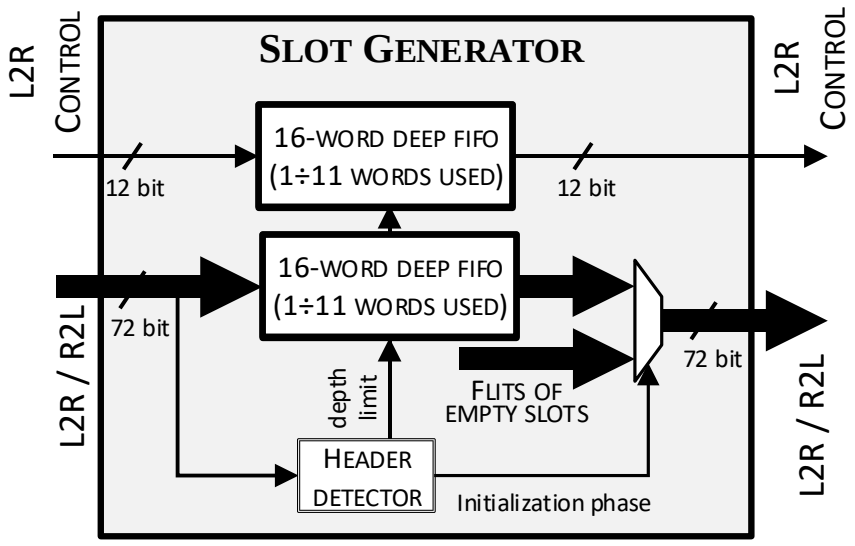


Fig. VI.5. Slot Generator.

Right after a ring is powered, no slots for packets are present in the ring. During the initialization phase, the Slot Generator inserts empty slots according to the pattern depicted in Fig. VI.3. The Slot Generator ends the initialization phase when it detects that the header of the first generated slot appears at its input. At the end of the initialization phase, the output of the module is switched to the output of the internal fifo. The generated slots start to circulate around the ring, also through the internal fifos of the Slot Generator.

The length of the ring is expressed in the number of registers that a flit needs to pass when it circulates around the ring. It is required for a RingNet ring to have a length of so many registers that an even number of packets fits into these registers. This restriction guarantees a constant flow of short and long slots. Therefore, the length of a ring is an integer multiple of the length of a long and a short slot, i.e., an integer multiple of 11 registers (9 flits are used for the long packet and 2 flits are used for the short one). The internal fifo in a Slot Generator is used to adjust the length of a ring to the required value. The Slot Generator uses 1 – 11 words of the internal fifo. The process of adjusting the length of a ring is automatic and eases the configuration of the ring and network. The internal fifo is meant to utilize LUTRAM, therefore its depth equals 16 flits, which is the most shallow depth supported in LUTRAM (cf. Table II.1).

6.3.1.2. L2R Manager

The access to the Leaf-to-Root (L2R) channel is controlled by the L2R Manager. In order to access packet slots in the L2R channel, LI sends a request to the L2R Manager and obtains permission. The requests and permissions are sent using the L2R control channel. Permission is sent synchronously to the header flit of a granted packet slot. Requests are sent in the remaining time slots (cf. Fig. VI.3). In Fig. VI.6 fields of the permission and request flits are presented.

	FIELD NAME	NUMBER OF BITS	
PERM	Permission Valid (PV)	1	12
	Packet Length (L)	1	
	Packet Priority (PP)	2	
	Leaf interface ID (LID)	4	
	Request ID (RID)	4	
REQ	Request Valid (PV)	1	12
	Packet Length (L)	1	
	Packet Priority (PP)	2	
	Leaf interface ID (LID)	4	
	Request ID (RID)	4	

Fig. VI.6. Permission and request format.

The permission and request flits have similar formats. One bit is used to indicate the permission or request validity (Permission Valid and Request Valid fields). One bit indicates if

the granted and requested slot is a long or a short one (Packet Length field). RingNet supports 4 packet priorities, ordered from the highest priority 3 to the lowest priority 0. Each packet and the associated request and permission has an assigned priority (Packet Priority field). The Leaf Interface ID field identifies the source of the request or the destination of a permission in the ring. The value of the Request ID field is copied from the request flit by the L2R Manager to the corresponding permission. It helps the Leaf Interface to identify which buffered packet the permission relates to, especially when it sends many overlapping requests for many slots.

A block diagram of the L2R Manager is depicted in Fig. VI.7. The L2R Manager keeps the requests in LUTRAM-based buffers. Individual buffers are used for requests of different priorities. If the L2R Manager detects a free packet slot, it generates a permission according to the buffered requests. The permissions are granted based on the priority of buffered requests and their order of arrival. Considering the order of arrival in arbitration logic is called age-based arbitration and can tighten the distribution of packet latency [Dal03].

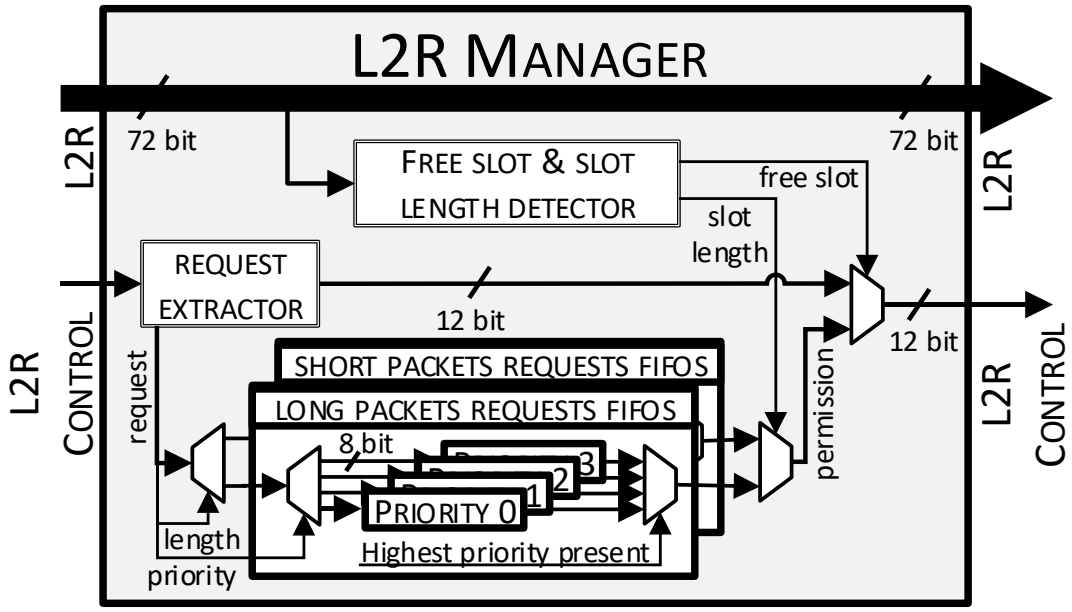


Fig. VI.7. L2R Manager.

The packets from the L2R channel leave a ring through Root Interface (RI). According to the virtual cut-through switching, the packets from the L2R channel leave a ring through RI only if the RI has buffer space available for a whole packet. Otherwise, the packet is rejected

by the RI and starts to cycle around the ring until enough space is available in the RI. A circulating packet is recognized by the L2R Manager, and no new permission is granted until all packets that circulate on L2R leave the ring. This way, all packets should finally leave the ring with limited differences in latency.

The presented request and permission strategy is sufficient to guarantee fair access to the L2R channel for all PEs in terms of the average latency and granted throughput (see simulation results in Chapter 7).

The flow control mechanism used for the L2R channel also guarantees fair access to the Root-to-Leaf (R2L) channel. Therefore, no flow control for the R2L channel is needed, and no additional resources are used. Details will be given in Section 6.3.3.

6.3.1.3. Network interfaces

Both the Root Interface (RI) and Leaf Interface (LI) are 3-port switches and they insert packets in a ring using a 2×1 multiplexer and take packets from a ring using 1×2 demultiplexers (see Fig. VI.8 and Fig. VI.9).

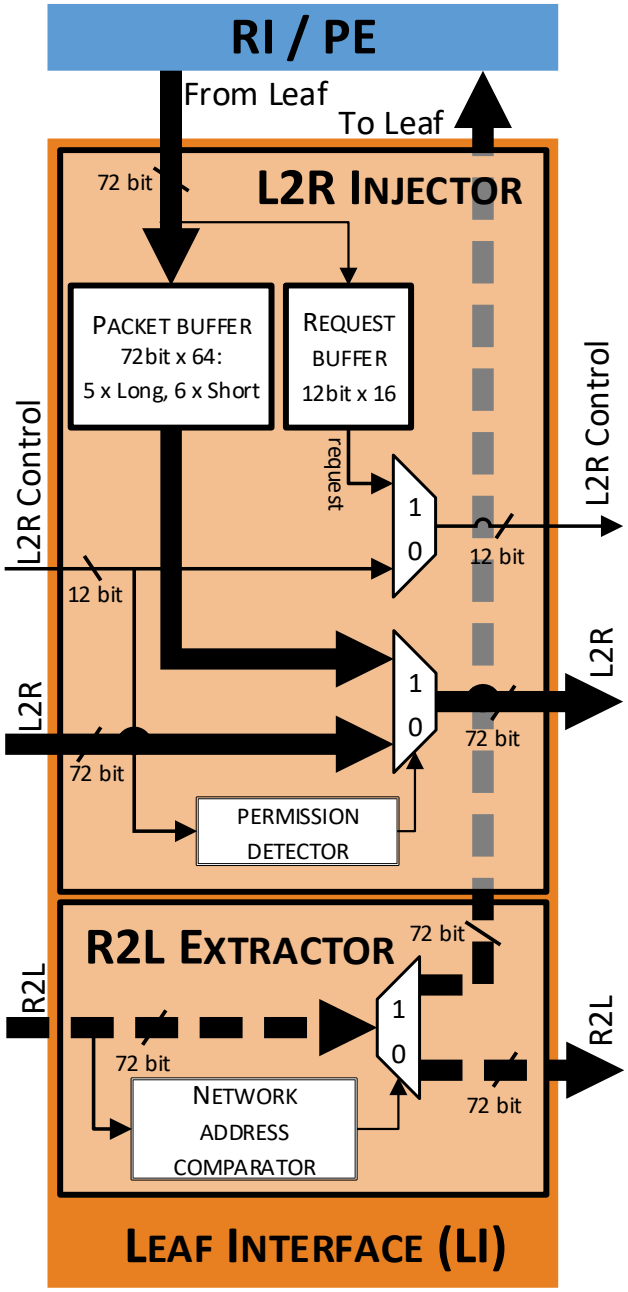


Fig. VI.8. Leaf Interface (LI).

The LI is divided into two parts: the Leaf-to-Root (L2R) Injector part and the Root-to-Leaf (R2L) Extractor part. The L2R Injector buffers packets, sends requests, and inserts buffered packets onto the L2R ring after acquiring permission. Sending a request and obtaining permission requires some additional time, so a constant flow of packets from one LI may be impossible. To overcome this problem, multiple packets can be buffered in LI and multiple requests can be sent without receiving permission. This overlap lets a single LI exploit the

maximum throughput of a ring. The packet buffer used in LIs and depicted in Fig. VI.9, utilizes LUTRAM with the depth of 64 words. It provides enough capacity for 6 short packets and 5 long packets. The request buffer utilizes 16-words deep LUTRAM. The requests are stored in the request buffer. Each of the requests corresponds to one packet stored in the packet buffer. When a permission reaches LI, it carries a Request ID number that identifies the corresponding request from the request buffer and the buffered packet.

The R2L Extractor part of LI extracts a packet from the R2L channel or bypasses the packet to the next LI on a ring. The respective decision is taken according to the addressing information encapsulated in the packet header (see Section 6.3.2). In contrast to the L2R Injector part of LI, the L2R Extractor part has no buffer for packets. The extracted packet will be accepted by the attached processing element (PE) or the attached Root Interface, depending on which one is connected to the LI. If a PE is connected to the LI, it should accept the packet as a result of the primary idea of RingNet expressed in Section 5.1.1 (a processing element controls the traffic load at an R2L channel, therefore, the packet has been requested by the PE and it should accept the requested packet). If, instead of a PE, an RI of higher network level is connected to the LI, this RI provides a small buffer used for synchronizing the extracted packet to slots on the higher ring level.

A block diagram of the RI is depicted in Fig. VI.9.

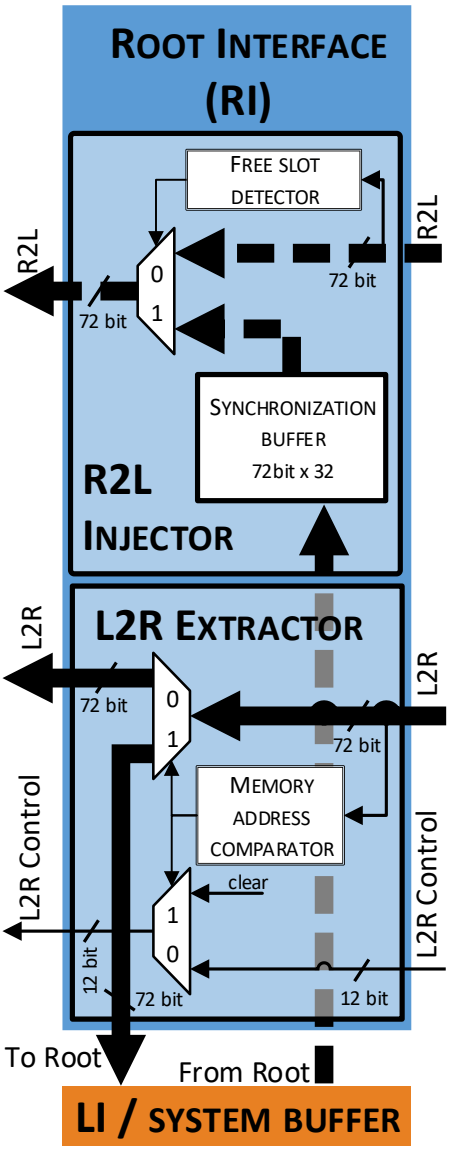


Fig. VI.9. Root Interface (RI).

The R2L Injector part of the RI accepts packets from the attached device (LI or system buffer) and stores them in the internal synchronization buffer. The synchronization buffer is used for synchronizing the incoming packet with slots at the R2L channel.

The L2R extractor part of the RI identifies the packet that should be extracted from the L2R channel, and checks if the attached LI or system buffer has buffer space available for the packet. If the buffer space is available, then the packet is extracted. Otherwise, the RI marks the packet as rejected (sets the Packet Rejected bit in the packet header) and bypasses it to the L2R channel. The packet marked with the Packet Rejected bit is recognized by the L2R Manager, and an

appropriate mechanism, described in Section 6.3.1.3, is initialized with the aim of extracting the rejected packet.

6.3.2. Network link layer

The network layer protocol defines RingNet addressing. Different addressing is used for the Leaf-to-Root (L2R) and the Root-to-Leaf (R2L) channels. For the L2R channel, 37-bit memory addressing is used, therefore up to 128 GB can be addressed in the network. Both the Reflector and System Memory have assigned memory address spaces recognized by Root Interfaces (RIs). The L2R extractor part of the RI reads the memory address from the header of a packet transported on the L2R channel. If this address is in the range assigned to the RI, then it is extracted and sent down the network tree, to the attached LI or to the attached system buffer.

For the R2L channel, the network address is used that contains five (each 4-bit wide) Leaf Interface ID numbers. Each number identifies the Leaf Interface (LI) that should accept the packet on its way up the network tree, at a certain RingNet tree level. When the value of the Leaf Interface ID carried in the header of the packet (see Fig. VI.4) matches the ID of the encountered LI, then the LI extracts the packet from the R2L ring and transfers it up the network tree to the attached processing element or attached Root Interface of the higher-level ring.

At each transition between rings of different levels, the network address carried in a packet header is updated. On the way down the network tree, via the L2R channel, the network address is extended by the ID of each LI through which the packet enters each ring. This way, at the root ring, the packet header comprises a full list of encountered LIs, i.e., a complete network address of a processing element. This address is used to route the corresponding response packet on its way up the network tree, via the R2L channel. At each network level the network address identifies the LI that should accept the packet. After the packet acceptance, the used part of the network address is cut off.

The network addressing used at the R2L channel limits the number of network tree levels to 5 and the number of network interfaces connected to a single ring to 15. The applied addressing scheme limits the number of PEs that can be connected to the RingNet network to 759,375.

6.3.3. Transport layer

The transport layer defines logical channels and describes the communication between PE and system buffers.

In RingNet, packets are transported to and from memory-mapped system buffers, therefore the RingNet protocol supports basic memory operations. The memory read or write operation is encoded in a packet header, thus creating a read or write packet. If a packet is sent from PE via the L2R channel to a system buffer, a response packet has to be sent back to the PE by the system buffer through the R2L channel. For the read operation, the response packet contains the data read from the memory, whereas for the write operation the response packet is a write operation acknowledgement.

6.3.3.1. Definition of logical channels

Logical channels are defined for RingNet. In each logical channel, a certain memory operation and packets of a certain length are used. Most transfers are realized by two logical channels:

a) Logical write channel. PE sends a long write packet through the L2R physical channel, with data to be stored to the memory. A short response packet is sent back by a system buffer through R2L as a write acknowledgement.

b) Logical read channel. PE sends a short read packet through the L2R physical channel, and a long packet with data read from the memory is sent back from a system buffer through the R2L physical channel.

c) Logical control channel. This channel is used by the session layer to send control data. In the logical control channel, short packets with the highest priority are sent between PEs and the Reflector, through the L2R physical channel and through the R2L physical channel.

What is very important is that for read and write logical channels, a single response packet is sent via R2L in response to the L2R packet. Therefore, the same number of packets are transferred through L2R and R2L channels. This way, fair access to the L2R channel guarantees a fair access to the R2L channel as well, despite the fact that for R2L no requests-permissions

mechanism is used explicitly. The proposed flow control prevents congestions in both physical channels, still utilizing resources in the L2R channel only.

6.3.3.2. Maximum throughput of the logical write and read channels

Theoretical maximum throughput T_{RW_MAX} of the logical read and write channels, expressed in bits per clock tick (bpt) can be calculated according to the given formula:

$$T_{RW\_MAX} = R \cdot 64 \cdot \frac{8}{11} \text{ [bpt]}, \quad (1)$$

where 64 is the number of data bits in a flit, the 8/11 factor is the share of flits carrying write data on the L2R physical channel, and the share of flits carrying read data at the R2L physical channel, and R is the number of parallel rings used at the root ring.

6.3.4. Session layer

The transport layer describes the communication between PEs and system buffers, but not between individual PEs. The communication between PEs is finally possible at the session layer. An example of communication between two PEs using the System Memory and the Reflector is depicted in Fig. VI.10. The example illustrates two transactions of sending data from PE1 to PE2.

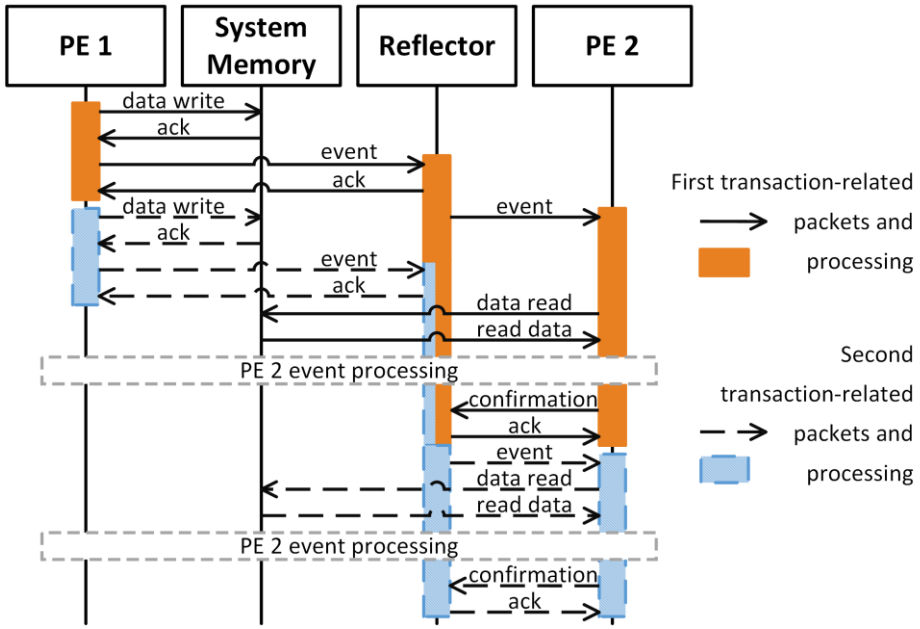


Fig. VI.10. Sequence diagram for two transactions between PEs.

PE1 starts the transaction by sending data to the System Memory (through logical write channel). Then, PE1 informs PE2 that data in the System Memory is ready to be processed by sending an *event* message to the Reflector (through the logical control channel). An event is a short packet with an address in a range reserved for the Reflector. The source PE (PE1 in the example) can start many transactions without considering the state of the destination PE. In the example, PE1 starts second transaction with PE2 without waiting for the first transaction to finish. Events are buffered in the Reflector and sent to the destination PE one at a time. After receiving the first event, PE2 reads the corresponding data from System Memory (through logical read channel) and starts processing of the data. During the processing, the Reflector buffers new event addressed to PE2. After processing the data related to the first event, PE2 sends an event confirmation to the Reflector which ends the first transaction. Receiving the event confirmation from PE2 indicates it is ready for a new processing, therefore the Reflector sends the second buffered event. As a result, PE2 starts processing the second portion of data. Finally, the second transaction ends after the second event confirmation reaches the Reflector.

The example demonstrates two advantages of the proposed indirect PE communication. First, congestions arising due to PE overload are prevented, because the destination PE (PE2 in the example) receives events only when it is in an idle state, whereas during the processing state

the destination PE controls its load. Second, the source PE (PE1 in the example) does not need to monitor the state of the destination PE before starting a new transaction.

The Reflector queues events for PEs, thanks to which a PE experiences a controlled load. Nevertheless, the Reflector itself needs to be prepared for the traffic of events with *a priori* unknown load. As already discussed in Section 3.1.1, setting the size of buffers when the traffic load is unknown is not trivial. Details about a possible implementation of the Reflector are provided in Appendix IV. Briefly, in the proposed implementation, the Reflector stores events in a general buffer shared between all the connected PEs. The usage of a shared buffer can be beneficial in a case where individual PEs experience uneven or time-varying load, which is a probable scenario in an SoC. The first benefit of the shared buffer approach is that the space of the shared buffer can be smaller than the aggregated space of individual buffers. The peak in time-varying load experienced by an individual PE determines its buffer space requirement. Due to a possible time shift of load peaks experienced by PEs, the required space of a shared buffer may be lower than the aggregated space of individual buffers. The second benefit is speeding up a development of an SoC as a result of the fact that adjusting the size of the shared buffer should be easier than adjusting the sizes of many individual PE buffers. For the given reasons, the proposed shared buffer approach, detailed in Appendix IV, can result in effective memory utilization. Relatively big buffers required for the purpose of event queuing in the Reflector can exploit block RAM available in all considered FPGAs. Preferably, there is one Reflector in a RingNet network, therefore the resources utilized for its buffers have a limited impact on the overall cost of the network.

The Reflector provides additional system functions, like informing a dedicated PE about PEs connected to a network, registering new PEs, informing about events buffered in the Reflector, alarming about the fullness of event buffers, resetting a PE, etc.

The presented session layer is the highest mandatory layer of protocol defined for RingNet. The protocol specified for four layers (data link layer to session layer) is sufficient to provide communication between PEs, featured with flow control for fair access and a priorities mechanism.

6.4. Summary

In this chapter, the author presents the implementation of ideas from Chapter 5. Other implementations are also possible, especially different values of many parameters can be chosen. The flit size is chosen to be 64 bits in order to provide the maximum throughput equal to or higher than the throughput of SDRAMs supported by FPGAs (see Section 5.2.3). Nevertheless, some applications may have lower throughput requirements and a lot of resources can be preserved when a narrower flit is used. On the other hand, using a narrower flit makes the header flit also narrower. A modified RingNet architecture with, e.g., a 32-bit wide flit is possible but the memory and network addresses needs to be cut. Although the implementation with a narrower flit is possible, the 64-bit wide flit and the support for high throughput applications, which utilize the maximum throughput of SDRAM devices is the essence of RingNet.

The sizes of packets utilized in the proposed implementation are other parameters that can be chosen differently. The limit on the packet size is discussed in Section 5.1.2 and the 32 flits are concluded to be the maximum size of a packet that provides efficient utilization of LUTRAM for FPGA NoC. In the implementation, packets of two sizes are used, i.e., 2 and 9-flits long with 1 and 8 data flits, accordingly. The use of just 2 packet sizes, compared to variable-length packets, generates a simpler and faster logic. The purpose for two packet lengths is discussed in Section 6.3.3. The long packets are used for data transmission and the short packets are used for control. The long and short packets are also used for time separation of read and write logical channels (see Section 6.3.3.1). Nevertheless, other packet lengths, shorter than 32 flits, can be chosen without loss to RingNet functionality. On the other hand, the author explored various configurations with different parameters in search of the optimal implementation, therefore any change in packet lengths may degrade the RingNet performance. Moreover, the objective of the implementation is to provide a functional design that can be simulated and synthesized in order to evaluate ideas presented in Chapter 4. The optimization of RingNet implementation is not essential, nevertheless, as discussed in Chapters 7 – 9, the proposed RingNet implementation is characterized as a high-performance NoC and high-performance FPGA-oriented digital design.

Chapter 7. Simulation results for RingNet

The proposed RingNet is simulated with the aim of assessing its throughput and latency. Moreover, the following reliability aspects are tested: inter-channel dependencies, network access fairness, and the priority mechanism.

7.1. Methodology

In the references, e.g., in [Dal03], [Ben06], [Che12], [Pap15], [Was17] it is shown that the performance of an NoC depends on the applied traffic pattern, i.e., the way that the destinations of simulated packets are chosen. This distribution is described by traffic matrix *M*, where each matrix element $M_{s,d}$ gives a fraction of traffic sent from PE *s* to PE *d*. Different traffic patterns are often simulated, e.g., a random pattern, where all entries of *M* are equal, or a permutation pattern, where entries of *M* are described by a non-constant function of *s* and *d* (more on that in [Dal03]). On the other hand, the fundamental feature of RingNet is that the System Memory is the destination of all data, therefore, the only traffic pattern simulated in this chapter is the one in which processing elements (PEs) generate packets addressed to the System Memory. This scenario describes a special case of the permutation traffic pattern. The pattern used for RingNet simulations is described by matrix *R* (2).

$$R = \begin{bmatrix} 0 & 1/(2N) & \cdots & 1/(2N) & 1/(2N) \\ 1/(2N) & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 1/(2N) & 0 & & 0 & 0 \\ 1/(2N) & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (2)$$

where entries in the first column and the first row give a fraction of the traffic sent to and from the System Memory, respectively. *N* is the number of interconnected PEs, and the size of *R* equals $(N+1) \times (N+1)$. Equal distribution of load to and from the System Memory is a consequence of the RingNet protocol, i.e., in response to a packet sent from a PE to the System Memory (the Leaf-to-Root (L2R) channel packet), a single response packet is sent from the System Memory to the PE (packet sent via Root-to-Leaf (R2L) channel, see Section 6.3.3).

PEs are simulated with the use of packet generators (PGs). Each PG independently generates packets for the logical read and write channels with the configurable average delay D_{aver}between two consecutive packets:

$$D_{aver} = \frac{N \cdot 64 \cdot 8}{T_{RW\_MAX} \cdot L \cdot 11}, \quad (3)$$

where T_{RW_MAX} is the theoretical throughput of each logical channel (see Eq. (1) in Section 6.3.3), L is the requested aggregated load generated by all PGs expressed as a percentage of the throughput T_{RW_MAX}. Eq. (3) is explained because 64 is the number of data bits in a flit, the 8/11 factor is the share of flits carrying write data on the L2R physical channel and the share of flits carrying read data at the R2L physical channel. The actual delay D is defined as the time interval that a PG waits before it sends another packet. D is an output of a random number generator with discrete uniform distribution $\mathcal{U}\{0.8D_{aver}, 1.2D_{aver}\}$ used for making the traffic more realistic.

The simulations use a model of the System Memory that supports unlimited throughput and introduces negligible latency, thanks to which the System Memory does not affect network performance results. Therefore, the presented latencies are the minimum latencies for the RingNet architecture. With the aim of estimating the latency in the actual SoC, the presented latencies should be increased by the latency featuring the memory device used as the System Memory.

In the experiments, RingNet with two levels of rings is simulated as depicted in Fig. 6. The size of the network is controlled using the following parameters:

- *R*: Multiplication degree of the root level, i.e., the number of parallel rings used at the root level (level 0).

- *F*: The number of 1st level rings.

- *G*: The number of PGs connected to a single 1st level ring.

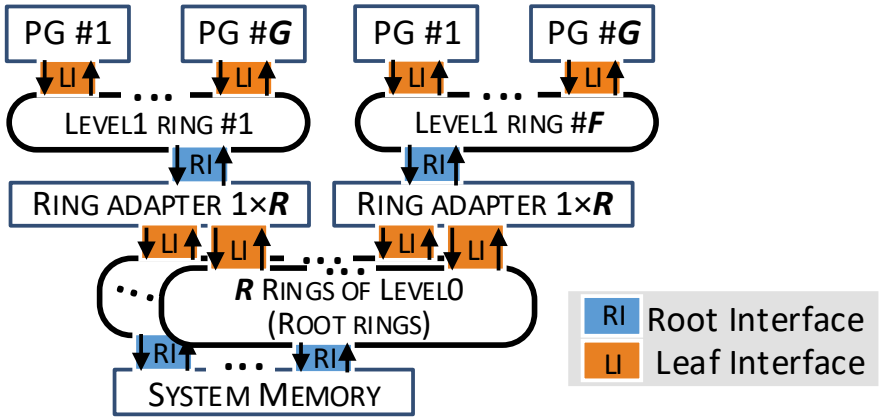


Fig. VII.1. Simulated RingNet topology.

In Section 4.2.5, memory-oriented SoCs are pointed out as the target application for the RingNet network. As SDRAM is widely supported by modern FPGAs (cf. Section 2.2.2), RingNet architecture is designed with the aim of fully exploiting its potential, i.e., it offers throughput which matches or exceeds the throughput of all kinds of SDRAMs supported by modern FPGAs. The throughput of RingNet is controlled by the ring multiplication mechanism discussed in Section 5.2.2. As discussed in Section 5.2.3, the throughput of the RingNet network matches the throughput of the most demanding SDRAM type for 4 parallel rings used at the root level of the RingNet network. Therefore, the range of the multiplication degree of the root level (R) used in simulation tests is restricted to the practical interval 1 – 4.

In the RingNet configuration depicted in Fig. VII.1, number N of interconnected PGs equals $F \cdot G$. The size of the RingNet network simulated in this chapter is limited by the capacity of considered FPGAs. A few largest FPGA devices are over one million logic cells (LCs) in size (cf. Table II.1), on the other hand, one thousand LCs per PE is the approximated cost of the RingNet network for those FPGA architectures (cf. synthesis results for RingNet modules presented in Section 8.3). As NoC is just a part of SoC, it can utilize only a part of the available resources. NoC can utilize as little as 6.6% of resources utilized by SoC [Dal01]. For this ratio, SoC comprising 75 PEs interconnected with the RingNet network will utilize 1.1 million LCs. Therefore, the author decided to simulate networks of up to 75 PEs, as it can cover the needs of many FPGA-based SoCs. As more comprehensive statistics for actual SoCs implemented in FPGAs are not easily available to the author, the above considerations demonstrate just a clever estimate of the limits of FPGA-based SoC. Nevertheless, for SoCs with more than 75 PEs,

RingNet network performance can be estimated by extrapolating the results presented for smaller networks, which is demonstrated in Section 7.2.

The results from each simulation are collected for the steady-state of a network according to the methodology discussed in [Dal03]. For this purpose, three phases of each simulation are distinguished: warm-up, measurement, and drain. For all phases, PGs try to insert packets into the network according to the abovementioned traffic pattern. First, the warm-up phase is conducted that allows the network to reach a steady-state. The steady-state is reached when the average buffer occupancy reaches its steady-state, i.e., the average buffer occupancy stops changing. During the warm-up phase, statistics are not collected. Next, the measurement phase starts. All the packets emitted by PGs during this phase are observed and information on their latency is collected, i.e., clock cycles that elapse between the emission of a packet header flit from a PG, and the acceptance of the last flit of the corresponding response packet (sent by the System Memory) by the PG. Packets received by each PG during the measurement phase are counted for the purpose of throughput computation. In the drain phase, new packets are emitted by PGs but their statistics are not included in the simulation results. The purpose of the drain phase is to keep the steady-state of the network until all the response packets that correspond to the packets that were emitted during the measurement phase reach their destinations.

In this thesis, latency is measured in terms of clock cycles. In many cases it is more convenient to express latency in seconds. Nevertheless, using clock cycles makes latency independent from the frequency of a clock signal which can be different for various FPGAs. It is still possible to translate clock cycles into seconds if it is needed by dividing the former by the clock frequency. The maximum clock frequency for RingNet synthesized for various FPGAs is reported in Chapter 8.

The Verilator compiler [Sny17] is used to convert Verilog code into C++ code that is further compiled by the GCC compiler and executed to perform functional simulations. The presented results summarize the simulations of over 3000 different configurations of network size and load.

7.2. Performance test

Achievable throughput and latency are basic performance parameters reported for NoCs. In this test, the achievable throughput of RingNet is checked together with the average latency that

can be expected under different conditions. Value of the theoretical maximum throughput for read and write logical channels can be calculated according to (1). Achievable throughput may be lower than the theoretical maximum throughput due to various factors, e.g., flow control or switching logic may require few clock cycles to elaborate control response, what may leave controlled channel idle for some time effectively lowering network throughput. Purpose of this test is to demonstrate that the achievable throughput of RingNet NoC matches the theoretical maximum throughput, and therefore it can be estimated during network configuration.

In the experiment, RingNet with two levels of rings is simulated. All packets are sent between PGs and the System Memory. Only packets with the lowest priority (priority 0) are generated.

The parameters of the test are:

- *R*: Multiplication degree of the root level, i.e., number of parallel rings used at the root level, is restricted to the interval 1 – 4.

- *F*: The number of 1st level rings is from the interval 1 – 5.

- *G*: The number of PGs connected to a single 1st level ring is set in the range of 1 – 15. Up to 75 PEs are connected for *F*=5 and *G*=15.

- Logical channel load. The aggregated load generated by all PGs is set in the range of 0% – 100% of the theoretical throughput T_{RW_MAX} (1). The logical channel load is separately set for the read and write channels.

In Table VII.1, the average latency for the read channel is presented for a high load (92% to 97%) and various network sizes (*R*=1, *F*=1–5 × *G*=1–15). One can see that latency increases with the increased number of connected 1st level rings and PGs.

TABLE VII.1

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT, AND WRITE AND READ CHANNEL LOADS IN RANGE 92% – 97%

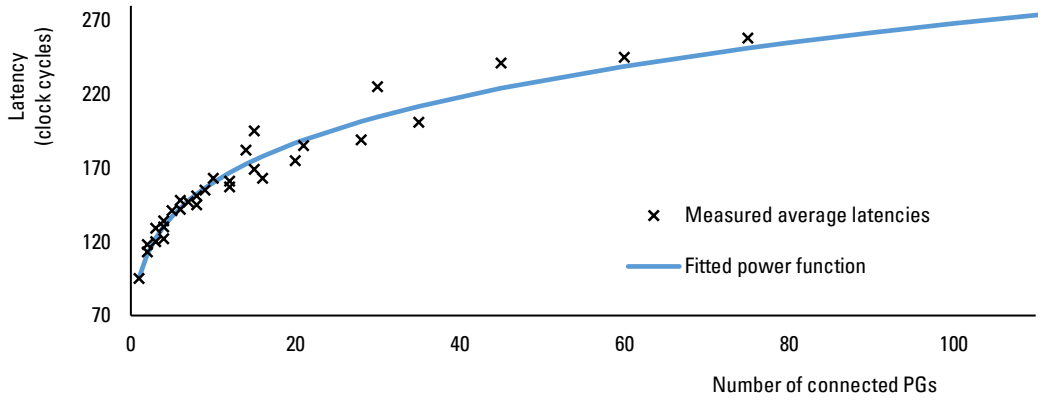
		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	95	118	120	122	147	194
	2	113	134	142	145	182	225
	3	129	148	155	161	185	241
	4	130	151	157	163	189	245
	5	140	163	169	175	201	258

For the write channel, the measured latencies are, on average, 7 clock cycles longer than those reported in Table VII.1. The results for the write channel and for the increased number of parallel rings used at the root level (R=2–4) are provided in Appendix V. It is tested that increasing the root ring multiplication degree by one increases the average latency reported in Table VII.1 by only 6 clock cycles (see Appendix V for details).

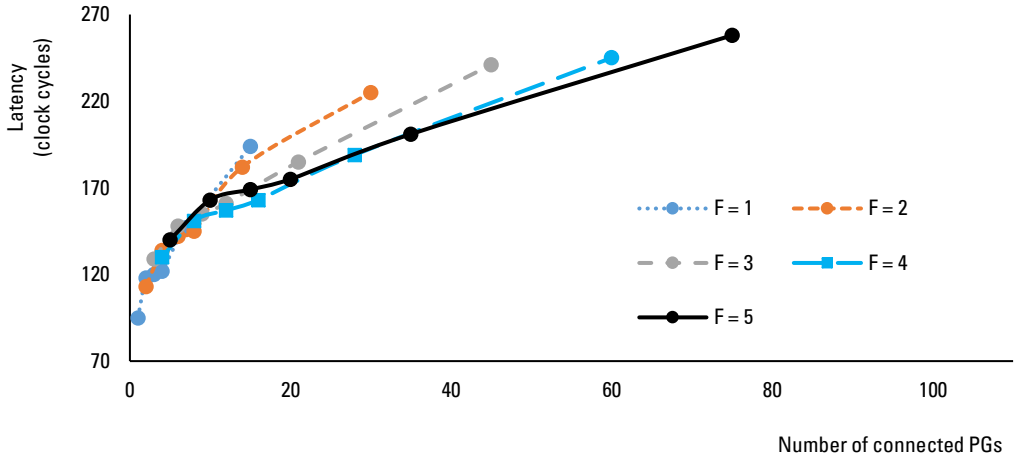
In Fig. VII.2, the latencies from Table VII.1 are presented as a function of the number of connected PGs. In Fig. VII.2a, the fitted latency function flattens for growing number of PG, i.e., for large networks, the latency increases slowly with the growth of the network. In Fig. VII.2b, the same data points from Table VII.1 are depicted, this time grouped in latency curves, one for each tested number of 1st level rings (F). For fewer than ten connected PGs, the curves overlap, i.e., similar latencies are observed for similar numbers of connected PGs, regardless of the number of 1st level rings used. As the number of connected PGs increases, the latency curves separate. The increased number of 1st level rings results in a lower slope of the latency curve. It can be concluded that for a given number of PGs, lower latency can be expected if more RingNet rings are used to interconnect them. This conclusion is in line with the idea presented in Section 5.2.1. In Section 5.2.1, the tree-of-rings topology is introduced, aiming at the reduction of the high latency present in the pure ring topology.

The results presented in Fig. VII.2b apply to the read logical channel for a high network load and a single ring used at root level of the tested network. It is tested that for the write logical channel, other network loads, and greater number of parallel rings the relation between the number of 1st lever rings and latency is similar to the presented case.

The number of RingNet rings used affects latency and resource utilization. On the one hand, increasing the number of used rings is expected to reduce the average latency. On the other hand, more rings connecting the same number of PE utilize more resources (cf. synthesis results in Section 8.3.2). Therefore, a decision on the number of used rings should depend on the acceptable latency level and the resource budget.



a) Latency as a function of the number of PG.



b) Latency curves, one for each tested number of 1st level rings (*F*).

Fig. VII.2. Average latency (expressed in clock cycles) for read channel for various network sizes.

NoCs are often characterized by providing load-latency curves that represent packet latency as a function of network load [Son03], [Ber04], [Yoo13], [Pos13], [LiuS14], [Hel15], [Pap15], [Kap15], [Kum16].

Fig. VII.3 depicts load-latency curves for the logical channels of the RingNet network with five 1st level rings and 15 PGs connected at each ring (F=5 × G=15, 75 PEs connected) and 4 parallel rings used at the network root (R=4). The results for other network configurations are provided in the supplementary material as Appendix VI.

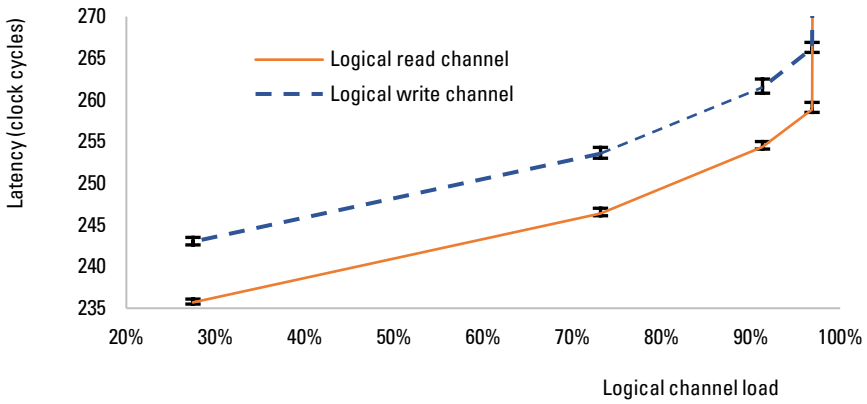


Fig. VII.3. Load-latency curves for RingNet with 75 PGs connected (F=5 × G=15) and four parallel rings used at the network root (R=4). These curves represent the average latency of each logical channel as a function of the channel load. Error bars represent the minimum and maximum average latencies of the channel at a given load when the second channel load changes in the range of 0% to 100%.

The average latency for both logical channels increases with the channel load. For a 100% channel load, all network buffers are full, and the average latency increases drastically, which seems to be obvious. On the other hand, an increase of the channel load from 27% to 97% results in the average latency increase by only 10%.

The inter-channel dependency is checked in this simulation and depicted in a form of error bars in Fig. VII.3. Those error bars represent the range of the average latency of the channel when the second channel load changes in the range 0% to 100%. One can see that both logical channels are independent, i.e., the load of one logical channel has a negligible impact on the average latency in the other logical channel, and does not influence its throughput.

For the presented network configuration (and for all other tested configurations), throughput T_{RW_MAX} calculated according to (1) is obtained for both the read and write channels, regardless

of the multiplication degree of the root ring (*R*), the number of 1st level rings (*F*), and the number of connected PGs (*G*) (cf. Appendix VI).

The most important conclusion drawn from the test is that the theoretical throughput T_{RW_MAX} (1) is guaranteed for RingNet, and the latency is correlated with network size, and it can be estimated during network configuration.

The performance of RingNet is compared with the performance of the state-of-the-art networks. For RingNet, an average latency of about 90 clock cycles is observed for a network with just one PE and increases to about 250 clock cycles when 75 PEs are connected. Negligible changes in latency are observed for traffic loads set in the range of 27% to 98%. For a mesh NoC with 64 PEs described in [Mai15], an average latency of 60 clock cycles is reported. Next, a latency of about 10 clock cycles is reported for CONNECT [Pap15] mesh with 16 PEs. The relatively high latency observed in the RingNet network is a result of two factors. First, the RingNet ring topology features relatively long paths when compared with a mesh. Next, latency is caused by the applied flow control mechanism that exchanges flow control messages before a packet can be sent. Therefore, RingNet is not recommended for designs requiring very low latency. Nevertheless, the applied flow control mechanism is demonstrated to provide a fair network access not reported for the state-of-the-art NoCs [Mai15] and [Pap15]. Moreover, in RingNet all the data is send to and from the System Memory. This is a special case of permutation traffic pattern, where all PEs send packets to just one destination. For both state-of-the-art NoCs [Mai15] and [Pap15], results for random traffic pattern only are reported in the source papers. From [Dal03] we know that random traffic pattern balances load even for topologies and routing algorithms that normally may have poor load balance. On the other hand, permutation traffic pattern stresses topology and routing algorithm because it concentrates load on individual paths. The author expects that in the case of permutation traffic pattern, like the one used in RingNet, latency for both state-of-the-art NoCs [Mai15] and [Pap15] can be higher than reported their source documents.

In order to compare the throughputs of RingNet and the NoCs proposed in [Mai15] and [Pap15], from the perspective of the memory-oriented SoCs, the author considers a traffic scenario where all PEs send packets to just one destination, i.e., a single memory device connected to the network. In such a simplified case, the throughput of the network is at most equal to the throughput of a network interface through which the memory is connected.

Whereas, the throughputs of a network interface for NoCs from [Mai15], [Pap15], and for RingNet are proportional to the flit width and frequency of a clock. In the following Chapter 8, it will be demonstrated that RingNet features substantially higher maximum frequency and uses less resources for the same flit width. It means that in the traffic scenario where all PEs sends packets to just one destination, RingNet provides higher throughput than the state-of-the-art NoCs from [Mai15] and [Pap15].

7.3. Network access fairness test

In test 7.2, the average parameters over all processing elements (PEs) are estimated. In real network applications, fair access to NoC for each PE can be an even more important aspect than the average parameters. Two types of fairness can be distinguished: latency fairness and throughput fairness. Latency fairness means that all PEs should experience the same latency, whereas throughput fairness means that PEs should experience throughput proportional to their relative request rates. From the literature it is known that fairness depends on the network load [Dal03]. In the case of a network experiencing such a low load that a single packet is transmitted through the network at a time, throughput is always granted according to request. For such a case, throughput fairness is observed. In the same case of a low load, latency fairness depends on the length of paths in a network and the routing algorithm and is not influenced by the flow control algorithm. As the network load increases and approaches 100%, the flow control mechanism starts to balance the traffic from different sources and its impact on fairness increases. The aim of this test is to examine the fairness of the RingNet network for a wide range of loads, especially for loads approaching 100%.

In the test, a RingNet network with 75 PEs is simulated (F=5 × G=15), and four parallel rings are used at the root level (R=4). PEs are simulated using packet generators (PGs) requesting the aggregated load of 27% to 100% of the logic channel throughput T_{RW_MAX} (1) for both logic channels. In this experiment, all PGs send packets to the System Memory with priority 0. In order to check the fairness, the statistics need to be gathered for each individual PG. For each PG, the author collects the values of the average latency L_{PG} expressed in clock cycles, and the values of throughput T_{PG} expressed in bits per clock cycle, for both logical channels.

In Table VII.2, the average values of L_{PG} and T_{PG}, calculated over all PGs ($\overline{L_{PG}}$ and $\overline{T_{PG}}$, respectively), are presented together with a standard deviation for those variables ($\sigma_{\overline{L_{PG}}}$ and $\sigma_{\overline{T_{PG}}}$, respectively).

TABLE VII.2

TRAFFIC STATISTICS OVER 75 PGS

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	236	6	243	6	0.7	0.01	0.68	0.01
97%	259	5	267	5	2.4	0.01	2.4	0.01
100%	1157	7	1175	9	2.5	0.00	2.48	0.00

Low values of the standard deviations mean that all PGs are expected to experience the same performance expressed in terms of the average latency and granted throughput. The results demonstrate that RingNet provides fair access for each connected PE. These conclusions are valid for a wide range of loads, nevertheless, different mechanisms are responsible for fairness for different loads values. For moderate load of 27%, most time slots in RingNet ring are empty and throughput can be granted as requested resulting in throughput fairness. In simulated RingNet topology, each PG is in the same distance from System Memory, i.e., total number of network switches that are passed, first by a packet on its way to System Memory, and second by a response packet on its way back to the PG, are equal for each PG. For moderate load of 27%, the equal distance from each PG to System Memory is enough to provide latency fairness. For high load of 97% and for network in saturation, throughput and latency fairness are still observed. For such high loads, flows from different PGs constantly compete for network resources and the traffic is successfully balanced in each ring by the flow control mechanism described in Section 6.3.1.

More results that demonstrate the fair access for other network sizes are available in Appendix VII.

7.4. Latency distribution test

In Sections 7.2 and 7.3, average latencies observed in RingNet network are discussed. It can also be informative to research the distribution of packet latency. Outliners in latency distribution may indicate problems with flow control mechanism. In the case of RingNet ring, outliners in latency distribution can indicate that some packets are constantly rejected by Root Interface (RI) and cycle around the ring. As stated in Section 6.3.1.2, special mechanism is implemented in Leaf-to-Root Manager, which should prevent continuous rejection of the same packet. In this section, latency distribution for a wide range of loads is studied, with the objective to confirm proper work of applied flow control mechanism.

A network with five 1st level rings (F=5) and 15 PGs connected at each ring (G=15) is simulated under the loads of 27% to 97%. Four parallel rings are used at the root level (R=4).

Fig. VII.4 depicts a histogram of packet latency for the read logical channel. The data are collected for a sample PE (PG connected to a 1st level ring as a 4th PE). For the write channel and for other PGs, the histograms look similar, especially width of latency distributions are of the same order. Those similarities are reasonable in view of the fact that compliance to the rule of latency fairness was demonstrated in test 7.3.

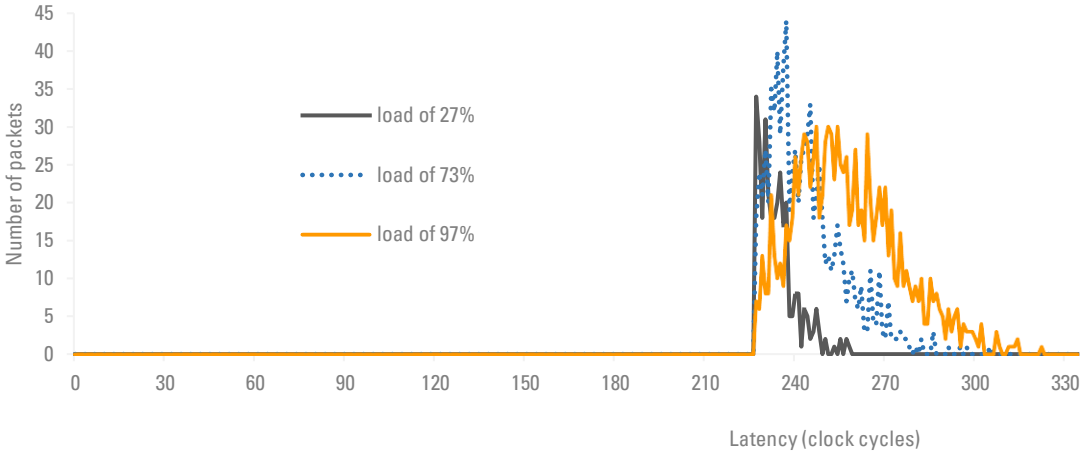


Fig. VII.4. Histogram of packet latency for read channel for an example of PE.

The values of 227 and 322 are the minimum and maximum latency observed, respectively. For low loads, more packets experience latency close to the minimum. An increase of the load

results in shifting the histogram towards higher latencies. The difference between the extreme values is 95 clock periods. The width of the observed latency distribution can be put into perspective by comparing it with characteristic delays introduced by properly functioning flow control mechanisms. For example, postponing the packet insertion to a ring due to occupancy of a time slot increases latency by 11 clock periods, as it is a distance in time that separates two consecutive time slots from the same logical channel (see Section 6.3.1.1). Latency can be increased multiple times when a number of time slots are occupied in a row, which is likely for a highly loaded network. Another example of a flow mechanism that increases latency is packet rejection that may happen in the Root Interface in the absence of available buffer space (see Section 6.3.1.2 for details). The rejected packet needs to cycle a ring and the introduced delay depends on the size of the ring. For a ring with 15 Leaf Interfaces (LIs), which is used as the 1st level ring in this test, the introduced delay equals 49 clock periods. For a ring with 5 LIs, which is used at the root of the network tree, this delay equals 23 clock periods. Both kinds of delay can be introduced at each level of the RingNet tree. Both discussed flow control mechanisms are likely to introduce delays under high loads. This is observed in Fig. VII.4 as the histogram shift towards higher latencies. The observed difference between the lowest and the highest latency can be a result of expected delays introduced by the flow control mechanism, therefore its correct operation cannot be questioned.

7.5. Packet prioritization test

The proposed RingNet architecture supports packet prioritization. Higher priority packets obtain access to the Leaf-to-Root (L2R) channels first. Under high load conditions, the high priority packets should experience lower latency; also, the requested throughput should be granted starting from the highest priority requests. To check the prioritization mechanism, a test is conducted under high load conditions:

- Network with 28 PGs is tested (F=4 × G=7).

- Two parallel rings are used at the root level (R=2).

- Each PG has 3 internal sources of packets with priorities 0, 1, and 3. The source of packets with priority 0 constantly tries to send a packet, and on its own it would generate the load of 100% of the theoretical throughput T_{RW_MAX} (1). The sources of packets with priority 1 and 3 are set to request various loads.

In Fig. VII.5, examples of load shares are presented for sources of packets with priority 1 requesting 20% of the throughput, and sources of packets with priority 3 requesting from 0% to 100% of the throughput. For the highest priority packets, the share of throughput is always granted as requested. 20% of the throughput, requested for priority 1 packets is granted when the packets with the highest priority use less than 80% of the RingNet throughput. The example demonstrates that the lower priority packets do not influence the share of throughput granted to the higher priority packets. It needs to be emphasized that using the prioritization mechanism does not limit the aggregated throughput below the theoretical value T_{RW_MAX} (1) in any tested case.

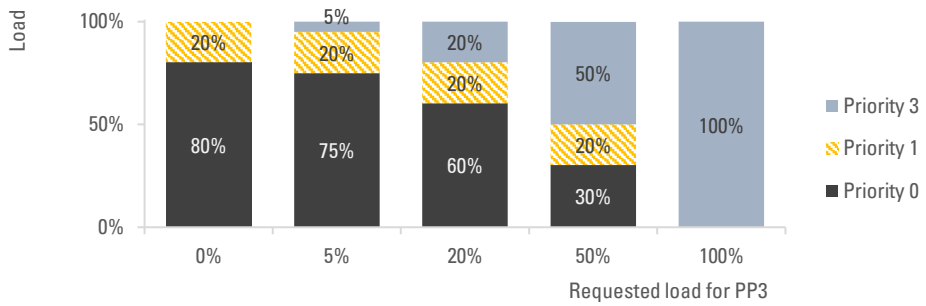


Fig. VII.5. Load, expressed as percentage of the theoretical throughput TRW_MAX (1), granted to each priority under constant requested load for priorities 0 and 1 and increasing requested load for priority 3.

It needs to be emphasized that introducing packets with different priorities did not limit the theoretical aggregated throughput in any tested case.

In Fig. VII.6, latency histograms are depicted for packets with different priorities. The presented statistics are collected for the case when packets with priority 3 and 0 utilize 5% and 95% of the network throughput, respectively. The histograms for both priorities are concentrated. The high priority packets experience moderate latencies even under the maximum network loads.

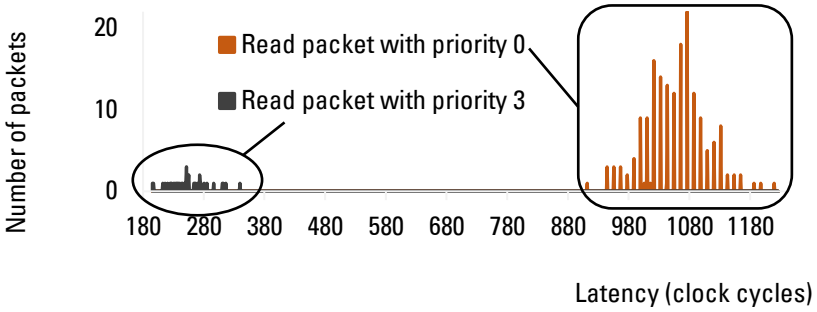


Fig. VII.6. Latency histogram for read channel.

7.6. Simulation summary

The proposed communication protocol with prioritization and flow control mechanisms was tested in a number of simulations. The results prove that the RingNet network throughput can be calculated according to (1) and the average latency can be estimated based on NoC size. Moreover, the demonstrated fair access to RingNet network is a distinctive property among other NoCs proposed for FPGA.

In addition to the simulations, RingNet was tested in a hardware application. It was successfully used as a communication backbone for an FPGA-based depth map estimation device [Dom15] demonstrating the above-mentioned features.

Chapter 8. RingNet synthesis

This chapter summarizes RingNet synthesis for various FPGAs and tests RingNet scalability. The objective of RingNet synthesis is to determine the usability of RingNet architecture for various FPGA architectures.

8.1. Methodology

The author of the dissertation synthesizes individual RingNet modules and rings of different sizes that are useful for networks with a tree topology, i.e., rings with one Root Interface (RI) and from 2 to 15 Leaf Interfaces (LIs) (Ring 2×1 – Ring 15×1). In the proposed RingNet implementation 15 is the maximum number of LIs that can be addressed in a ring (see Section 6.3.2). Resource utilization and the maximum clock frequency is reported.

8.1.1. Representation of FPGAs

RingNet is synthesized for chosen FPGAs from Xilinx (Artix7, Kintex UltraScale, Virtex7), Intel (Stratix V, Arria V) and Lattice (ECP5). The newest devices from Xilinx (UltraScale+ series) are not supported by the synthesizer used in the research (Synplify Premier 2017.03), therefore they could not be tested. Nevertheless, the tested FPGAs provide examples of distinctive architectures used by each vendor (cf. Table I.1 and Appendices I – III).

Providing synthesis results for FPGA architectures offered by three of the leading FPGA manufacturers is exceptionally comprehensive, as for other NoCs architectures synthesis results are provided for individual FPGA device types [Kap15], [Ret14], [Mai15], [She14], [Mai17], [Red19] or for single FPGA device lines [Pap15]. The objective of RingNet synthesis is to determine its usability as an inter-FPGA compatible NoC architecture.

8.1.2. Synthesis software

For fair comparison, the same synthesis software – Synplify Premier 2017.03-SP1 – is used for all devices. According to its producer, Synplify is the industry's most advanced FPGA design and debug environment [Syn19]. Synplify Premier is chosen mostly for its multi-FPGA vendor support, i.e., syntheses for FPGAs from Achronix, Intel, Microsemi, Xilinx, and Lattice are supported [Syn17]. Lattice uses Synplify as a default synthesis tool in its FPGA design and

verification environment called Lattice Diamond. Moreover, Synplify Premier reports synthesis results in a unified format, which eases the comparison between various FPGA architectures.

Synplify contains many Electronic Design Automation (EDA) optimization tools, like the shift register inferring and the register and logic replication [Syn17]. Those tools can balance the resource cost and clock frequency, nevertheless, the tools perform differently for different devices. With the aim of obtaining fair results for all devices, the author disables the shift register inferring tool. Also, the register and logic replication tools are effectively disabled by setting the requested frequency to a low value of 1 MHz.

8.1.3. Reference for maximum clock frequency

RingNet NoC is designed for high-throughput applications, e.g., for multimedia processing systems. The throughput of a RingNet network is proportional to the clock frequency, therefore, keeping the maximum applicable clock frequency as high as possible is important for RingNet implementation performance. In Section 5.2.2, the multiple physical technique designed for RingNet is discussed that aims at increasing RingNet throughput beyond the throughput supported by a single ring. Nevertheless, this technique comes at the cost of increased FPGA resource usage, therefore, it should be applied when adjusting the throughput using clock frequency is not possible.

In Section 8.3 the maximum clock frequencies obtained for RingNet modules are presented for various FPGA architectures. On the one hand, the obtained results should be compared in order to evaluate RingNet performance across various FPGA architectures. On the other hand, each type of FPGA device has its own frequency characteristics related to the applied technology. The quantitative comparison of RingNet performance across various FPGA devices should consider the frequency characteristics of those FPGAs. On way to accomplish this objective is to normalize the obtained maximum clock frequency of RingNet modules with respect to the maximum clock frequency of a given FPGA device. Nevertheless, the maximum clock frequency of an FPGA device is not reported in the literature. One reason for that is the great complexity of FPGA architecture and a great number of different frequencies that characterize different FPGA resources (LUT, multiplexer, FF, DSP, buffer, etc.) [Alt11]–[Alt16], [Int16], [Int17], [Xil16a]–[Xil17], [Lat13]–[Lat16]. Moreover, the maximum clock frequency for a resource depends on its configuration. Which of the many characteristic

frequencies should be used as the intrinsic maximum frequency of a given FPGA is not clearly specified.

The authors of [K12] suggest that the maximum clock frequency of a single DSP block is the intrinsic frequency of an FPGA device. Therefore, it is used in this dissertation as a reference for the maximum clock frequencies obtained for RingNet modules. For better representation of FPGA frequency characteristics, the maximum frequency of BRAM is also used for reference.

BRAMs and DSPs can be used in many configurations exhibiting different maximum clock frequencies. In this dissertation, the maximum clock frequencies of the fastest, fully pipelined configurations of DSPs and BRAMs are used:

- The maximum frequency of the fastest, fully pipelined configuration of a DSP block (optional input, output and intermediate registers are utilized),
- The maximum frequency of fully pipelined BRAM (optional input and output registers are utilized) in two configurations: one with simultaneous read and write to the same address handled in additional logic (*RW check* configuration), and second without this additional logic (*no RW check*). Usually, the *RW check* configuration supports a lower maximum clock frequency than the *no RW check* configuration, due to the delay introduced in signal paths in the additional logic.

8.1.4. Types of FPGA resources reported for RingNet syntheses

In the literature, a number of utilized FFs and LUTs are used to compare competing NoCs [She14], [Ret14], [Mai15], [Pap15], [Kap15], [Mai17], [Red19]. Also in the dissertation, the author reports the utilization of those resources for RingNet implemented in various FPGAs. Nevertheless, it needs to be emphasized that LUT architectures used by individual vendors are different. Moreover, routing for LUTs and FFs is different for FPGAs from individual vendors. It is clear that comparing the number of FFs and LUTs used in different FPGAs can be meaningless without a thorough analysis of the differences between FPGA architectures. Such an analysis of available architectures and their impact on expected LUT and FF utilization is considered in Section 8.2.

8.2. Utilization of LUTs and FFs in different architectures

The FPGA architectures applied by individual vendors are described in Appendices I to III. In this section, the impact of the differences between the architectures on LUT and FF utilization is analyzed for general FPGA-oriented designs and for the RingNet design.

Through Sections 8.2.1 to 8.2.3, the author estimates the expected ratios between resources utilized for a design implemented in different FPGA architectures. The author distinguishes between LUTs utilized as LUTRAM and LUTs utilized for logic implementation. Those are discussed in Sections 8.2.1 and 8.2.2, respectively. The utilization of FFs is discussed in Section 8.2.3.

8.2.1. Utilization of LUTs for LUTRAM implementation

Memory blocks utilizing LUTRAM can be implemented in single-port, dual-port or quad-port configurations [Xil16e], [Int18], [Lat15b]. A special case of the dual-port configuration is called a simple dual-port (also called pseudo dual-port). In this configuration, two ports are available, and the first port is used for writing, while the second port is used for reading. RingNet, and other designs, exploit memories in the simple dual-port configuration for first-in-first-out (fifo) buffer implementation. Table VIII.1 summarizes the simple dual-port configurations available in FPGAs from individual vendors.

TABLE VIII.1

LUTS UTILIZED FOR SIMPLE DUAL-PORT MEMORY

Vendor	LUT type	The smallest simple dual-port memory		Average memory bits per LUT
		Number of utilized LUTs	Memory size	
Xilinx	LUT6	4	3b×64-word deep or 6b×32-word deep RAM	48
Intel	LUT6	20	20b×32-word deep or 10b×64-word deep RAM	32
Lattice	LUT4	6	4b×16-word deep RAM	10.7

From the average memory bits available per LUT, one can estimate the ratio of LUTs utilized in FPGA of one vendor to LUTs utilized in FPGA of another vendor for the simple dual-port memory (see Table VIII.2).

TABLE VIII.2

RELATIVE NUMBER OF LUTS UTILIZED FOR SIMPLE DUAL-PORT MEMORY, ESTIMATED FOR FPGAS OF DIFFERENT VENDORS

Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
1.5 : 1	4.5 : 1	3 : 1

From Table VIII.1 one can see that the size of the smallest memory block differs from one vendor to another. The size of the memory block possible to be implemented is quantized and must be an integer multiple of the sizes presented in Table VIII.1. If the size of a memory block required in a design does not fit the implementable block sizes, then a bigger block is instantiated and part of the block is wasted. For example, a buffer in the RingNet Root Interface requires a 72b×32-word deep memory block. In an FPGA by Xilinx, it can be implemented as 12 blocks of 6b×32-word deep RAM. In an FPGA by Lattice, 36 blocks of 4b×16-word deep RAM can be used. For both architectures no memory will be wasted. However, in the Intel architecture, 4 blocks of 20b×32-word deep RAM will be instantiated, while the capacity of 3.6 would be enough. This way 0.4 of the 20b×32-word deep LUTRAM will be wasted due to quantized sizes of implementable memory blocks.

The size of the wasted part of instantiated LUTRAM differs from case to case and should be calculated individually. Nevertheless, on average, the wasted part will be smaller if the step of the memory size is smaller, i.e., the size of the smallest LUTRAM block is smaller. Lattice uses LUTRAM with the smallest step in width and depth, i.e., 4 bits and 16 words, respectively. The smallest depth of LUTRAM used by Intel and Xilinx is 32 words. For this depth, the step of width for Xilinx equals 6 bits, whereas for Intel the corresponding step is greater and equals 20 bits. On average, due to the quantized sizes of implementable LUTRAM, the wasted part of instantiated memory will be the greatest in Intel devices, and the smallest in Lattice FPGAs. This can affect the expected ratios presented in Table VIII.2. Updated estimates of the ratios are presented in Table VIII.3.

TABLE VIII.3

RELATIVE NUMBER OF LUTS UTILIZED FOR SIMPLE DUAL-PORT MEMORY, ESTIMATED FOR FPGAS OF DIFFERENT VENDORS

Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
1.5↑ : 1	4.5↓ : 1	3↓ : 1

Suffix "↑" indicates a value greater or equal to presented, and suffix "↓" indicates a value lower or equal to presented.

RingNet modules use simple dual-port memories of the following sizes:

- 72b×64-word deep, used in Leaf Interface.
- 72b×32-word deep, used in Root Interface and Ring Adapter.
- 72b×16-word deep, used in Slot Generator.
- 12b×16-word deep, used in Slot Generator and in Leaf Interface.
- 8b×16-word deep, used in L2R Manager.

For the presented memory blocks, their implementation in the Lattice architecture gives LUTRAM with the least wasted capacity, more LUTRAM capacity is wasted in the Xilinx architecture, whereas the implementation in the Intel architecture suffers from the greatest waste. This is in line with the general conclusion about the rate of wasted-to-used LUTRAM capacity in FPGAs of different vendors. Therefore, LUTRAM utilized for the RingNet network is expected to follow the estimated ratios presented in Table VIII.3.

8.2.2. Utilization of LUTs for logic functions implementation

A logic function implemented in an FPGA device utilizes LUTs. The number of utilized LUTs depends on the FPGA architecture and the function to be implemented. In particular, the number of available inputs of a single LUT and the number of inputs of the function are the most important factors.

Lattice uses LUTs with four inputs (LUT4s), whereas Xilinx and Intel offer LUTs with six inputs (LUT6s). Without the knowledge about the functions to be implemented, the expected ratio of LUTs utilized in FPGA of one vendor to LUTs utilized in FPGA of another vendor can be estimated roughly. The ratios are presented in Table VIII.4.

TABLE VIII.4

RELATIVE NUMBER OF LUTS UTILIZED FOR LOGIC, ESTIMATED FOR FPGAS OF DIFFERENT VENDORS

Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
1 : 1	1↑ : 1	1↑ : 1

Suffix "↑" indicates a value greater or equal to presented.

The presented ratios are rough and may be insufficient for predicting LUT utilization for different vendors. More accurate ratios can be estimated when the following knowledge is applied: 6-input and 5-input functions utilize a single LUT6 in FPGAs by Xilinx and Intel, whereas the same functions implemented in Lattice architecture utilize 4 and 2 LUT4s, respectively. However, this knowledge can be utilized only for a design with a known share of 6-inputs and 5-inputs functions. For the RingNet network about 15% of all functions have 6 inputs and 16% have 5 inputs, the rest have 4 or fewer inputs. Considering this knowledge, Table VIII.5 is presented with the expected ratio of LUTs utilized for logic in the RingNet network.

TABLE VIII.5

RELATIVE NUMBER OF LUTS UTILIZED FOR LOGIC IN RINGNET NETWORK, ESTIMATED FOR FPGAS OF DIFFERENT VENDORS

Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
1 : 1	1.62 : 1	1.62 : 1

As stated in Section 8.1.4, in the literature, FPGA resource usage is reported mostly in terms of the number of utilized FFs and LUTs. Nevertheless, the primary resource of FPGA is a logic block, which consists not only of LUT and FF, but also additional hardware resources, e.g., multiplexers, carry logic or adders (see introduction to FPGA architecture presented in Chapter 2). This additional hardware can implement simple functions, which otherwise utilize LUTs. Especially in the Intel architecture, 2×1 multiplexers are associated with each LUT (see Appendix II) and are utilized for 2×1 signal multiplexing. In Xilinx and Lattice architectures, such a multiplexing operation utilizes LUT. In RingNet, the 2×1 multiplexing amounts to at least 10% of all logic operations (it can be more, depending on the network size), therefore, utilizing the multiplexers can preserve a lot of LUTs. The ratios presented in Table VIII.6 consider the utilization of multiplexers in FPGAs by Intel for RingNet.

TABLE VIII.6

RELATIVE NUMBER OF LUTS UTILIZED FOR LOGIC IN RINGNET NETWORK, ESTIMATED FOR FPGAS OF DIFFERENT VENDORS

Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
0.91↓ : 1	1.62 : 1	1.78↑ : 1

Suffix "↑" indicates a value greater or equal to presented, and suffix "↓" indicates a value lower or equal to presented.

The presented estimate does not consider a special function of LUT6 by Xilinx. A single LUT6 in the Xilinx architecture can generate two functions of 5 or fewer inputs, especially two independent functions of 3 and 2 inputs can be generated (see Appendix III). The decision on exploiting this feature is made by the synthesizer, depending on its settings. In general, the synthesizer exploits this feature only when there is not enough LUTs for independent implementations of each function. It is not the case for synthesis scenarios presented in this dissertation, therefore its impact on the LUT utilization ratio is not discussed here.

8.2.3. FFs utilization

FFs are similar in all considered architectures. The proposed NoC should utilize approximately the same number of FFs at any considered FPGA. Small discrepancies can still occur due to the optimization performed by a synthesizer (a mechanism called register duplication can sometimes be used in order to increase the maximum frequency of a clock signal).

8.2.4. Summary

The ratios estimated in Sections 8.2.1 – 8.2.3 are useful for predicting resource utilization for designs transferred between different FPGA architectures. Moreover, the ratios can be used as a proofing tool, i.e., discrepancy in estimated and obtained resource utilization can indicate that the definition of a design prepared in a hardware description language (HDL) was mapped to unintended hardware structures. For example, a memory buffer intended to be mapped to LUTRAM can be mapped to FFs if specific HDL rules are not followed [Alt09].

In this dissertation, the presented ratios are used in Section 8.3.2, in which the author discusses resource utilization for RingNet modules for different FPGA architectures. Following

the ratios may confirm that the discrepancies in resource utilization do not come from any overlooked differences between FPGA architectures, which may be critical for the search for a universal FPGA NoC.

8.3. Synthesis results

The RingNet syntheses for various FPGAs are summarized in Table VIII.7, which reports the maximum clock frequency and the utilized FFs and LUTs. Utilized LUTs are presented for two categories: LUTs utilized as LUTRAM and LUTs utilized for logic implementation.

TABLE VIII.7

RESOURCES UTILIZATION AND ESTIMATED MAXIMUM FREQUENCY AFTER SYNTHESIS FOR RINGNET MODULES, RINGNET RINGS, AND THE STATE-OF-THE-ART SWITCHES

RingNet modules, RingNet rings, state-of-the-art. switches, and FPGA hardware blocks	FFs utilized / LUTs utilized as logic + LUTs utilized as RAM / Maximum clock frequency in MHz					
	Xilinx			Intel		Lattice
	Artix7 xc7a100tcsg324-1	Kintex UltraScale xcku060-ffva1156-3-e	Virtex7 xc7vx550tffg1158-1	Stratix V 5SGXMABK2H40C3	Arria V 5AGXBA7D6F31C6	ECP5 lfe5u_85f-8
Ring adapter 1×2	383 / 93 + 48 / 485	383 / 94 + 48 / 837	383 / 94 + 48 / 469	383 / 105 + 144 / 551	383 / 102 + 144 / 377	383 / 189 + 216 / 312
Ring adapter 2×1	462 / 184 + 96 / 459	462 / 184 + 96 / 837	462 / 184 + 96 / 486	462 / 123 + 288 / 529	462 / 127 + 288 / 345	462 / 351 + 432 / 312
L2R Manager	499 / 271 + 96 / 337	499 / 264 + 96 / 875	499 / 286 + 96 / 461	499 / 317 + 128 / 527	507 / 349 + 128 / 334	499 / 373 + 384 / 374
Slot Generator	106 / 11 + 56 / 487	106 / 11 + 56 / 837	106 / 11 + 56 / 487	106 / 14 + 168 / 983	106 / 12 + 168 / 468	106 / 16 + 126 / 372
Leaf Interface	759 / 227 + 104 / 405	759 / 218 + 104 / 807	759 / 227 + 104 / 509	759 / 145 + 166 / 524	761 / 160 + 166 / 293	761 / 391 + 450 / 235
Root Interface	648 / 125 + 48 / 469	648 / 125 + 48 / 837	648 / 125 + 48 / 483	648 / 56 + 144 / 779	648 / 57 + 144 / 381	648 / 204 + 216 / 314
Ring 2×1	2k8 / 805 + 424 / 354	2k8 / 801 + 424 / 753	2k8 / 805 + 424 / 445	2k8 / 631 + 916 / 449	2k8 / 648 + 914 / 273	2k8 / 1k3 + 1k4 / 235
Ring 4×1	4k3 / 1k2 + 632 / 363	4k3 / 1k2 + 632 / 700	4k3 / 1k2 + 632 / 460	4k3 / 934 + 1k2 / 430	4k3 / 966 + 1k2 / 250	4k3 / 2k0 + 2k3 / 235
Ring 6×1	5k8 / 1k7 + 840 / 333	5k8 / 1k7 + 840 / 763	5k8 / 1k7 + 840 / 440	5k8 / 1k3 + 1k6 / 409	5k8 / 1k3 + 1k6 / 262	5k9 / 2k8 + 3k3 / 235
Ring 8×1	7k3 / 2k1 + 1k0 / 328	7k3 / 2k1 + 1k0 / 726	7k3 / 2k1 + 1k0 / 440	7k3 / 1k6 + 1k9 / 397	7k3 / 1k7 + 1k9 / 272	7k4 / 3k6 + 4k2 / 235
Ring 11×1	9k5 / 2k8 + 1k4 / 333	9k5 / 2k8 + 1k4 / 753	9k5 / 2k8 + 1k4 / 446	9k5 / 2k1 + 2k4 / 434	9k6 / 2k2 + 2k4 / 241	9k7 / 4k9 + 5k8 / 235
Ring 15×1	12k5 / 3k7 + 1k8 / 325	12k5 / 3k7 + 1k8 / 763	12k5 / 3k7 + 1k8 / 443	12k5 / 2k7 + 3k1 / 403	12k5 / 2k8 + 3k0 / 222	12k8 / 6k5 + 7k6 / 235
Switch from [3]		--- / 1678 / 470				
Switch from [4]	--- / 430 / 241 (results for Virtex6 LX760 speed grade -2)					
DSP	/ 392	/ 687	/ 463	/ 400	/ 200	/ 185
BRAM *no RW check*	/ 388	/ 660	/ 458	/ 650	/ 285	/ 272
BRAM *RW check*	/ 339	/ 575	/ 400	/ 455	/ 240	/ 214

For each FPGA device, for RingNet rings and for hardware blocks, the lowest value of the maximal clock frequency is marked in red.

8.3.1. Maximum clock frequency estimation

The obtained frequency values are compared with the maximum frequencies specified by vendors for hardware blocks of each FPGA [Alt11]–[Alt16], [Int16], [Int17], [Xil16a]–[Xil17], [Lat13]–[Lat16]. The frequencies for the common DSP and BRAM blocks are presented. The maximum frequencies are given for the fastest, fully pipelined configuration of each hardware block. For BRAM, two versions are given: one with simultaneous read and write to the same address handled in additional logic (*RW check*), and another one without this extension (*no RW check*). The provided frequencies are a raw estimation of what maximum frequency can be expected for a typical, high performance project implementation for each device (cf. Section 8.1.3). From Table VIII.7, it can be seen that the maximum frequencies obtained for RingNet modules are comparable with or higher than those given for DSPs and BRAMs. In Table VIII.7, for each FPGA device, the lowest frequency reported for the RingNet rings and the lowest frequency reported for the hardware blocks are marked in red. On average, the frequency of the RingNet ring is 20% higher than the frequency of the hardware block. Nevertheless, in case of Artix7 and Stratix V devices, the frequency for the RingNet ring is lower than the frequency for the hardware block by 4% and 1%, respectively.

Still, the results are generated for the required clock frequency set to 1 MHz. If the required frequency is tuned, the Synplify synthesizer can optimize modules using techniques like register replication, and higher maximum frequencies can be obtained, gaining several dozens of MHz on average. The obtained maximum clock frequencies are estimations, and actual frequency values should be generated by EDA tools provided by a device vendor like Vivado from Xilinx (see Chapter 9).

8.3.2. Resource utilization

Resource utilization for basic RingNet modules and rings is reported in Table VIII.7. The utilization of flip-flops (FFs) and look up tables (LUTs) is presented. For all tested FPGAs from Xilinx, the number of utilized resources are almost equal for each FPGA, no matter which RingNet module or ring is considered. The same is true for FPGAs from Intel. This is expected and is a consequence of similar architectures of logic element used by a vendor for all his FPGAs (see Section 8.2 and the architectures description in Appendices I – III).

Aiming at brevity, the resource utilization results from Table VIII.7 can be averaged over each vendor's devices. This is eligible due to the above-mentioned similarities in resource utilization in all FPGAs from a single vendor. The averaged results for RingNet rings are presented in Tables VIII.8 – VIII.10. The results are expressed as a ratio of a resource utilized in FPGAs of one vendor to the same resource utilized in FPGAs of another vendor. In this form, the results can be compared with the ratios estimated in Section 8.2.

The utilization of FFs is presented in Table VIII.8. As discussed in Section 8.2.3, similar utilization of FFs is expected in all considered FPGA architectures. The utilization of FFs in different architectures is almost the same, i.e., the maximum deviation from the average module size is of 2%.

TABLE VIII.8

RELATIVE NUMBER OF FFS UTILIZED IN FPGAS OF DIFFERENT VENDORS

RingNet rings	Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
Ring 2×1	1 : 1	1.01 : 1	1.01 : 1
Ring 4×1	1 : 1	1.02 : 1	1.02 : 1
Ring 6×1	1 : 1	1.02 : 1	1.02 : 1
Ring 8×1	1 : 1	1.02 : 1	1.02 : 1
Ring 11×1	1 : 1	1.02 : 1	1.02 : 1
Ring 15×1	1 : 1	1.02 : 1	1.02 : 1
Average ring	1 : 1	1.02 : 1	1.02 : 1
Estimated in 8.2.3	1 : 1	1 : 1	1 : 1

In Table VIII.9, the utilization of LUTs used for logic is summarized.

TABLE VIII.9

RELATIVE NUMBER OF LUTS UTILIZED FOR LOGIC IN FPGAS OF DIFFERENT VENDORS

RingNet rings	Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
Ring 2×1	0.79 : 1	1.57 : 1	2.00 : 1
Ring 4×1	0.75 : 1	1.65 : 1	2.19 : 1
Ring 6×1	0.75 : 1	1.68 : 1	2.26 : 1
Ring 8×1	0.73 : 1	1.71 : 1	2.32 : 1
Ring 11×1	0.74 : 1	1.76 : 1	2.38 : 1
Ring 15×1	0.73 : 1	1.76 : 1	2.43 : 1
Average ring	0.75 : 1	1.69 : 1	2.26 : 1
Estimated in 8.2.2	0.91↓ : 1	1.62 : 1	1.78↑ : 1

Suffix "↑" indicates a value greater or equal to presented, and suffix "↓" indicates a value lower or equal to presented.

The number of LUTs utilized as logic in the Lattice device is 1.69 and 2.26 times greater than in devices from Intel and Xilinx, on average, respectively. The values are close to the ratios estimated in Section 8.2.2 (1.62 and more than 1.78, respectively). The main reason for the lower utilization of LUTs in Xilinx and Intel FPGAs is that 6-input LUTs are used in their architectures, whereas Lattice uses smaller, 4-input LUTs (cf. Section 8.2.2).

The number of LUTs utilized as logic for Xilinx devices is in the range of 0.73 to 0.79 of the number reported for Intel devices. In Section 8.2.2, the ratio was estimated to be less than 0.91. The differences in LUT utilization between Xilinx and Intel FPGAs are the result of differences in the architectures used by each vendor. In particular, Intel associates its LUTs with 2×1 multiplexers that are utilized instead of LUTs in the Leaf Interface and the Root Interface for multiplexing signals.

The number of LUTs utilized as RAM differs significantly between vendors, as it is presented in Table VIII.10. As discussed in Section 8.2.1, it is the result of different numbers of RAM bits available per utilized LUT. The obtained ratios for RingNet rings match the expected ratios.

TABLE VIII.10

RELATIVE NUMBER OF LUTS UTILIZED FOR LUTRAM IN FPGAS OF DIFFERENT VENDORS

RingNet rings	Intel : Xilinx	Lattice : Xilinx	Lattice : Intel
Ring 2×1	2.16 : 1	3.41 : 1	1.58 : 1
Ring 4×1	1.97 : 1	3.71 : 1	1.88 : 1
Ring 6×1	1.88 : 1	3.98 : 1	2.12 : 1
Ring 8×1	1.82 : 1	4.04 : 1	2.22 : 1
Ring 11×1	1.73 : 1	4.15 : 1	2.40 : 1
Ring 15×1	1.70 : 1	4.18 : 1	2.46 : 1
Average ring	1.88 : 1	3.91 : 1	2.11 : 1
Estimated in 8.2.1	1.5↑ : 1	4.5↓ : 1	3↓ : 1

Suffix "↑" indicates a value greater or equal to presented, and suffix "↓" indicates a value lower or equal to presented.

The average ratio between the number of utilized FFs and the number of utilized LUTs is equal to 2.2 : 1 for RingNet rings synthesized for Intel and Xilinx devices and 1 : 1 for Lattice. Those ratios are close to the ratios of FFs and LUTs available in the devices (see Table II.1) that the author finds desirable for an FPGA design.

For most NoCs, a significant part of FPGA resources is utilized for buffers. It is also true for RingNet, where buffers are implemented using LUTRAM, whereas RAM-capable LUTs are a limited resource (see Table II.1). The shares of LUTs that are RAM-capable equals 40%, 50%, and 50% for the tested FPGAs from Xilinx, Intel, and Lattice, respectively. As for any limited resource, using less LUTRAM can make routing of synthesized design easier and provide shorter links and higher clock frequency. The author checks what share of the LUTs utilized for RingNet rings is utilized for LUTRAM. This share happens to be constant for a given FPGA vendor: about 33% for rings synthesized for FPGAs from Xilinx, 55% for Intel and 54% for ECP5 from Lattice. The share of utilized RAM-capable LUTs exceeds the share of available RAM-capable LUTs, but only by 4% and 5% in the case of Lattice and Intel, respectively. In the case of NoC, which in general consume a significant part of resources for buffers, this shows reasonable LUTRAM utilization, suitable for FPGA implementation.

Through the presented synthesis results, it is shown that RingNet modules can be efficiently implemented in devices of various types from different vendors, with comparable resource utilization and a high value of the maximum clock frequency.

8.3.3. Comparison between state-of-the-art switches and RingNet network

In Table VIII.7, the results for the state-of-the-art 64-bit flits switches are presented for reference [Mai15], [Pap15]. The 5-port switch proposed in [Mai15] is reported to utilize 1678 LUTs per PE when implemented in the Xilinx Kintex UltraScale device. The CONNECT network using 3-port switches and configured in ring topology [Pap15] is reported to utilize 430 LUTs per PE when implemented in Xilinx Virtex-6 LX760. On the other hand, RingNet utilizes only 367 LUTs per PE if 15×1 ring is used. It is clear that 3-port switches used in RingNet and CONNECT utilize fewer LUTs than the 5-port switch from [Mai15]. In RingNet ring, unlike in [Mai15] and [Pap15], some network modules, i.e., Slot Generators and L2R Manager are common to the number of connected PEs. The cost of the modules is shared among the number of the connected PEs, which decreases the resources utilized per PE. It is the reason for fewer LUTs utilized per PE in RingNet when compared with CONNECT. It needs to be pointed out that the shared L2R Manager used in RingNet not only lowers resource utilization but also increases latency, which is discussed in Chapter 7. On the other hand, the L2R Manager has knowledge about the required load, therefore it can provide a fair access for all its associated PEs.

When implemented in the devices of the same speed class, the RingNet rings can be clocked with higher clock frequency than switches proposed in [Mai15], as well as the CONNECT network. This feature is achieved by the use of small and optimized 3-port switches in RingNet. For example, for the Kintex UltraScale Xilinx devices, the maximum clock frequency is above 700 MHz and about 470 MHz for RingNet and the network of switches [Mai15], respectively (cf. Table VIII.7). Another example shows that for a Virtex6 device, the maximum frequency for CONNECT network is about 31% – 59% of the maximum DSP frequency, whereas it is 84% to 107% of the maximum DSP frequency for a RingNet ring.

The RingNet network utilizes less resources than the NoCs from [Mai15], and [Pap15], but it comes at the expense of increased latency (see Chapter 7). On the other hand, the high latency can be mitigated with high maximum clock frequency of RingNet.

8.3.4. Summary

In Section 8.3, synthesis results are presented for sample FPGA devices representing architectures used by leading vendors. In Section 8.1.1, the clock frequency applicable to RingNet rings is checked to be equal or higher than 96% of the maximum clock frequency of FPGA hardware resources. In Section 8.1.2, resource utilization for RingNet modules and rings is presented for different FPGA architectures. The differences in resource utilization observed between the FPGA architectures follow the expected ratios. Therefore, it is most probable that no NoC-related, critical differences between FPGA architectures are overlooked at the design stage of the RingNet architecture. Obtaining high frequency NoC and predictable resource utilization shows that the identified common features of considered FPGA architectures are exploited, which results in inter-FPGA compatible NoC.

In Section 8.3.3, the performance of the RingNet architecture is compared with state-of-the-art NoCs [Mai15], [Pap15] for sample FPGAs. It turned out that the RingNet network utilizes less resources than the state-of-the-art NoCs. Moreover, the RingNet network can be clocked with substantially higher frequency than both state-of-the-art NoCs. The advantage of the RingNet architecture comes from the use of small and optimized 3-port switches. The 3-port switch efficiently utilizes LUTs with a restricted number of inputs (see Sections 2.2.3 and 5.1.3), and the applied ring topology lets the network spread over the whole FPGA (see Section 5.2.1). The prohibition of direct communication between processing elements (see Section 5.1.1) allows for the use of fixed-size network buffers that utilize highly-available distributed RAM (see Sections 5.1.2 and 5.2.3) and keep the resource cost of the RingNet below the cost of the state-of-the-art NoCs.

Chapter 9. Comparison between implemented RingNet ring and AXI4 Interconnect

The author compares RingNet with AXI4, a widely used communication infrastructure for FPGAs. Both have common features. According to [Xil15], AXI4 is designed for high-performance memory-mapped requirements, just like RingNet. Both use packets with a single address flit and separate write and read channels. In contrast to RingNet, AXI4 supports packets of different sizes up to 256 flits and various data widths from 32 up to 1024 bits. AXI4 implementation for Xilinx devices is provided in [Xil15].

AXI4 connects memory-mapped devices using AXI Interconnect. It is built of a crossbar, a data fifo used for buffering, pipeline flip-flops used to break a critical timing path, and an address range decoder (see Section 4.1 for more information). AXI Interconnect is available as a standalone IP in the Xilinx IP Catalog.

9.1. Methodology

Aiming at fair comparison, the author configured a RingNet ring and AXI4 Interconnect to have similar features. The configurations are described in Table IX.1.

TABLE IX.1

RINGNET RING AND AXI4 INTERCONNECT CONFIGURATION

Parameter	RingNet ring	AXI4 Interconnect
Address width	37 bits	37 bits
Data width	64 bits (single ring)	64 bits
Number and types of available interfaces	*N*×1: - 1 RI for connecting another ring or a memory device - *N* (2 – 15) of NIs for connecting PEs	*N*×1: - 1 slave interface for connecting memory device, - *N* (2 – 15) of master interfaces for connecting PEs
Arbiter	L2R Manager	Round-Robin arbiter
Buffering	LUTRAM-based fifos	
Performance optimization	---	Crossbar in performance optimized version named Shared-Address, Multiple-Data (SAMD), Use of pipelining FFs called AXI Register Slice
Data enable	Enable bit for each transmitted data byte.	

The RingNet architecture is compared with AXI4 Interconnection in terms of the resource utilizations and the maximum clock frequencies.

The implementation results are obtained just for a sample FPGA device, namely Artix7 from Xilinx (xc7a100tcsg324-1). The synthesis discussed in Chapter 8 already showed comparable results of RingNet for different types of FPGA. Therefore, the conclusions from the implementation can also be extended to the other FPGAs.

The synthesis results presented in Chapter 8 are obtained using the Synplify Premiere synthesizer. Synthesis results are estimations and minor differences between resource utilization and clock frequency of the synthesized design and the design implemented in an FPGA device are common. In order to check the actual resource utilization and the maximum frequency of a design, an implementation needs to be performed using EDA tools provided by the device vendor. Vivado Design Suite is a native set of EDA tools from Xilinx, therefore it is used to obtain implementation results for RingNet rings and AXI4 Interconnect for Artix 7 FPGA. AXI4 Interconnect was configured using the AXI Interconnect RTL 1.7 generator from Vivado Design Suite 2016.4. The implementations were performed using Vivado 2016.4.

The goal of the implementations is to find the maximum clock frequencies for the RingNet ring and AXI4 Interconnect. The implementation is performed by the Vivado packing algorithm. As discussed in [Luu16], a packing algorithm iteratively assigns parts of user-defined digital design to logic blocks of a given FPGA architecture. At each iteration the algorithm tries to satisfy user constraints, e.g., requested clock frequency of the design. The iterative nature of this algorithm gives suboptimal packing and suboptimal maximum clock frequency of the packed design. In consequence, it can happen that the increased frequency of the requested clock results in a lower maximum clock frequency of the packed design. Therefore, in order to find the actual maximum clock frequency for the RingNet ring and AXI4 Interconnect, multiple implementations were performed for the requested clock frequencies changed with the resolution of 1 MHz.

9.2. Implementation results

The results obtained for the maximum obtained frequencies are presented in Table IX.2.

TABLE IX.2

UTILIZED RESOURCES AND THE MAXIMUM FREQUENCY OF RINGNET RING AND AXI4 INTERCONNECT IMPLEMENTATIONS

		LUTs utilized		FFs utilized	Max freq. [MHz]
AXI4 Interconnect	2×1	1370	(22% as RAM)	2801	268
	4×1	2205	(23% as RAM)	4480	226
	6×1	3175	(22% as RAM)	6151	192
	15×1	7181	(23% as RAM)	13650	151
RingNet ring	2×1	1185	(44% as RAM)	2047	396
	4×1	1846	(43% as RAM)	3188	392
	6×1	2557	(44% as RAM)	4343	382
	15×1	5650	(43% as RAM)	9595	365

One can compare the results obtained for the implementation and synthesis of RingNet rings (see Table VIII.7 for Artix7 and Table IX.2). The differences in the number of utilized LUTs reported in both tables are minor. The maximum clock frequency is higher for the implementation by 12% on average. This increase is expected and is the result of optimizations enabled during the implementation.

9.2.1. Resource utilization

For all the cases RingNet ring requires less resources than AXI4 Interconnect. The FF utilization is about 40% lower for all configurations. The reduction in the number of used LUTs ranges from 16% (for the 2×1 configuration) to 27% (for the 15×1 configuration). Despite utilizing less LUTs, a RingNet ring utilizes more LUTs as RAM than a corresponding AXI4 Interconnect, providing more buffer space.

Both RingNet rings and AXI4 Interconnect utilize a part of the resources for core modules that are shared between a number of attached PEs. Those core modules for RingNet ring are: L2R Manager, Slot Generators, and Root Interface, whereas the core modules of AXI4 Interconnect are the Round-Robin arbiter and a slave interface. For both architectures the numbers of utilized LUTs and FFs follow the equation (4).

$$R_{Total} = p \cdot R_{PE} + R_{core} \quad (4),$$

where p is the number of connected PEs, R_{PE} is the number of LUTs or FFs utilized per PE and R_{core} is a constant number of LUTs or FFs utilized for the core modules. The parameters R_{PE} and R_{core} can be calculated for LUTs and FFs based on the results presented in Table VI by using linear regression. Results of the regression are presented in Table IX.3.

TABLE IX.3
RESOURCES UTILIZED FOR RINGNET RING AND AXI4 INTERCONNECT IMPLEMENTATIONS

	Resources added per PE (R_{PE})		Resources utilized for core modules (R_{core})	
	LUTs	FFs	LUTs	FFs
AXI4 Interconnect	449	834	473	1133
RingNet ring	344	581	497	884

For other state-of-the-art NoCs from [Mai15], [Pap15], one switch is added per processing element (PE), therefore the utilization of resources is proportional to the number of PEs. For interconnects like RingNet or AXI4, core modules are used, and their cost is shared between all the connected PEs. As already stated in the previous section, this approach can reduce the overall resource utilization of a network.

9.2.2. Maximum clock frequency

From Table IX.2, one can conclude that AXI4 Interconnect does not scale well and the maximum clock frequency decreases rapidly for a growing size of the AXI4 crossbar. On the other hand, RingNet, due to its optimized 3-port switches and ring topology, provides high maximum clock frequency across a wide range of network sizes. For the corresponding configurations, RingNet supports higher frequencies than AXI4 Interconnect. The increase in the maximum clock frequency is from 48% (for the 2×1 configuration) up to 142% (for the 15×1 configuration).

9.2.3. Summary

In Section 9.2, RingNet rings are compared with the state-of-the-art crossbar AXI4 Interconnect. Both interconnections are configured to have similar features, and the results of their implementation in Xilinx FPGA are presented. RingNet rings are demonstrated to use 40% fewer FFs and 16% to 27% fewer LUTs. Moreover, RingNet rings support clock frequencies from 48% to 142% higher than AXI4 Interconnect, depending on the number of interconnected processing elements. The reasons for the high performance of the RingNet architecture have already been discussed in Section 8.3.4 and the utilization of an optimized 3-port switch is pointed out as its main advantage. Long connections are one of the reasons of the poorer performance of AXI4 Interconnect [Mai15]. Moreover, the logic for the AXI4 Interconnect crossbar does not scale well. How well or poorly does an interconnection scale can be evaluated in terms of the degradation of the maximum clock frequency observed between the smallest and the biggest tested interconnection size. For RingNet rings, the maximum clock frequency degradation observed between a 2×1 configuration and a 15×1 configuration is 8% (from 396 MHz to 365 MHz), whereas 44% of clock frequency degradation is observed for AXI4 Interconnect (from 268 MHz to 151 MHz).

Chapter 10. Summary of the dissertation

In Section 1.3 two theses, T1 and T2, have been formulated. Both theses of the dissertation have been proven, which is summarized in Section 10.1. Further, in Section 10.2 the main achievements of the dissertation are gathered. Section 10.3 briefly discusses the application of the proposal. Finally, in Section 10.4 the possible direction for future research is presented.

10.1. Achievements related to theses

In both theses of the dissertation, the author states that it is possible to develop a network-on-chip featured with guaranteed throughput and network access fairness. In the dissertation, a novel RingNet network-on-chip is presented with the aim of validating the theses. RingNet architecture is described in Chapter 5. The description of the RingNet protocol and its modules are presented in Chapter 6. In Chapter 7, the property of guaranteed throughput is demonstrated through simulations. Particularly, the maximum achievable throughputs of logical channels are shown to match the value calculated according to the formula (1) given in Section 6.3.3.2. Moreover, the simulation results demonstrate network access fairness expressed as fair throughput allocation between interconnected processing elements and similar experienced latency.

Ad. T1) It is possible to develop a network-on-chip architecture and protocol featured with controlled throughput, network access fairness, and maximum clock frequency higher than 90% of the maximum clock frequency of FPGA hardware resources, across FPGAs of leading vendors.

The postulated clocking frequency for RingNet is demonstrated in Section 8.3. The maximum clock frequency for RingNet modules is compared with the maximum frequency reported for hardware DSP and BRAM blocks in highly pipelined, high performance configurations. For FPGA architectures of leading vendors, the maximum frequency that RingNet modules and rings can be clocked at, is equal to or higher than 96% of the frequency applicable to the DSP and BRAM blocks.

The reported high maximum clock frequency is a result of thorough utilization of resources available in all considered FPGA architectures. In particular, 3-port switches are used (see Section 5.1.3), which requires simple logic, suitable for high-frequency, LUT-based

implementations. Moreover, the applied prohibition of direct communication of processing elements lowers the memory requirements for network buffers (see Section 5.1.1), therefore small-capacity, highly-available, high-frequency distributed RAM is utilized for this purpose. Resource utilization for chosen FPGAs from different vendors is discussed in Chapter 8. In Section 8.2, the expected ratios of the resources utilized in FPGA from one vendor to the resources utilized in FPGA from another vendor are estimated. In Section 8.3, it is shown that the results from the syntheses of RingNet modules follow the expected ratios. Adherence to the estimated ratios clearly shows that no difference between FPGA architectures, which may be critical for NoC implementation, is overlooked.

Ad. T2) It is possible to develop a network-on-chip architecture and protocol with controlled throughput and network access fairness for FPGAs which would use less resources and would be featured with a higher maximum clock frequency than the state-of-the-art crossbar (AXI4 Interconnect).

RingNet rings are compared with AXI4 Interconnect in Chapter 9. It is shown that the RingNet ring requires less resources than AXI4 Interconnect. The FF utilization is about 40% lower and the reduction in the number of used LUTs is by 16% to 27% for various configurations. The reported clock frequencies are higher in favor of RingNet by at least 48%. Therefore, the RingNet architecture is shown to clearly outperform the state-of-the-art solution significantly.

10.2. Important original achievements of the dissertation

The primary original achievement of the dissertation is the proposal of an inter-FPGA compatible NoC architecture. For all tested FPGA devices, which represent FPGA architectures from leading vendors, the proposed RingNet NoC can work with frequencies as high or higher than 96% of the maximal clock frequency of FPGA hardware resources (Section 8.3.1). Moreover, for sample FPGA devices, maximal clock frequencies reported for RingNet are substantially higher than the frequencies reported for state-of-the-art FPGA-oriented NoCs (Section 8.3.3) and the widely-accepted crossbar (Section 9.2.2). The high clock frequency of RingNet goes hand in hand with its lower resource cost per interconnected processing element when compared with state-of-the-art FPGA-oriented NoCs (Section 8.3.3) and the widely-accepted crossbar (Section 9.2.1). The high clock frequency observed for RingNet network and

its efficient FPGA resource utilization observed across FPGA architectures of leading vendors makes RingNet an inter-FPGA compatible NoC. RingNet is the only NoC designed for FPGAs that is known to the author of the dissertation that reaches the demanding goal of inter-FPGA compatibility (Section 4.2.3).

The author proposed an original approach to the problem of determining the buffer size in NoC architectures. The author identified the *a priori* unknown traffic load as the main cause of the problem (Section 3.2) and postulated that traffic load in RingNet is controlled by a destination processing element (Section 5.1.1). Aiming at this objective, direct communication between processing elements is prohibited in RingNet NoC (Section 5.1.1) and processing elements exchange data through System Memory (e.g., external SDRAM) and control information is exchanged through a RingNet module called a Reflector (its idea and implementation are presented in Section 6.3.4 and Appendix IV, accordingly). This original approach to FPGA-based NoC limits congestions (discussed in Section 6.3.4), and therefore, lets the fixed-size network buffers be exploited. The applied indirect communication comes at the expense of increased latency when compared with state-of-the-art NoCs (see Section 7.2), though on the other hand, the high latency can be mitigated with higher maximum clock frequency reported for RingNet (see Section 8.3.3).

The virtual cut-through switching technique and LUTRAM-based buffers are adopted for RingNet, as this combination of switching technique and buffer type is known to be efficient for NoCs implemented in Xilinx FPGAs (Section 4.2.2). The original achievement of the author of the dissertation is that he demonstrated that virtual cut-through switching and LUTRAM-based buffers are efficient also in other FPGA architectures (synthesis results are presented in Chapter 8, and simulation results are presented in Chapter 7).

The author proposed an original tree-of-rings network topology suitable for FPGA (Section 5.2.1). The aim of the proposed hybrid topology is to combine the advantages of ring topology and tree topology into one efficient NoC topology for FPGA. The ring topology can result in low resource cost and high frequency for FPGA implementation, as it utilizes 3-port switches (discussed in Section 5.1.3). The combination of the ring topology with tree topology aims at reducing the latency of pure ring topology (simulation results are presented in Section 7.2).

The author proposed an original flow control mechanism for the RingNet architecture that provides throughput fairness and latency fairness for all connected processing elements,

provides priority support, and predictable maximum throughput described by a given formula (the formula is given in Section 6.3.3.2, simulation results are presented in Section 7.2). The flow control mechanism is described in Section 6.3.1.2 and the results of the fairness test are discussed in Section 7.3 and Appendix VII.

In order to make a reliable assessment of the achievement of the dissertation, the author has carried out a series of experiments described in Chapters 7 – 9. To this end, the author has implemented RingNet components in Verilog language: Leaf Interface and Root Interface, library of LUTRAM-based fifo buffers used in wide range of RingNet components, L2R Manager, Ring Adapter, memory interfaces for RingNet, Reflector, Slot Generator. The author adapted RingNet components in Verilog language for FPGAs from various producers. This required preparing alternative parts of code (especially defining memory buffers) according to specifications provided by the producers. The author estimates the total number of his own Verilog code lines at 8000. The experiments that assessed the achievement of the dissertation, especially simulations that the author has performed, consumed substantial amounts of computational costs. The results presented in the dissertation summarize over 3000 simulations. Additional simulations were performed with the aim of network tuning and debugging. More than 15000 simulations were performed that took the equivalent of about 500 days of continuous computations of a single-threaded computer. The preparation of such a wide range of experiments made up a substantial part of the dissertation.

Moreover, the author analyzed the common features and differences in logic block architectures across FPGAs from leading vendors (Chapter 2, Appendices I – III) and proposed an original set of ratios useful for estimating the utilization of LUTs and FFs for designs transferred between FPGAs from Lattice, Intel and Xilinx (Section 8.2).

The original achievements of the dissertation have been presented in the paper “RingNet: A Memory-Oriented Network-On-Chip Designed for FPGA,” published in Institute of Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integrated (VLSI) Systems in 2019 [Sia19].

10.3. Application of RingNet

SoCs are built out of modules, i.e., processing elements (PEs) and interconnects. An attractive aspect of SoC modularity is that the once defined modules can be used again in other

SoCs. Reuse of modules can save the time and cost of future developments and it justifies putting more effort into a one-off development that will be averaged over many SoCs. Those optimizations are essential for developers being under time-to-market pressure.

The interconnection utilized for SoC can impact the reusability of developed modules. In particular, the interconnect determines the interface of PEs, and it is desired to use a standardized interface in order to maximize module reusability. This is one reason why NoC with its defined interface is preferred to design-specific interconnections. RingNet is one of a number of NoCs developed for FPGAs ([Pap12], [She14], [Ret14], [Mai15], [Pap15], [Kap15], [Was17], [Kap17a], [Kap17b], [Kap17c], [Vip17], [Mai17], [Sid18], [Ahm18], [Red19]) and like others, it can be utilized in FPGA-based SoCs. However, like no other NoC, RingNet implementations are demonstrated to be efficient in terms of the maximum clock frequency and resource consumption for flagship FPGA devices from major manufacturers. Modules featured with the RingNet interface can potentially be reused for different FPGA architectures, as RingNet can be transferred between different FPGA architectures and keep its high performance.

RingNet is a network architecture offering quality tools for applications that are sensitive to traffic parameters. The simulation results demonstrate that RingNet features guaranteed throughput, predictable latency, traffic prioritization, and fair network access.

In the introduction to the dissertation (in Section 1.2), examples of multimedia processing considered suitable for FPGAs have been given, e.g., HEVC coding [Buk17], audio beamforming and the audio wave field synthesis [The11]. Due to the complexity of the multimedia processing algorithms, those are suitable to be implemented as SoCs. For this purpose, complex processing is divided into steps and those steps are implemented as processing elements, e.g., video coding can be divided into steps of input signal noise reduction, prediction, transform coding, deblocking filtering, entropy coding, etc. In [Stę13] one example of the AVC coder implementation following the idea is given. Moreover, multimedia processing SoCs in general require high throughput and substantial memory capacity which can be satisfied using SDRAM memory devices and RingNet designed to provide enough throughput to fully exploit the SDRAM. Therefore, the author believes that the RingNet NoC architecture and protocol may be widely adopted in FPGA-based SoC designs, especially in high-volume data processing applications like video processing and compression.

10.4. Future work

In complex SoCs, it may be desired to establish clock domains with individual clock frequencies, e.g., with the aim of optimizing power dissipation. The author believes that the RingNet architecture, and especially the RingNet ring can be used to isolate parts of a complex SoC, which can operate in separate clock domain. Processing elements with similar maximum clock frequency, or in general with common power management, can be grouped in a common ring. Nevertheless, the usage of rings in separate clock domains requires methods for inter-domain transitions. Considerations on such methods for RingNet would be an interesting direction for future research, in particular, when searching for techniques that would be efficient for several families of FPGA devices.

Appendices

The primary resource of an FPGA is an array of logic blocks. A logic block contains look-up tables (LUTs) and flip-flops (FFs). Different FPGA architectures use logic blocks with different numbers of FFs and LUTs, and various additional hardware resources that increase logic block capabilities. In Appendices I – III, logic block architectures from three leading FPGA vendors are presented: Lattice Semiconductor Corp., FPGA department of Intel (formerly Altera), and Xilinx Inc. The purpose of this review is to give insight into differences between the three architectures, which may affect the implementation of the inter-FPGA compatible NoC.

I. Lattice architecture description

The appendix discusses details of the logic block architecture applied in Lattice FPGAs (LatticeECP2, LatticeECP3, and ECP5). A logic block in the Lattice architecture is called a slice. Most slices in Lattice FPGAs contain two LUTs and two FFs, as depicted in Fig. A.I.1.

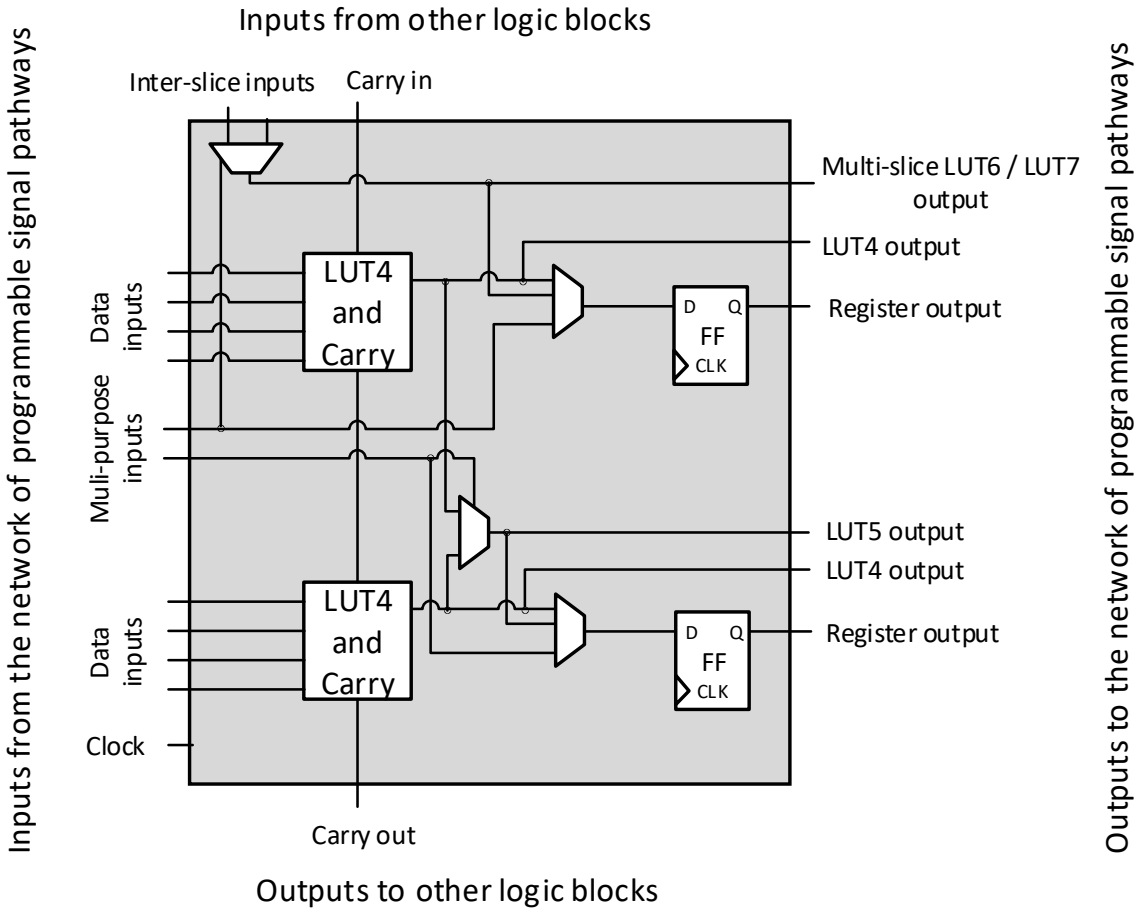


Fig. A.I.1. Block diagram of a slice, i.e., logic block from FPGAs by Lattice.

In LatticeECP2 and LatticeECP3 devices, some slices may have no FFs.

FFs can be used to store an output value from associated LUTs. On the other hand, LUTs and FFs can be used independently. For this purpose, a signal path skipping LUT and a signal path skipping FF are present in the Lattice slice architecture. A similar feature can be identified in all considered FPGAs from other vendors. For Intel FPGAs, independent usage of the associated LUT and FF is called register packing.

Considering FPGAs produced by Intel, Xilinx, and Lattice, the last one offers the simplest logic block architecture utilizing LUTs with 4 inputs (LUT4).

Thanks to additional multiplexers available in the Lattice slice and inter-slice paths, a function with more than 4 inputs can be constructed by utilizing a higher number of LUT4s. The inter-slice path connects slices that are grouped into structures called Programmable

Functional Units (PFUs) [Lat13]–[Lat16]. A single PFU can be configured to provide up to LUT7 functionality. The architecture of PFU differs slightly between LatticeECP2, LatticeECP3, and ECP5 devices. A simplified block diagram of PFU for LatticeECP2 and LatticeECP3 is presented in Fig. A.I.2

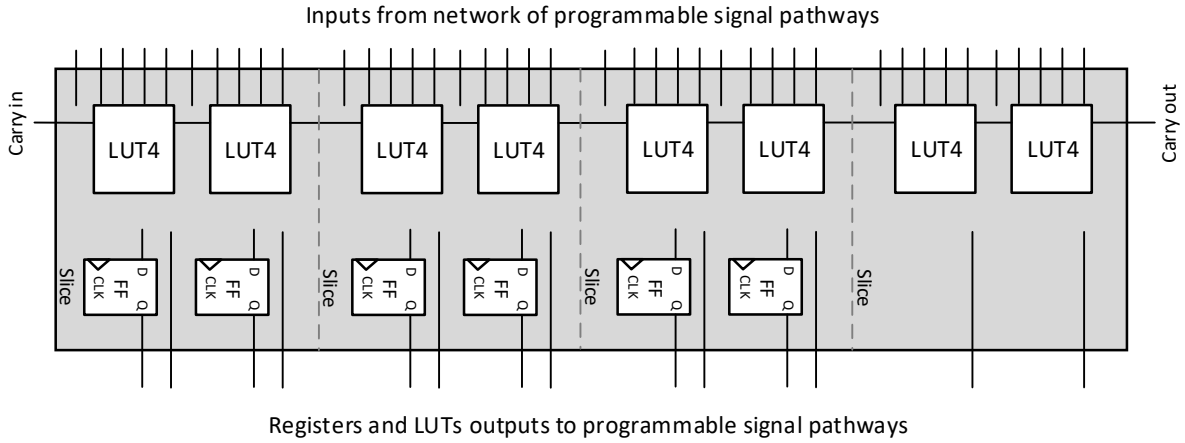


Fig. A.I.2. Block diagram of a Programmable Functional Unit (PFU) from LatticeECP2 and LatticeECP3 FPGAs by Lattice.

PFU is connected to programmable signal pathways, i.e., a configurable communication backbone of every FPGA device (cf. Chapter 2). PFU contains four logic blocks (slices). In LatticeECP2 and LatticeECP3 devices, the last slice has no FFs. In ECP5 devices, FFs are also present in the last slice. In LatticeECP2, LatticeECP3, and ECP5, all slices in all PFUs can be configured to realize logic functions or ROM.

PFU can be configured to realize a distributed RAM (LUTRAM) function. In ECP5 all PFUs are LUTRAM-capable, whereas in LatticeECP2 and LatticeECP3 just a part of PFUs can realize LUTRAM.

The LUTRAM configuration of PFU utilizes 3 out of 4 slices in the PFU to build a 4b×16-word deep memory block with separate read and write ports (pseudo dual-port RAM configuration) or with a single read and write port (single-port RAM configurations). In those configurations, one slice is used as an input for memory address and control signals, whereas LUTs of the remaining two slices are used as memory blocks, each providing a 1b×16 space. Bigger blocks of LUTRAM can be constructed utilizing a higher number of PFUs.

II. Intel ALM description

A logic block in FPGAs manufactured by Intel is called an Adaptive Logic Module (ALM) [Alt11]–[Alt16], [Int16], [Int17]. In all considered Intel devices, ten ALMs constitute a bigger structure called a Logic Array Block (LAB).

The architecture of ALM is similar for all FPGA families offered by Intel. A few differences between the families lie in technology process, DSP functionality and additional registers inserted on programmable signal paths (HyperFlex registers added in Stratix 10 devices). Nevertheless, those functionalities are not essential for RingNet implementation. A block diagram of the ALM architecture is presented in Fig. A.II.1.

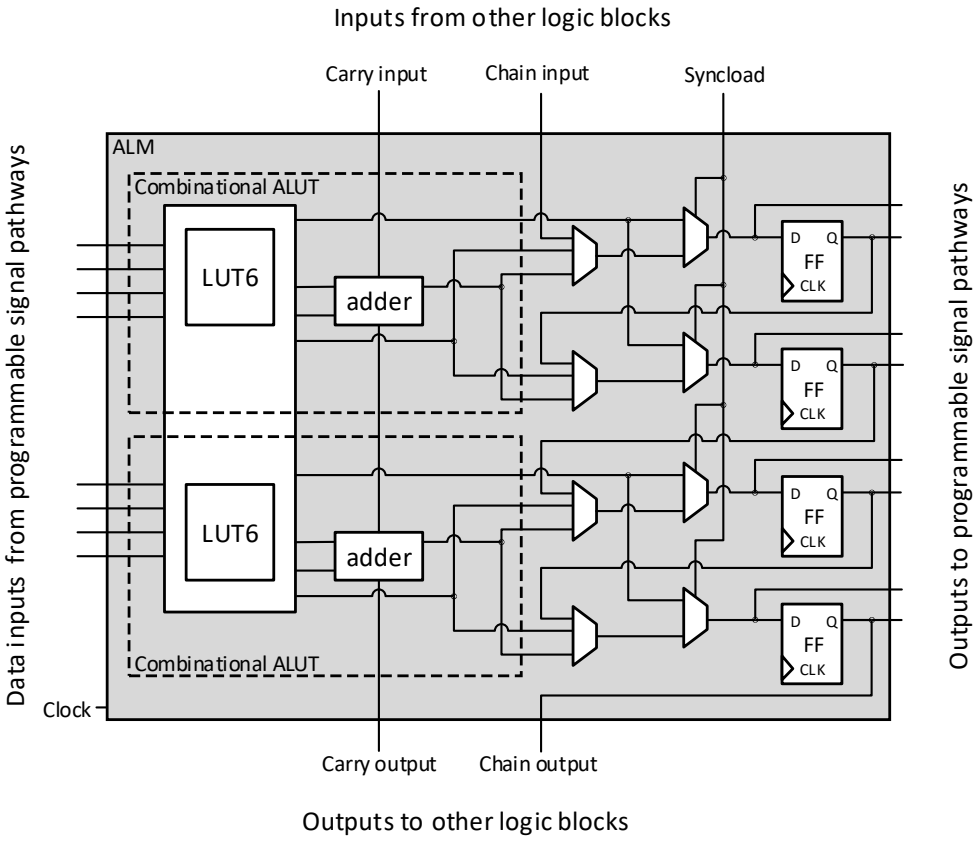


Fig. A.II.1. Block diagram of an Adaptive Logic Module from FPGAs by Intel.

An ALM is composed of two combinational adaptive LUTs (ALUTs) and 4 FFs. An ALUT is defined as a LUT and an associated two-bit adder.

ALM, as a logic function generator can realize:

- 1 × 7-input function (only for a subset of possible 7-input functions),
- 1 × 6-input function (for all possible functions),
- 2 × 6-input function (with at least 4 inputs shared),
- 2 × 5-input functions (with at least 2 inputs shared),
- 2 × 4-input functions,
- Other combinations of two functions, each having 6 inputs at most and with a total number of 8 inputs at most.

Intel describes its LUTs as LUT6 [Alt11], [Alt15a], [Alt15b], [Alt16], [Int16], [Int17]. In the actual ALM architecture LUT6s are not present. Each LUT6 is realized as one LUT4 and two LUT3s connected with multiplexers. This combination can realize a subset of functions possible to be implemented in a true LUT6. The size of a true LUT6, in terms of the number of bits stored in its table equals 64, whereas the total number of bits stored in one LUT4 and two LUT3s equals 32 bits. Nevertheless, for the sake of simplicity in the dissertation the author describes LUTs in Intel FPGAs as LUT6 in the same way, as it is presented in Intel documents [Alt11]–[Alt16], [Int16], [Int17].

ALM output can be driven by a LUT, a two-bit adder, or an FF. If the output from a LUT or two-bit adder is not stored in associated FF then the FF can be used to store data unrelated to the data processed in the LUT. This feature can improve resource utilization and is called register packing. This feature is also available in FPGAs by Lattice and Xilinx, as discussed in Appendices I and III.

From the RingNet perspective, an important feature of ALM is related to the presence of 2×1 multiplexers at the input of each FF (the second column of multiplexers depicted in Fig. A.II.1). This multiplexer is controlled by an external signal called a *syncload*. This construction increases the number of possible functions performed in ALM, especially a 2×1 signal multiplexer can be implemented without utilizing LUTs. 2×1 multiplexing is a common operation in RingNet, therefore the discussed multiplexer in the ALM architecture can potentially preserve a lot of LUTs.

From 24% to 50% of all ALMs in FPGAs manufactured by Intel can be configured as LUTRAM. Each ALM contains 64 bits of memory in its two LUTs. For Intel FPGAs it is not

possible to use a single ALM as LUTRAM, rather a whole LAB (10 × ALM) needs to be configured together as one memory block. As a result, the smallest LUTRAM block contains 640 bits of memory. It can be used as 20b×32-word deep memory or 10b×64-word deep memory with separate read and write ports (pseudo dual-port RAM configuration) or with a single read and write port (single-port RAM configurations).

III. Xilinx CLB description

A block diagram of a logic block available in Xilinx Spartan6 and series 7 FPGAs is presented in Fig. A.III.1.

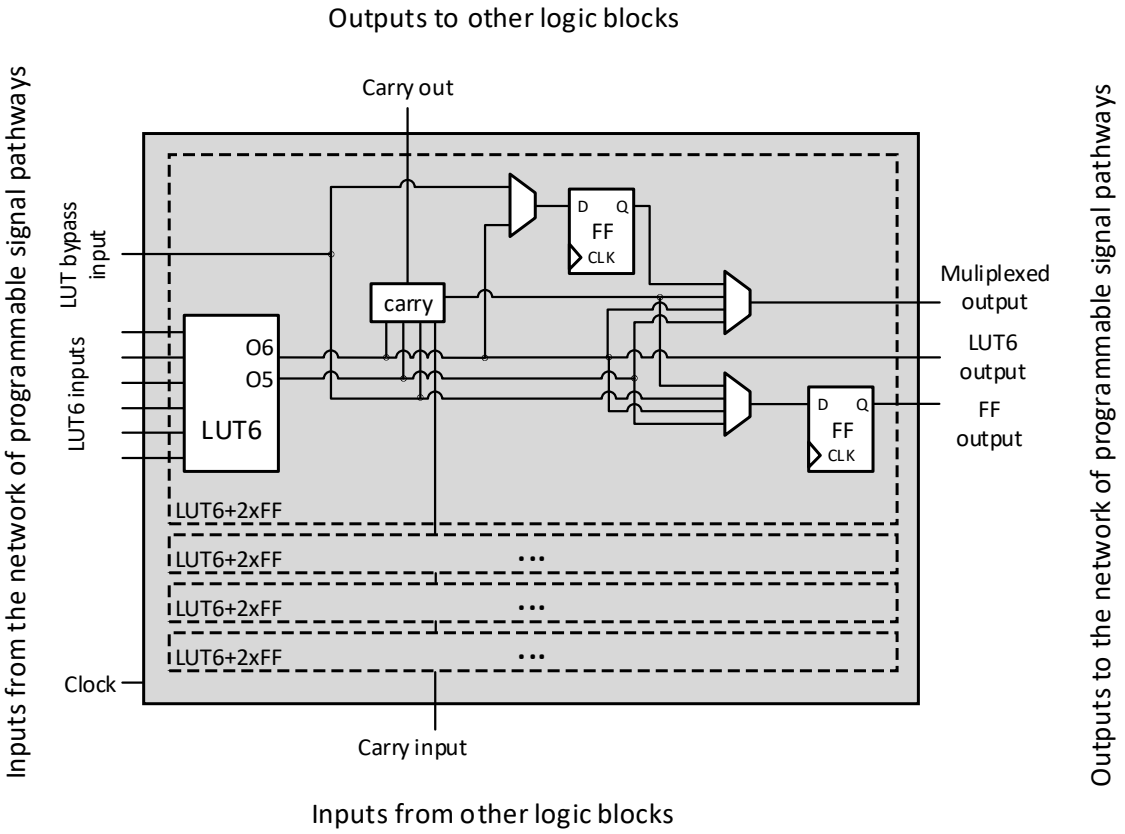


Fig. A.III.1. Block diagram of a slice from Spartan6 and series 7 FPGAs manufactured by Xilinx.

A logic block in FPGAs manufactured by Xilinx is called a slice. The slice consists of four smaller blocks, each containing one LUT6 and two FFs. In Spartan6, a simplified version of the slice is also available, called SLICEX which does not have multiplexers depicted in Fig.

A.III.1 at the output of a slice. In the UltraScale and UltraScale+ series the slice is twice the size and has eight LUT6s and sixteen FFs.

Slices in Xilinx devices are grouped in Configurable Logic Blocks (CLBs). A CLB available in the Spartan6 and 7 series contains two slices, whereas a CLB available in UltraScale and UltraScale+ contains only one slice but twice as big.

The LUTs in Xilinx FPGAs have 6 inputs (LUT6s). There are two independent outputs (O5 and O6) from each LUT. Thanks to the two independent outputs, a single LUT6 can generate two functions at once, improving the utilization of an FPGA. A single LUT6 can generate:

- 1 × arbitrarily defined 6-input combinational function,
- 2 × arbitrarily defined 5-input combinational functions, as long as these two functions share common inputs,
- 2 × arbitrarily defined combinational functions of 3 and 2 inputs or less.

The slice has inputs that bypass LUT6s. Moreover, an output from the slice is available that bypasses FFs. Those bypasses make it possible to use both LUT6 and FF for unrelated data. A similar mechanism is known from Intel and Lattice FPGAs and it is called register packing.

From 14% up to 50% of the slices in Xilinx FPGAs are LUTRAM-capable SLICEMs. A LUT6 in the SLICEM has additional inputs for address and data (not depicted in Fig. A.III.1). Using a combination of LUT6s, a wild range of memories can be constructed in single, dual and quad-port configurations. RingNet modules exploit memories with separate read and write ports (pseudo dual-port RAM configuration). The smallest available blocks of pseudo dual-port LUTRAM are:

- 6b×32-word deep memory utilizing four LUT6,
- 3b×64-word deep memory utilizing four LUT6.

A bigger block of RAM can be constructed utilizing more LUT6s.

IV. Example of Reflector implementation design

As stated in Chapter 5, RingNet is dedicated for memory-oriented SoCs, i.e., systems where most of the traffic starts or ends in the memory. Therefore, the performance of RingNet is assessed in terms of the performance of data transport between PEs and System Memory (throughput, latency, and other reliability parameters are presented in Chapter 7). Moreover, in Chapter 8, synthesis results are presented for all the RingNet modules that create paths between PEs and System Memory, i.e., ring adapters and modules that create RingNet rings (L2R Manager, Slot Generator, Leaf Interface, and Root Interface). The Reflector is not on the path of data transported between PEs and System Memory. The Reflector is a control information buffer and, therefore, only control packets (events and event confirmations) reach this module. Nevertheless, the Reflector is required for the management of RingNet-interconnected SoC (see Section 5.1.1) and it is important to demonstrate that the Reflector module with functions postulated in Section 6.3.4 can be implemented. Therefore, in Appendix IV, an example of a Reflector architecture design is discussed and its synthesis results for FPGA architectures considered in the dissertation are presented. Obviously, the author's design is not the only one possible.

IV.1. Reflector architecture

The Reflector mediates in the communication between processing elements (PEs). A PE can communicate with another PE by sending an *event* message through the Reflector. The events are buffered in the Reflector and sent to the destination PE, one at a time. After receiving the event, the destination PE sends the *event confirmation* to the Reflector.

Fig. A.IV.1 depicts a block diagram of an example of Reflector implementation for a network of up to 256 PEs.

The Reflector contains three large buffers:

1) Bank of first-in-first-out buffers (fifos) for events, each dedicated for a single PE,
2) Bank of state registers for processing elements (PEs),
3) General event buffer.

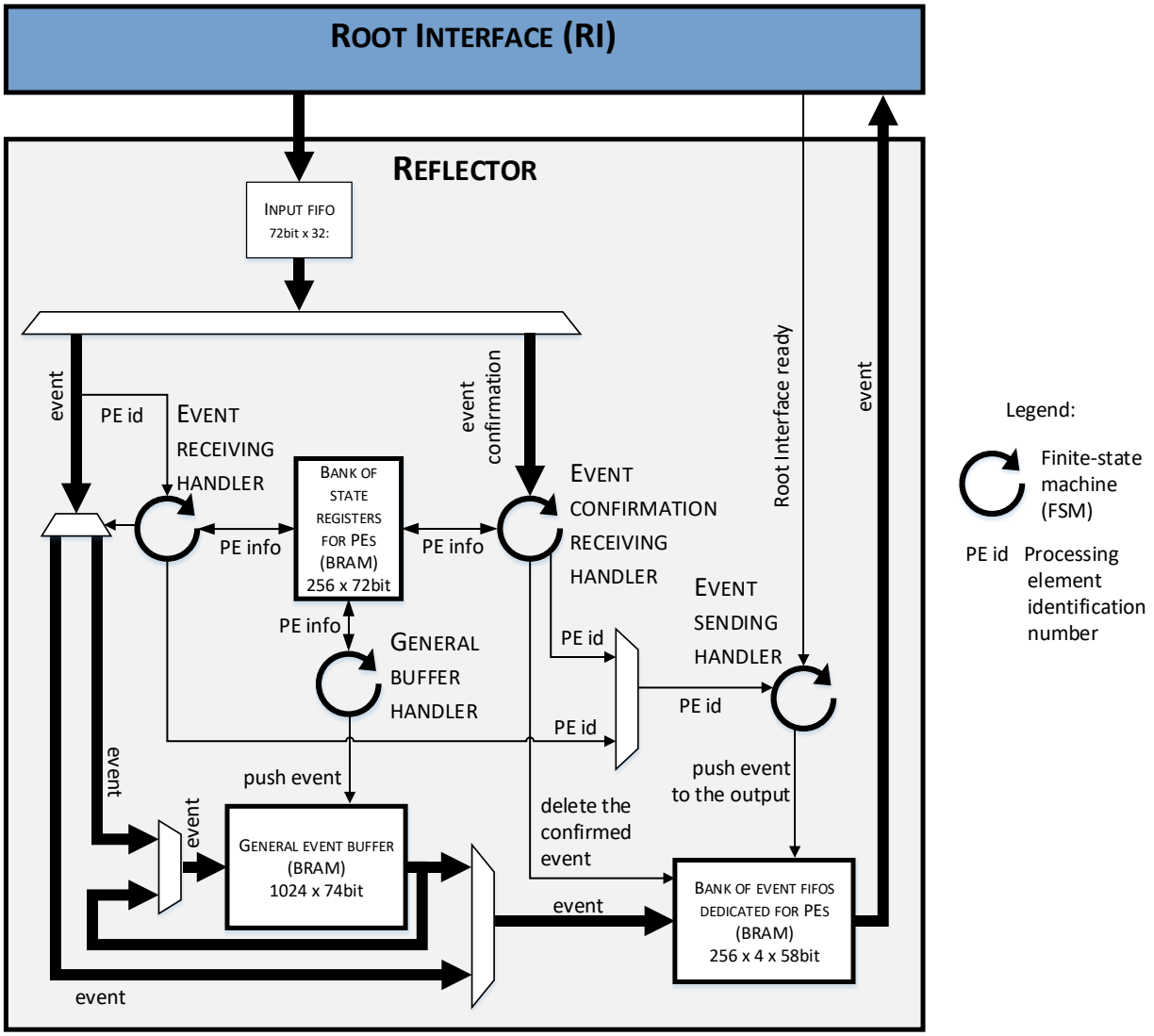


Fig. A.IV.1. Block diagram of the Reflector.

Individual fifos, used to separate events for each PE, are gathered into the bank of event fifos. Each fifo can store 4 events. The total capacity of the bank for 256 PEs equals 58kb. The memory with substantial capacity is efficiently implemented with the use of block RAM (BRAM). Utilizing LUTRAM or FF for the memory can result in low clock frequency and a high number of utilized LUTs. The drawback of BRAM utilization is that it provides a common port for all the fifos, so it limits the access to just one fifo at a time. Nevertheless, events and event confirmations reach the Reflector one at a time and there is no need for parallel access to event fifos.

The bank of state registers for PEs stores information about each PE, i.e., its network address, the number of events waiting in the Reflector, etc. For the 256 PEs it utilizes 18kb of memory. The bank of state registers for PEs is implemented using BRAM, just like the bank of event fifos.

The general event buffer is introduced for events that do not fit into the event fifos due to the fifos fullness. When a slot in the appropriate event fifo is released, an event from the general event buffer is transferred into the event fifo. The proposed general buffer for 1024 events utilizes 74kb of BRAM memory.

Up to 1028 events dedicated for a single PE can be buffered in the Reflector: 4 slots are reserved for each PE in its event fifo, and 1024 slots are shared between all PEs in the general event buffer. The general event buffer provides flexible allocation of buffer space between PEs realized according to their needs, whereas an event fifo dedicated for a single PE guarantees minimal allocation for each PE.

The buffers in the Reflector are controlled by four finite-state machines (FSMs). Those FSMs are:

1) Event receiving handler,
2) Event confirmation receiving handler,
3) General buffer handler,
4) Event sending handler.

Algorithms realized by the FSMs are described using flowcharts depicted in Figures A.IV.2 – A.IV.5.

The event receiving handler executes the algorithm depicted in Fig. A.IV.2 each time a new event is received.

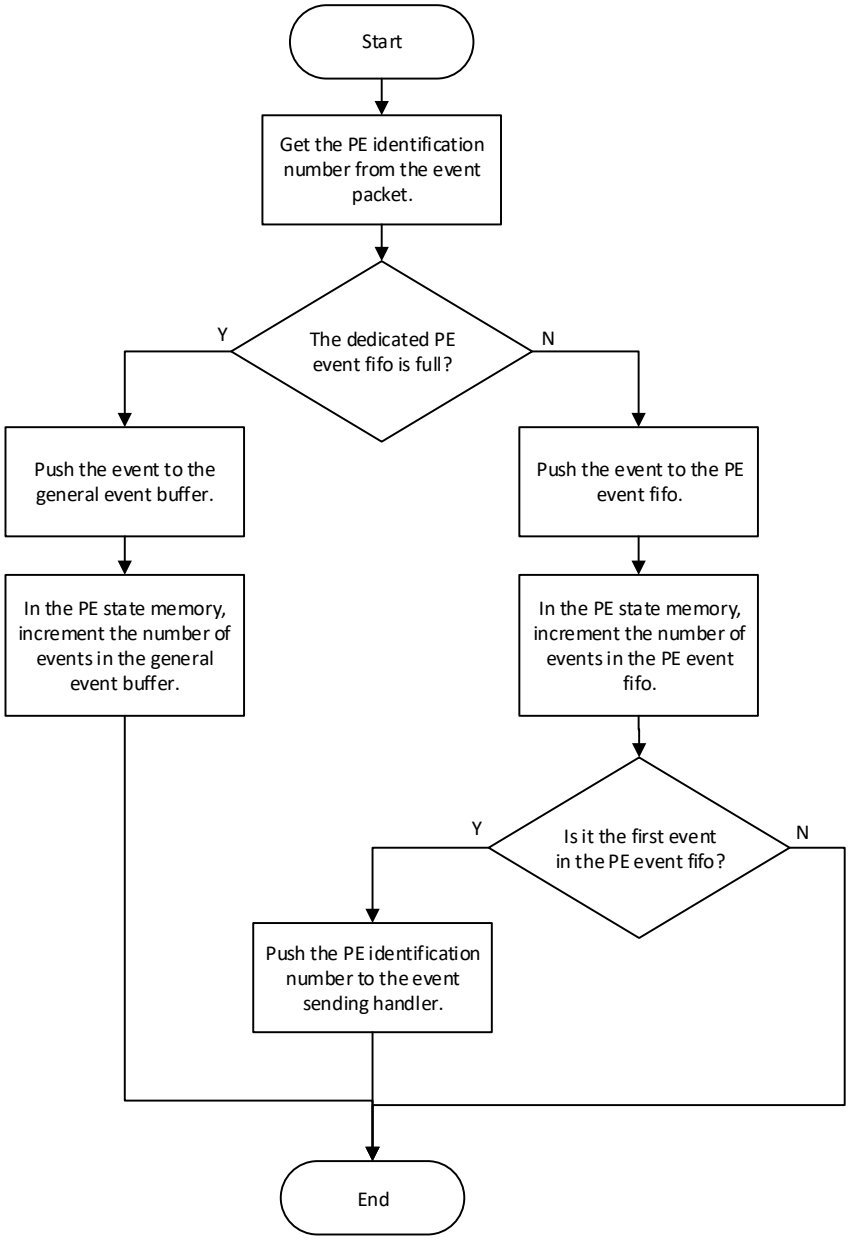


Fig. A.IV.2. Flowchart of an event receiving handler.

The handler reads the ID of the destination PE from the received event packet. Next, from the bank of states, the handler determines the state of the event fifo assigned to the destination PE. If the fifo is not full, the handler pushes the event there, otherwise, the event is pushed to the general event buffer. Pushing the event is recorded in the bank of PEs states. If there is just one event in the event fifo, then the event sending handler is triggered and the event sending process is started.

The event sending handler executes the algorithm depicted in Fig. A.IV.3.

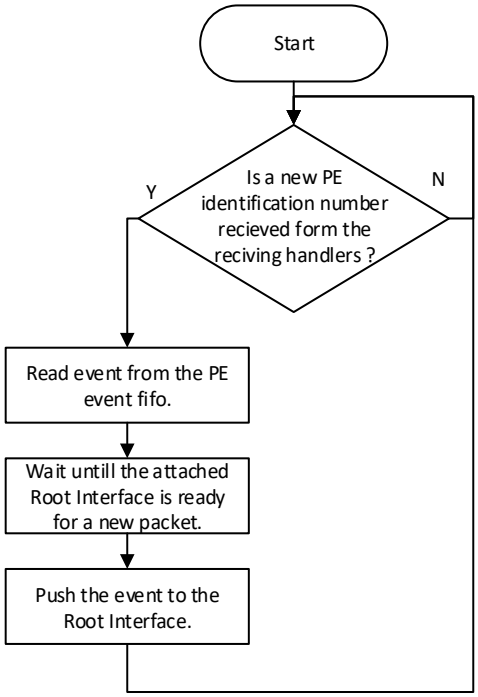


Fig. A.IV.3. Flowchart of an event sending handler.

The handler waits for an ID of a PE that can come from two receiving handlers: the event receiving handler or the event confirmation receiving handler. According to the received ID an event is read from a proper event fifo. The event is send through an attached Root Interface when the interface is ready to accept a new packet. Nevertheless, the event is not deleted from the event fifo until an event confirmation reaches the Reflector.

The algorithm executed by the event confirmation receiving handler is depicted in Fig. A.IV.4. The handler deletes the confirmed event from the event fifo, and updates information in the PE state memory accordingly. In the case of an event existing in the event fifo, the handler triggers the event sending handler to process it.

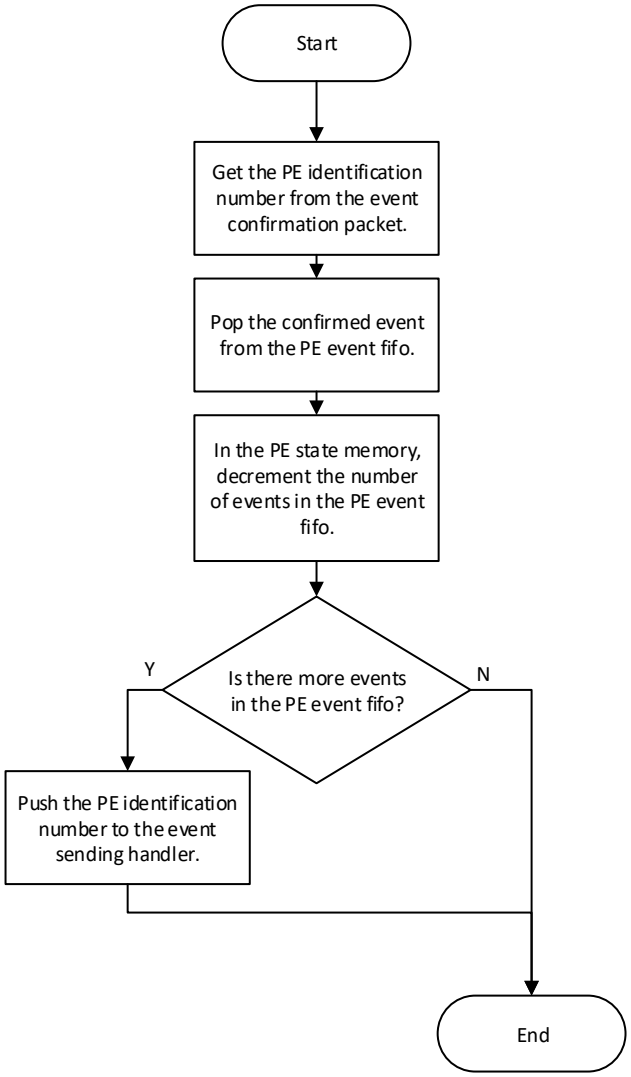


Fig. A.IV.4. Flowchart of a new event confirmation handler.

The general buffer handler executes the algorithm depicted in Fig. A.IV.5. The handler deals with the events that were once rejected from the bank of event fifos due to fifo fullness. Those events, stored in the general event buffer, are constantly searched by the handler and the state of their corresponding fifos is checked. An event is transferred to a corresponding fifo if an empty space is available.

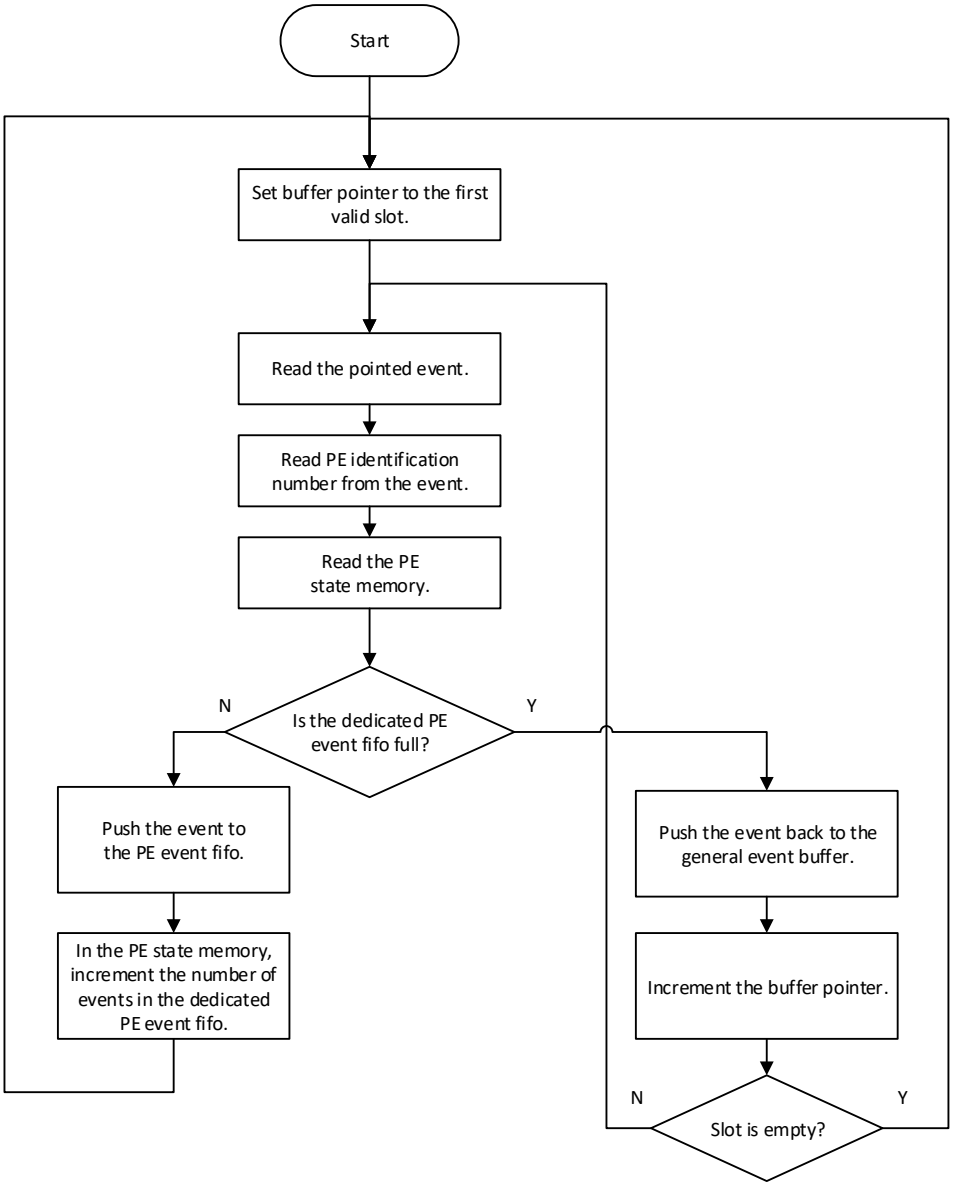


Fig. A.IV.5. Flowchart of a general buffer handler.

For the sake of brevity, only a minimal set of services provided by the Reflector are discussed, i.e., events buffering, sending, and confirming. Nevertheless, the Reflector is a device that contains full information about the system, and it provides other services. Those services include informing a dedicated PE about malfunctions detected in the system or about the risk of general event buffer overflow. The Reflector also informs about new PEs connected to the network, e.g., as a result of a dynamic system reconfiguration.

IV.2 Synthesis results

The author implemented the presented Reflector architecture in Verilog HDL language. The proper work of the proposed Reflector architecture has been verified in simulations. The prepared definition is synthesized according to the methodology presented in Chapter 8. As the presented Reflector architecture aims at efficient BRAM utilization, the number of used BRAMs is included in the synthesis report. Resource utilization and maximum clock frequency for Reflector synthesis is presented in Table A.IV.1

TABLE A.IV.1
RESOURCE UTILIZATION AND MAXIMUM CLOCK FREQUENCY FOR SAMPLE REFLECTOR IMPLEMENTATION.

		Xilinx			Intel		Lattice
		Artix7 xc7a100tcsg324-1	KintexUS xcku060-ffva1156-1-i	Virtex7 xc7vx550tffg1158-1	Stratix V 5SGXMABK2H40C3	Arria V 5AGXBA7D6F31C6	ECP5 lfe5u_85f-8
Utilized resources	LUTs	574	585	577	651	665	802
	FFs	1070	1144	1070	1294	1297	1152
	BRAM	6 × RAMB36 (36kb)	6 × RAMB36 (36kb)	6 × RAMB36 (36kb)	10 × M20K (20kb)	17 × M10K (10kb)	11 × EBR (18kb)
Maximum clock frequency	In MHz	275	303	313	234	183	154
	In relation to hardware blocks frequency	81%	66%	78%	59%	64%	83%

The number of LUTs and FFs utilized for the Reflector and a small RingNet ring in 2×1 configuration can be compared. The ring utilizes about 2800 FFs and from 1200 to 2700 LUTs, depending on the FPGA device used (cf. Table VIII.7 from Section 8.3). The proposed implementation of the Reflector utilizes less than half of those resources. The presented results demonstrate that the cost of the Reflector in terms of utilized LUTs and FFs is comparable to connecting one additional processing element to the RingNet network. BRAMs are the resources which are not utilized for any RingNet module other than the Reflector. BRAM in the Xilinx architecture is denoted as RAMB36 and has the capacity of 36 kb. Six RAMB36 are utilized for all tested FPGAs from Xilinx. In the Intel architecture blocks denoted as M10K and M20K are available with the capacity of 10kb and 20kb. Ten M20Ks are utilized for the Reflector in Stratix V devices, and seventeen M10K for Arria V. In the Lattice device BRAM is denoted as Embedded Block RAM (EBR). Each EBR has the capacity of 18kb and eleven EBRs are utilized for the Reflector. The RAM capacity required for the presented Reflector

architecture is equal to 150kb. The actual capacity of utilized BRAMs is equal to 216kb, 200kb, 170kb, and 198kb in devices from Xilinx, two devices from Intel, and the device from Lattice, respectively. The reason for the wasted capacity is the mismatch between the width of the available BRAMs and the widths required for Reflector buffers. Nevertheless, successful utilization of BRAMs for the implementation of the required 150kb capacity of fifo buffers is the achievement of the presented Reflector architecture. The alternative, i.e., implementation of buffers with that capacity in LUTRAM, can degrade the frequency characteristic of the design and increase the usage of LUTs multiple times (3125 and 14000 LUTs are required in Xilinx and Lattice architectures for this purpose, respectively).

In Table A.IV.1, the maximum clock frequencies reported for the Reflector are compared with the maximum frequency applicable to the hardware block available in FPGA devices. The hardware blocks of DSP and BRAM are used for this comparison according to the methodology discussed in Section 8.1.3. The methodology is used in Section 8.3.1 also for other RingNet modules. The maximum clock frequencies reported for the proposed Reflector architecture are in the range of 59% – 83% of the reference clock frequencies. For comparison, other RingNet modules and rings are featured with the maximum clock frequency as high as 96% of the reference clock frequencies or higher. On the one hand, the presented frequency results demonstrate that the provided Reflector implementation is not at the same level of optimization as other RingNet modules. On the other hand, the prepared Verilog source code for the Reflector is not optimized and the author expects that higher frequencies can be achieved. Nevertheless, the proposed implementation of the Reflector should be used in a separate clock domain so it does not limit the maximum clock frequency of the whole RingNet network and does not limit the network throughput.

IV.3 Summary

The aim of the appendix is to present details of an example of Reflector architecture and to demonstrate that the module can be realized in FPGA architectures considered in the dissertation. Both those goals have been obtained. Especially the utilization of BRAMs for the purpose of implementation of 256 fifo buffers in all tested FPGA architectures is a success of the presented design. Thanks to the BRAM utilization, the number of LUTs and FFs used for the Reflector is as low as the number of LUTs and FFs required to connect one processing element to the RingNet network.

V. Simulation results of the average latency test

In Appendix V, the average latency for the read and write channels is presented for various loads and network sizes. The results illustrate the conclusions of Section 7.2. The parameters of the test are:

- *R*: Multiplication degree of the root level, i.e., the number of parallel rings used at the root level, set in the range of 1–4.
- *F*: The number of 1st level rings, set in the range of 1–5.
- *G*: The number of packet generators (PGs) connected to a single 1st level ring, chosen from the set {1, 2, 3, 4, 7, 15}.
- Logical channel load. The aggregated load generated by all PGs is set in the range of 0%–100% of the theoretical throughput T_{RW_MAX} (1). The logical channel load is separately set for the read and write channels.

The results for the read channel are presented in Tables A.V.1 – A.V.20. The results are grouped according to the logical channel loads:

- Tables A.V.1 – A.V.4 correspond to loads equal 100% (network in saturation),
- Tables A.V.5 – A.V.8 correspond to loads in the range of 92% – 97% (network near saturation),
- Tables A.V.9 – A.V.12 correspond to loads in the range of 85% – 92% (high load),
- Tables A.V.13 – A.V.16 correspond to loads in the range of 69%–73% (moderate load),
- Tables A.V.17 – A.V.20 correspond to loads in the range of 27% – 28% (low load).

In each group of four tables, the results for increased numbers of parallel rings used at the root level are presented. The results for the write channel are presented in Tables A.V.21 – A.V.40:

- Tables A.V.21 – A.V.24 correspond to loads equal 100% (network in saturation),
- Tables A.V.25 – A.V.28 correspond to loads in the range of 92% – 97% (network near saturation),
- Tables A.V.29 – A.V.32 correspond to loads in the range of 85% – 92% (high load),

- Tables A.V.33 – A.V.36 correspond to loads in the range of 69%–73% (moderate load),
- Tables A.V.37 – A.V.40 correspond to loads in the range of 27% – 28% (low load).

Latencies observed for the read logical channel and write logical channel can be compared for the same network size and for the same load. On average, the latency observed in the write logical channel is 7 clock cycles longer than the latency observed in the read logical channel. In the RingNet network, parallel rings are used aiming at increasing the network throughput (see Section 5.2.2). Nevertheless, the increased number of parallel rings introduces additional registers at network paths and increases the observed latency. For a network below saturation (load below 100%), increasing the number of parallel rings at the root level by one results in latency increased by 6 clock cycles, on average. For a saturated network, all the network buffers are occupied and the latency is correlated with the capacity of network buffers, i.e., with the number of packets kept in the network buffers that wait to access the same path. As the second ring at the root level doubles the number of available paths without doubling the total buffer capacity in the network, reduced latency can be observed (compare Table A.V.1 with Table A.V.2, and Table A.V.21 with Table A.V.22). A further reduction in latency for a network in saturation can be observed when 3 and 4 parallel rings are used at the root level (Tables A.V.3, A.V.4, A.V.23, A.V.24).

TABLE A.V.1
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	88	166	209	252	394	764
	2	210	297	390	478	749	1528
	3	318	459	594	729	1127	2300
	4	417	602	780	958	1516	3058
	5	514	750	980	1193	1877	3672

TABLE A.V.2
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	124	193	236	280	421	791
	3	272	341	407	476	667	1257
	4	353	449	541	632	892	1672
	5	436	550	666	780	1108	2002

TABLE A.V.3
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	152	220	264	308	449	819
	4	311	365	432	494	674	1188
	5	374	452	533	610	830	1441

TABLE A.V.4
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	156	223	272	316	454	824
	5	349	401	459	515	689	1157

TABLE A.V.5
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	95	118	120	122	147	194
	2	113	134	142	145	182	225
	3	129	148	155	161	185	241
	4	130	151	157	163	189	245
	5	140	163	169	175	201	258

TABLE A.V.6
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	132	144	146	154	179	226
	3	139	156	158	163	186	235
	4	141	158	163	167	190	238
	5	155	170	175	180	202	253

TABLE A.V.7
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	152	169	171	177	202	249
	4	149	165	170	174	196	244
	5	159	174	178	186	207	254

TABLE A.V.8
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	155	170	176	181	204	254
	5	162	179	184	188	211	259

TABLE A.V.9

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–92%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	94	118	119	120	144	189
	2	113	132	139	141	169	214
	3	127	145	151	157	178	230
	4	128	147	153	157	179	231
	5	138	158	164	168	192	242

TABLE A.V.10

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE CHANNEL LOAD IS IN THE RANGE 85%–91%, READ CHANNEL LOAD IS IN THE RANGE 85%–92%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	130	143	145	151	175	221
	3	138	155	156	161	183	229
	4	140	156	160	164	186	232
	5	153	168	172	177	197	244

TABLE A.V.11

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–91%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	150	167	169	174	197	243
	4	147	163	168	171	193	239
	5	158	173	177	182	203	249

TABLE A.V.12

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–91%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	152	169	174	177	199	247
	5	162	176	181	186	208	254

TABLE A.V.13

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	93	116	117	118	140	183
	2	111	130	136	137	160	203
	3	126	142	146	151	170	216
	4	125	143	147	150	170	214
	5	136	153	158	160	181	224

TABLE A.V.14

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	129	141	142	148	170	213
	3	137	153	153	158	178	221
	4	138	153	156	160	180	223
	5	150	165	168	172	190	235

TABLE A.V.15

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	147	163	164	169	190	235
	4	145	160	164	168	188	232
	5	156	170	174	178	197	241

TABLE A.V.16

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	149	164	169	172	193	238
	5	158	173	178	182	202	246

TABLE A.V.17

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 27%–28%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	93	115	115	115	136	176
	2	110	127	132	132	152	192
	3	123	138	141	144	162	205
	4	122	138	141	143	161	204
	5	132	147	151	153	171	213

TABLE A.V.18
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 27%–28%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	127	138	138	143	162	204
	3	134	149	149	152	171	213
	4	136	149	152	154	173	215
	5	147	160	162	166	184	227

TABLE A.V.19
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE CHANNEL LOAD IS IN THE RANGE 27%–28%, READ CHANNEL LOAD EQUALS 27%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	142	157	158	162	181	223
	4	141	155	159	161	180	222
	5	152	165	168	172	190	233

TABLE A.V.20
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR READ CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 27%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	144	159	162	165	184	226
	5	154	168	171	175	193	236

TABLE A.V.21
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	95	172	216	260	401	771
	2	217	304	398	485	756	1535
	3	325	468	601	736	1134	2307
	4	425	611	790	969	1523	3064
	5	521	762	978	1201	1879	3712

TABLE A.V.22
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	132	200	238	287	428	798
	3	296	365	430	500	691	1281
	4	379	478	571	661	916	1700
	5	469	589	699	812	1153	2037

TABLE A.V.23
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	163	224	275	318	456	830
	4	334	393	454	515	696	1210
	5	402	476	558	633	856	1466

TABLE A.V.24
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 100%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	168	230	279	323	462	834
	5	368	423	481	538	708	1175

TABLE A.V.25
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	100	125	127	129	154	202
	2	120	141	149	152	187	232
	3	135	155	162	167	193	249
	4	136	158	163	170	196	252
	5	147	169	177	183	210	266

TABLE A.V.26
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	140	152	154	162	187	234
	3	146	163	166	170	194	242
	4	148	165	170	174	198	246
	5	162	176	182	187	209	261

TABLE A.V.27
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	159	177	181	187	211	260
	4	157	173	178	183	205	254
	5	167	183	187	194	215	263

TABLE A.V.28
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 92%–97%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	162	179	185	191	213	264
	5	170	185	191	195	218	267

TABLE A.V.29
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–92%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	99	124	126	127	151	196
	2	119	139	146	148	175	222
	3	134	152	158	164	185	237
	4	134	154	159	164	186	237
	5	145	164	172	176	200	249

TABLE A.V.30

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE CHANNEL LOAD IS IN THE RANGE 85%–91%, READ CHANNEL LOAD IS IN THE RANGE 85%–92%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	138	151	152	159	182	228
	3	145	162	163	168	190	236
	4	147	163	167	171	193	238
	5	160	175	179	184	204	252

TABLE A.V.31

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–91%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	157	175	178	183	206	254
	4	155	171	176	180	202	248
	5	166	181	185	191	211	258

TABLE A.V.32

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 85%–91%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	159	177	182	186	207	257
	5	168	184	188	194	215	262

TABLE A.V.33
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	97	122	123	124	146	189
	2	118	137	143	144	166	210
	3	131	148	153	157	175	221
	4	130	149	153	156	175	220
	5	142	160	165	168	188	231

TABLE A.V.34
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	136	148	149	155	177	221
	3	142	159	160	164	184	227
	4	144	159	163	166	186	229
	5	158	172	175	179	197	242

TABLE A.V.35
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	153	170	172	177	198	243
	4	151	167	172	175	196	240
	5	163	178	182	186	205	250

TABLE A.V.36
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 69%–73%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	154	171	175	179	200	246
	5	166	180	185	189	209	254

TABLE A.V.37
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 1 RING USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 27%–28%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	91	114	114	114	135	176
	2	117	134	139	138	159	200
	3	123	138	141	144	162	205
	4	121	138	140	143	161	204
	5	139	154	158	160	178	220

TABLE A.V.38
AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 2 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS ARE IN THE RANGE 27%–28%.

		Number of PGs connected to a 1st level ring (G)					
		1	2	3	4	7	15
Number of 1st level rings (F)	1	---	---	---	---	---	---
	2	134	145	145	150	169	211
	3	134	149	149	152	171	213
	4	135	149	152	154	174	215
	5	154	167	169	173	191	234

TABLE A.V.39

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 3 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE CHANNEL LOAD IS IN THE RANGE 27%–28%, READ CHANNEL LOAD EQUALS 27%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	142	158	159	162	181	224
	4	141	155	159	162	181	223
	5	159	172	175	179	197	240

TABLE A.V.40

AVERAGE LATENCY (EXPRESSED IN CLOCK CYCLES) FOR WRITE CHANNEL FOR 4 PARALLEL RINGS USED AT ROOT LEVEL, AND WRITE AND READ CHANNEL LOADS EQUAL 27%.

		Number of PGs connected to a 1st level ring (*G*)					
		1	2	3	4	7	15
Number of 1st level rings (*F*)	1	---	---	---	---	---	---
	2	---	---	---	---	---	---
	3	---	---	---	---	---	---
	4	144	158	162	165	184	226
	5	161	175	178	182	200	243

VI. Load-latency curves for RingNet

In Appendix IV, load-latency curves for RingNet of various sizes are presented. The results illustrate the conclusions of Section 7.2. The parameters of the test are:

- *R*: Multiplication degree of the root level, i.e., the number of parallel rings used at the root level, set in the range of 1–4.
- *F*: The number of 1st level rings, set in the range of *R*–5. A RingNet network with fewer 1st level rings (*F*) than there are parallel rings used at the root level (*R*) is not tested, as it cannot generate a 100% load.
- *G*: The number of packet generators (PGs) connected to a single 1st level ring, chosen from the set {1, 2, 3, 4, 7, 15}.
- Logical channel load. The aggregated load generated by all PGs is set in the range of 0%–100% of the theoretical throughput T_{RW_MAX} (1). The logical channel load is separately set for the read and write channels.

Load-latency curves for each combination of *R*, *F*, and *G* parameters are depicted in separate figures. 84 figures are presented in the Appendix. Table A.VI.1 lists the ranges of figures corresponding to a given value of the multiplication degree of the root level (*R*) and a given number of PGs connected to a single 1st level ring (*G*). Each range presented in Table A.VI.1 comprises figures for a given *R* and *G* and for an increasing number of 1st level rings (*F*).

TABLE A.VI.1
SUMMARY FOR THE FIGURES PRESENTED IN APPENDIX VI.

		Multiplication degree of the root level (*R*)			
		1	2	3	4
Number of PGs connected to a 1st level ring (*G*)	1	A.VI.1 – A.VI.5	A.VI.31 – A.VI.34	A.VI.55 – A.VI.57	A.VI.73 – A.VI.74
	2	A.VI.6 – A.VI.10	A.VI.35 – A.VI.38	A.VI.58 – A.VI.60	A.VI.75 – A.VI.76
	3	A.VI.11 – A.VI.15	A.VI.39 – A.VI.42	A.VI.61 – A.VI.63	A.VI.77 – A.VI.78
	4	A.VI.16 – A.VI.20	A.VI.43 – A.VI.46	A.VI.64 – A.VI.66	A.VI.79 – A.VI.80
	7	A.VI.21 – A.VI.25	A.VI.47 – A.VI.50	A.VI.67 – A.VI.69	A.VI.81 – A.VI.82
	15	A.VI.26 – A.VI.30	A.VI.51 – A.VI.54	A.VI.70 – A.VI.72	A.VI.83 – A.VI.84

For all tested configurations, the average latency for both logical channels increases with the channel load. An increase of the channel load from 27% to 97% results in the average latency increase by only 10%, whereas the most significant increase of 19% is observed for the configuration *G*=15, *F*=5, *R*=1 (Fig A.VI.30). For the logical channel in saturation, i.e., when the channel load reaches 100%, all network buffers are full, and the average latency increases drastically, as for many other NoCs [Dal03].

The dependence of the read and write logical channels is checked in simulations and depicted in the form of error bars. Error bars represent the minimum and maximum average latency of the channel at a given load when the second channel load changes in the range of 0% to 100%. Dots denote the results for the load of the second channel in the range of 85% to 92% of its maximum throughput. In the tested RingNet implementation, the interaction between the read and write logical channels happens through a physical Leaf-to-Root (L2R) control channel. This is the physical channel used by both logical channels for sending requests to the L2R Manager (see Section 6.3.1 for details). Requests sent for the use of one logical channel can affect the other logical channel, i.e., they can delay the time that the second logical channel sends its request by a few clock cycles. The requested time slots for short and long packet start on a RingNet ring once per each 11-clock cycle burst (cf. Section 6.3.1 and Fig. VI.3), therefore the requested packet slots can be granted every 11-clock cycle. A delayed request may cross the border of the 11-clock cycle burst and postpone the grant of a packet slot by additional 11 clock cycles. This additional latency is observed in many of the figures as error bars with the height of about 12 clock cycles. E.g., in Fig.A.VI.1 such error bars are observed for both logical channels, and in Fig.A.VI.31 and in Fig.A.VI.55 the error bars are observed for the read logical channel and write logical channel, respectively. Nevertheless, the increased latency is negligible and both logical channels can be treated as independent, i.e., the load of one logical channel has a negligible impact on the average latency in the other logical channel, and does not influence its throughput.

For all the 84 figures, which represent 84 different RingNet network configurations, the load-latency curves reach the 100% load value, i.e., throughput TRW_MAX calculated according to (1) is obtained for all tested configurations. As discussed in Section 7.2, the results demonstrate that the RingNet network throughput can be controlled and that it follows equation (1).

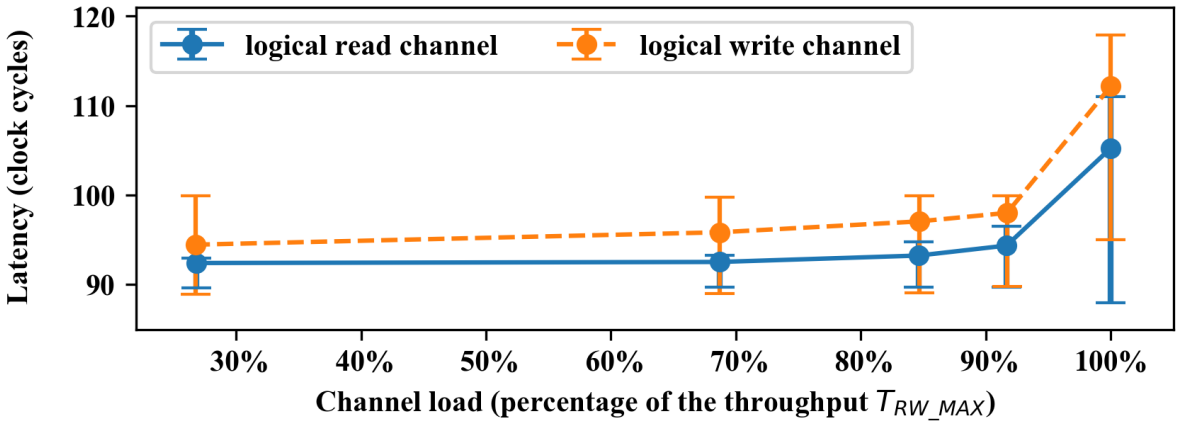


Fig.A.VI.1. Load-latency curve for RingNet with one 1st level ring and 1 PG connected at the ring, and one ring used at the network root (F=1 × G=1, R=1).

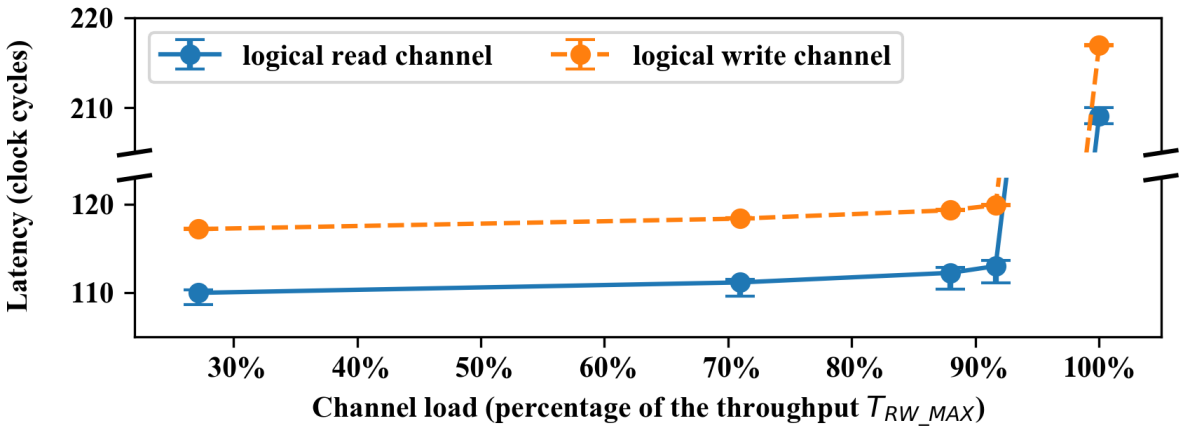


Fig.A.VI.2. Load-latency curve for RingNet with 2 1st level rings and 1 PG connected at each ring, and one ring used at the network root (F=2 × G=1, R=1).

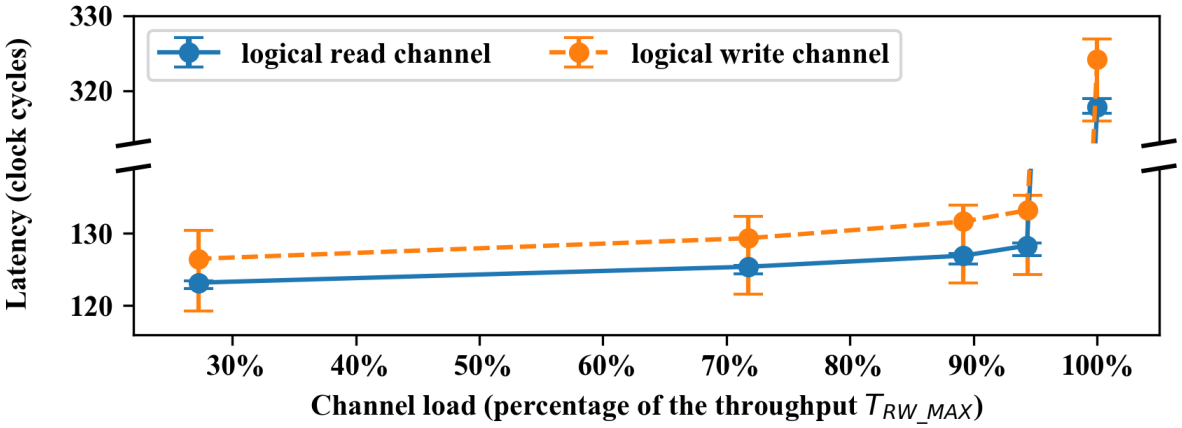


Fig.A.VI.3. Load-latency curve for RingNet with 3 1st level rings and 1 PG connected at each ring, and one ring used at the network root (F=3 × G=1, R=1).

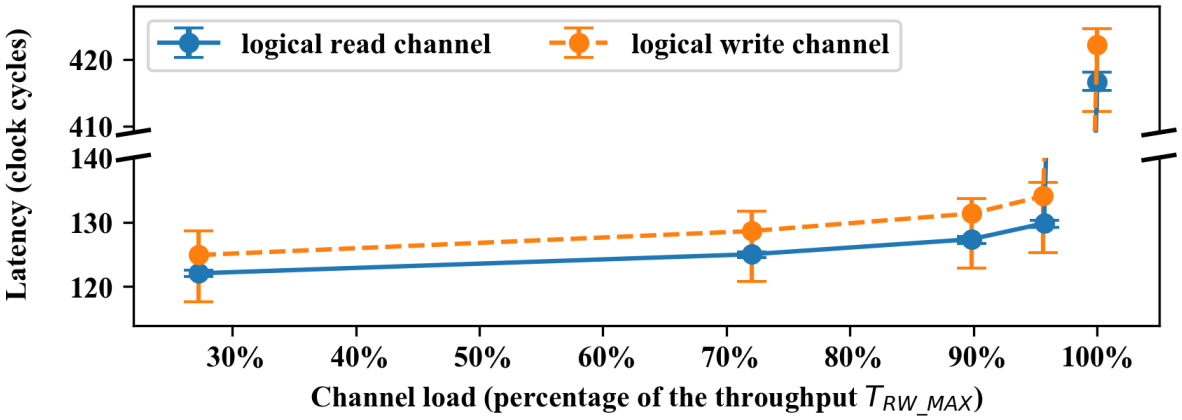


Fig.A.VI.4. Load-latency curve for RingNet with 4 1st level rings and 1 PG connected at each ring, and one ring used at the network root (F=4 × G=1, R=1).

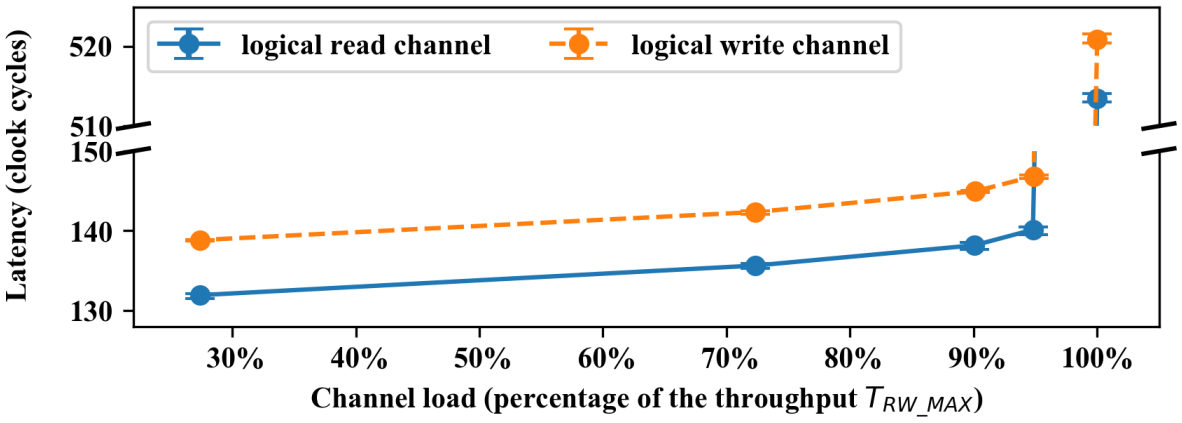


Fig.A.VI.5. Load-latency curve for RingNet with 5 1st level rings and 1 PG connected at each ring, and one ring used at the network root (F=5 × G=1, R=1).

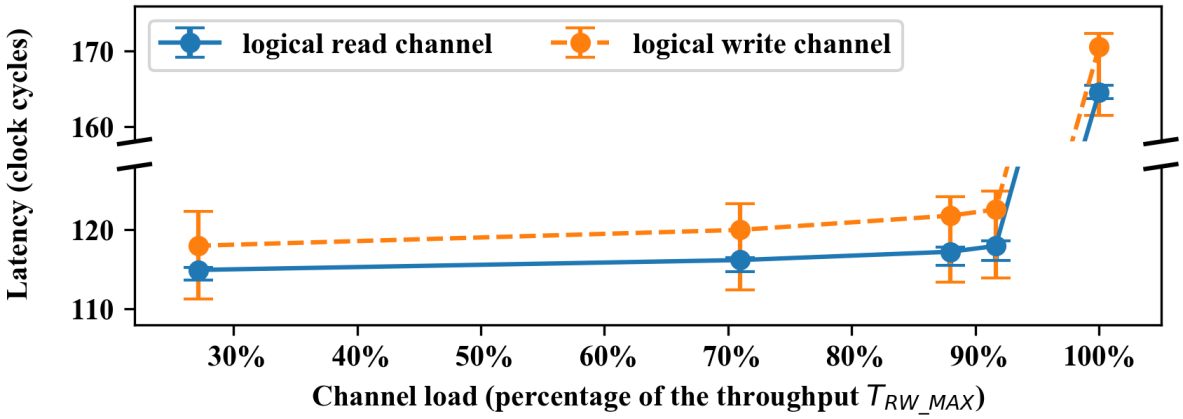


Fig.A.VI.6. Load-latency curve for RingNet with one 1st level ring and 2 PGs connected at the ring, and one ring used at the network root (F=1 × G=2, R=1).

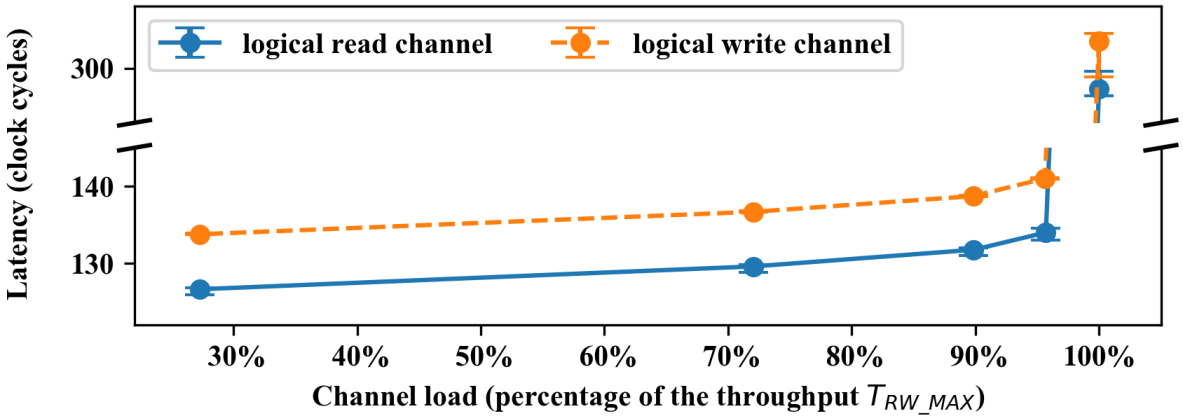


Fig.A.VI.7. Load-latency curve for RingNet with 2 1st level rings and 2 PGs connected at each ring, and one ring used at the network root (F=2 × G=2, R=1).

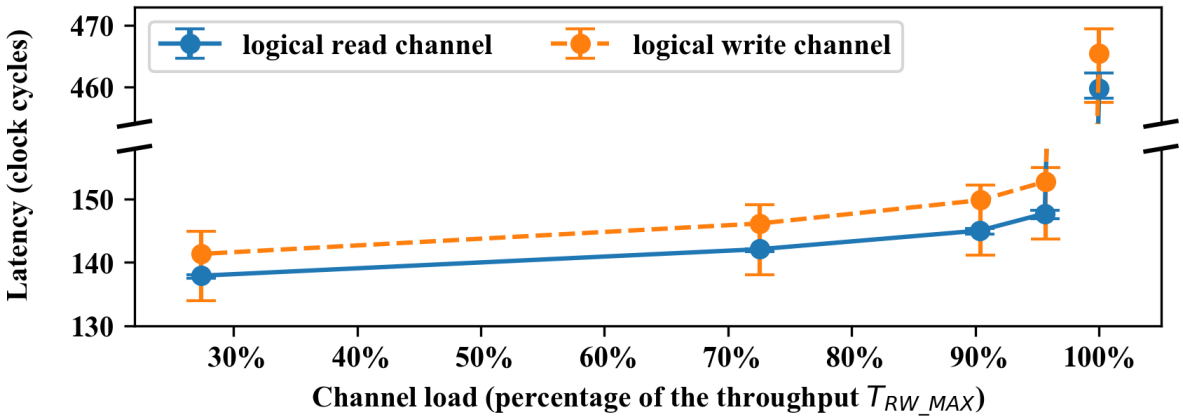


Fig.A.VI.8. Load-latency curve for RingNet with 3 1st level rings and 2 PGs connected at each ring, and one ring used at the network root (F=3 × G=2, R=1).

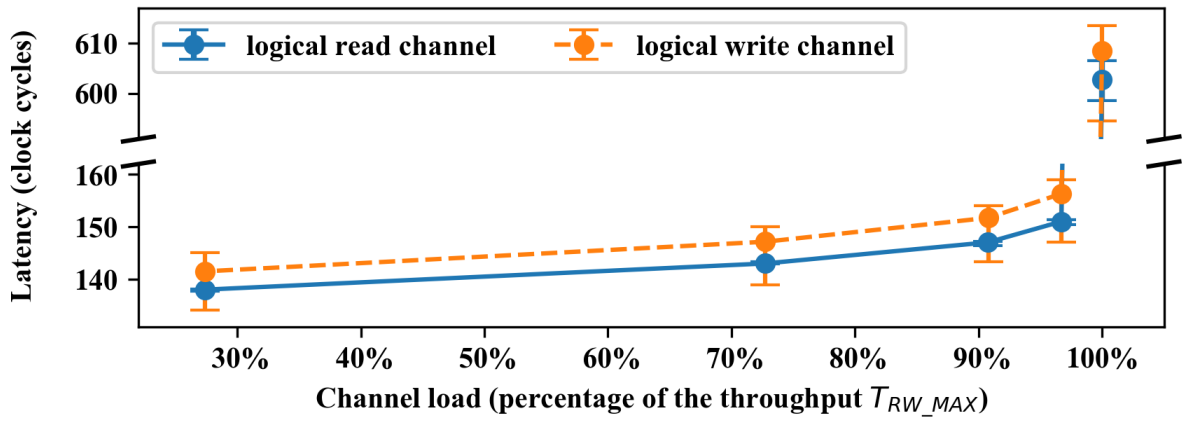


Fig.A.VI.9. Load-latency curve for RingNet with 4 1st level rings and 2 PGs connected at each ring, and one ring used at the network root (F=4 × G=2, R=1).

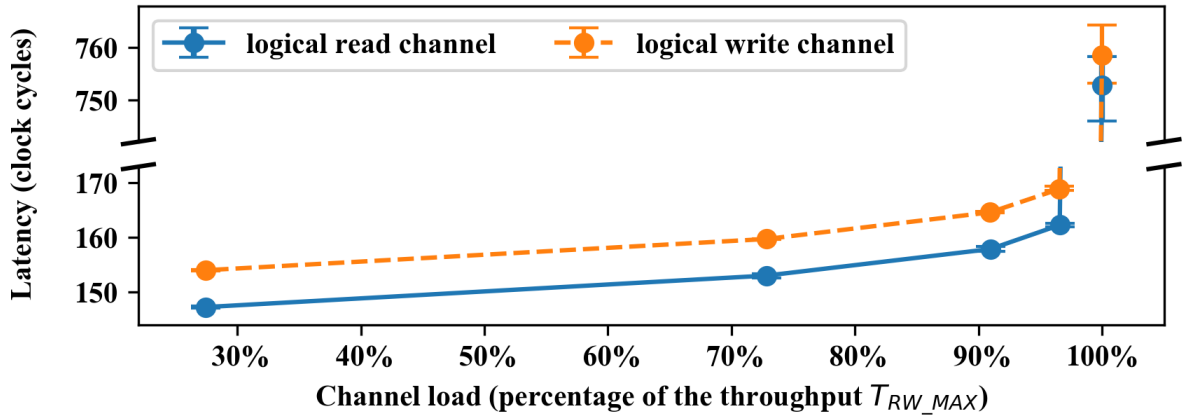


Fig.A.VI.10. Load-latency curve for RingNet with 5 1st level rings and 2 PGs connected at each ring, and one ring used at the network root (F=5 × G=2, R=1).

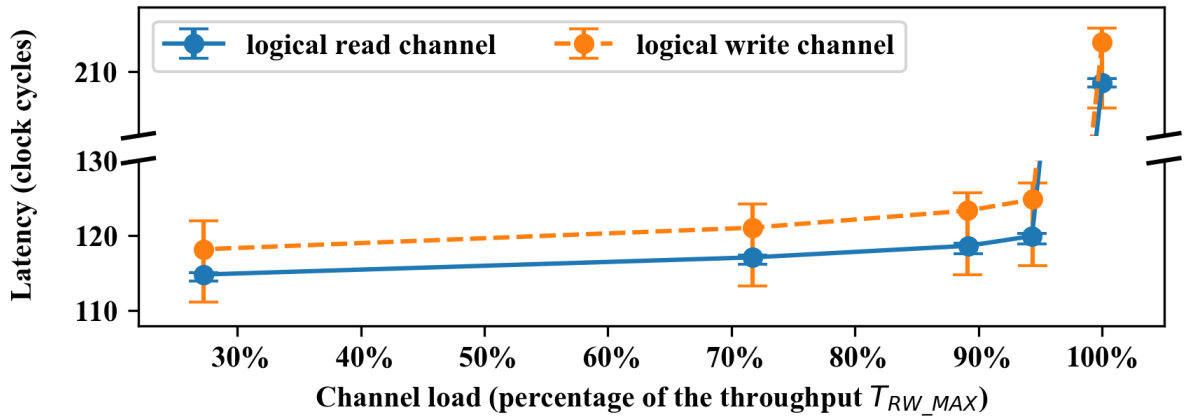


Fig.A.VI.11. Load-latency curve for RingNet with one 1st level ring and 3 PGs connected at the ring, and one ring used at the network root (F=1 × G=3, R=1).

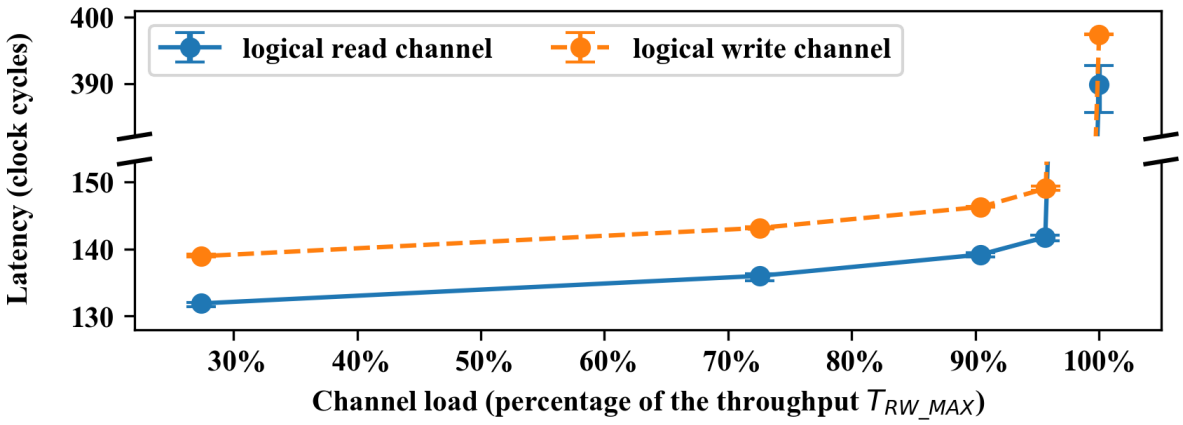


Fig.A.VI.12. Load-latency curve for RingNet with 2 1st level rings and 3 PGs connected at each ring, and one ring used at the network root (F=2 × G=3, R=1).

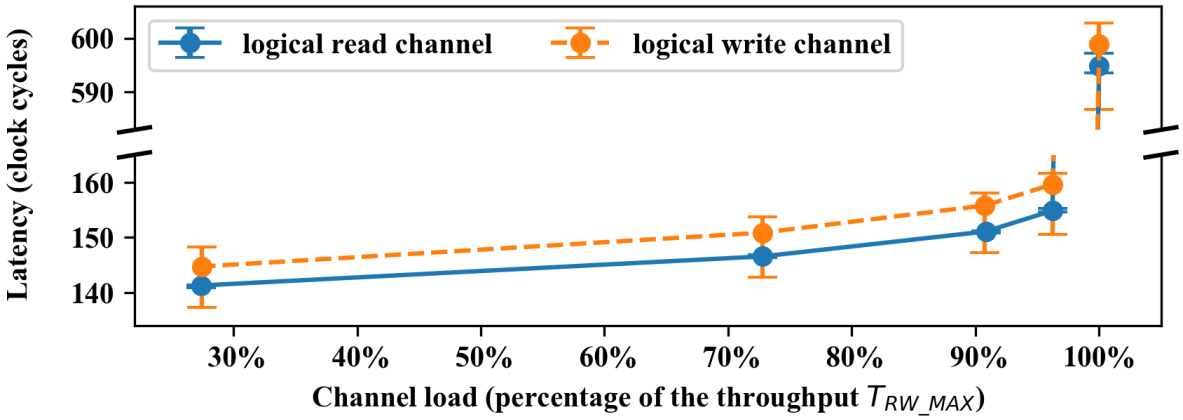


Fig.A.VI.13. Load-latency curve for RingNet with 3 1st level rings and 3 PGs connected at each ring, and one ring used at the network root (F=3 × G=3, R=1).

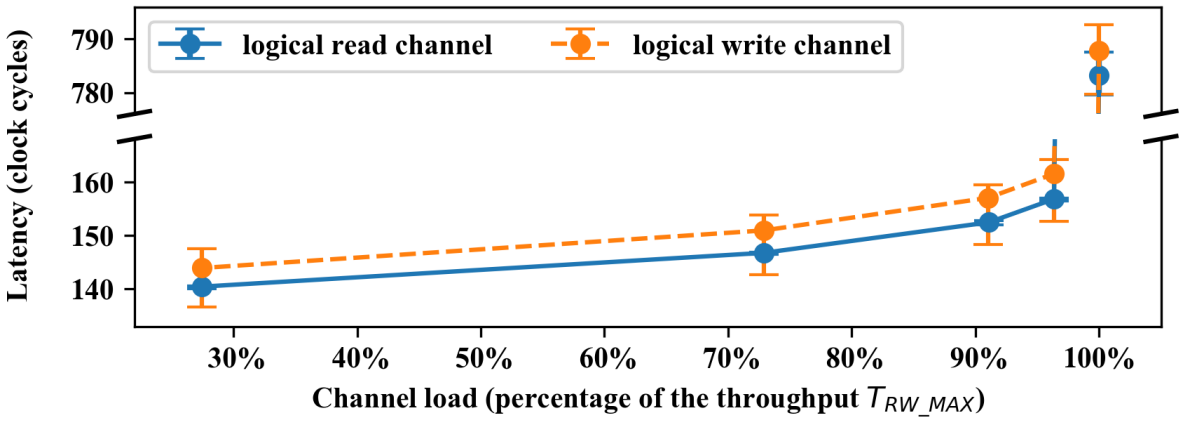


Fig.A.VI.14. Load-latency curve for RingNet with 4 1st level rings and 3 PGs connected at each ring, and one ring used at the network root (F=4 × G=3, R=1).

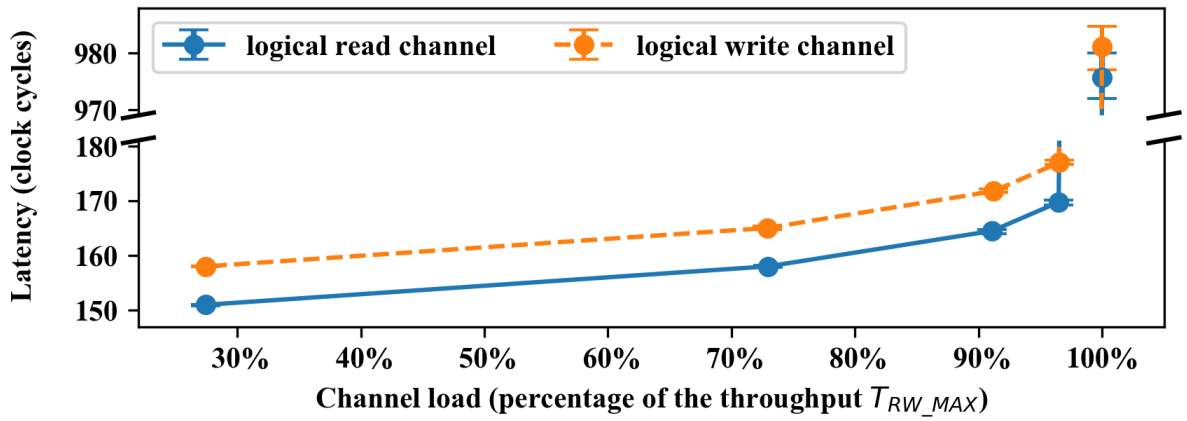


Fig.A.VI.15. Load-latency curve for RingNet with 5 1st level rings and 3 PGs connected at each ring, and one ring used at the network root (F=5 × G=3, R=1).

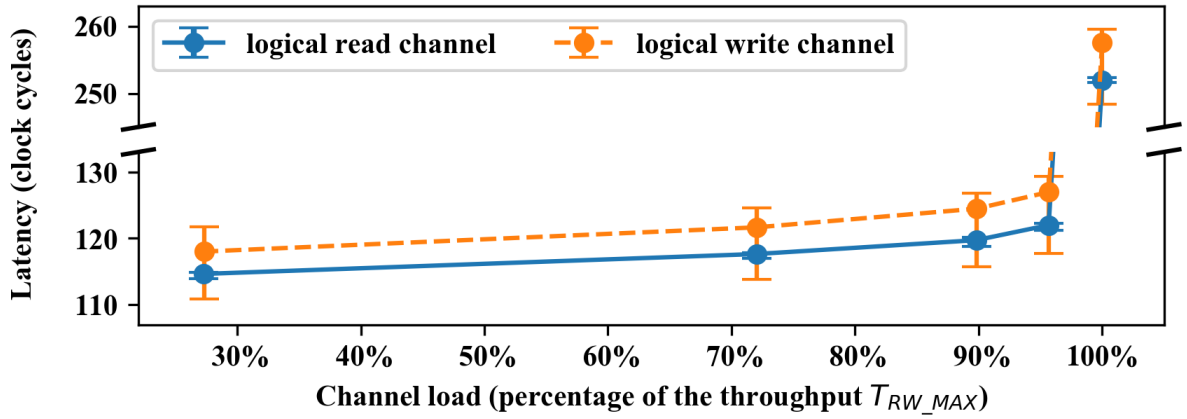


Fig.A.VI.16. Load-latency curve for RingNet with one 1st level ring and 4 PGs connected at the ring, and one ring used at the network root (F=1 × G=4, R=1).

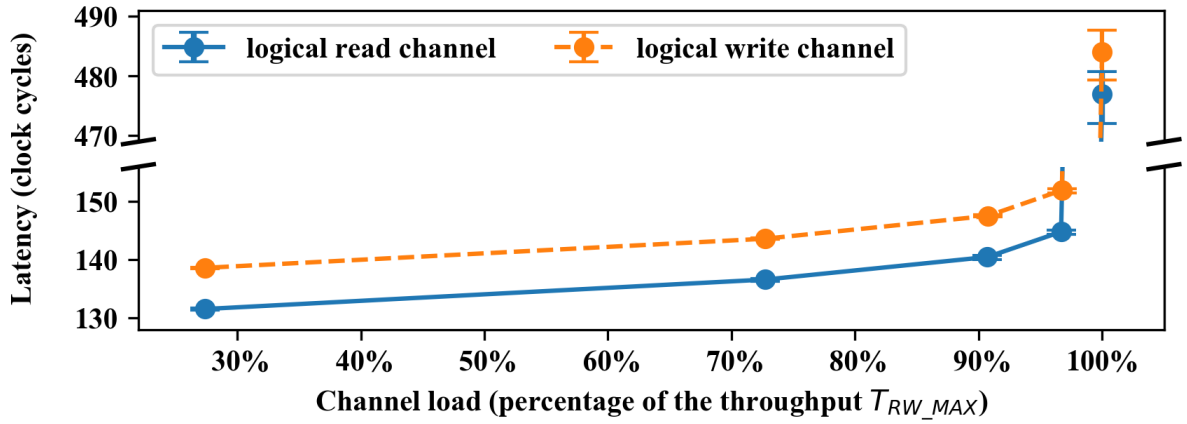


Fig.A.VI.17. Load-latency curve for RingNet with 2 1st level rings and 4 PGs connected at each ring, and one ring used at the network root (F=2 × G=4, R=1).

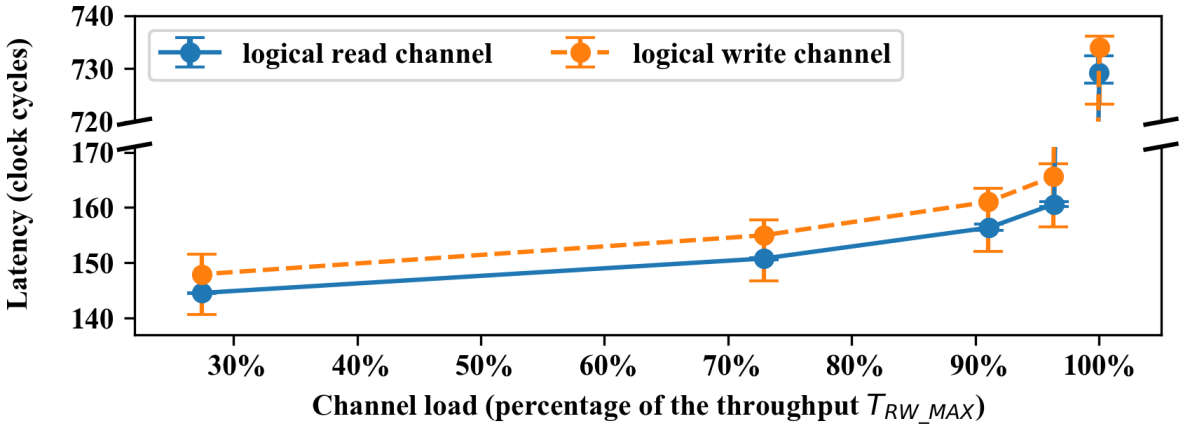


Fig.A.VI.18. Load-latency curve for RingNet with 3 1st level rings and 4 PGs connected at each ring, and one ring used at the network root (F=3 × G=4, R=1).

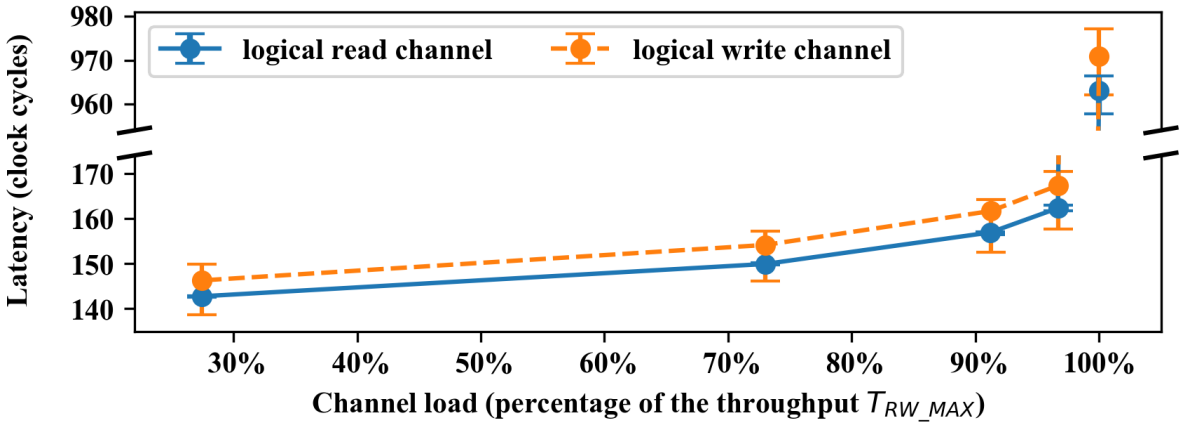


Fig.A.VI.19. Load-latency curve for RingNet with 4 1st level rings and 4 PGs connected at each ring, and one ring used at the network root (F=4 × G=4, R=1).

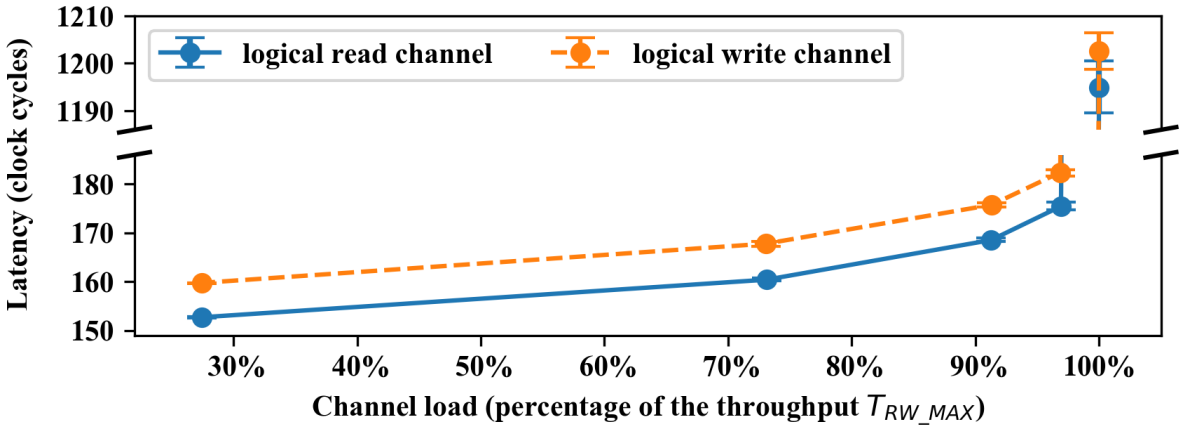


Fig.A.VI.20. Load-latency curve for RingNet with 5 1st level rings and 4 PGs connected at each ring, and one ring used at the network root (F=5 × G=4, R=1).

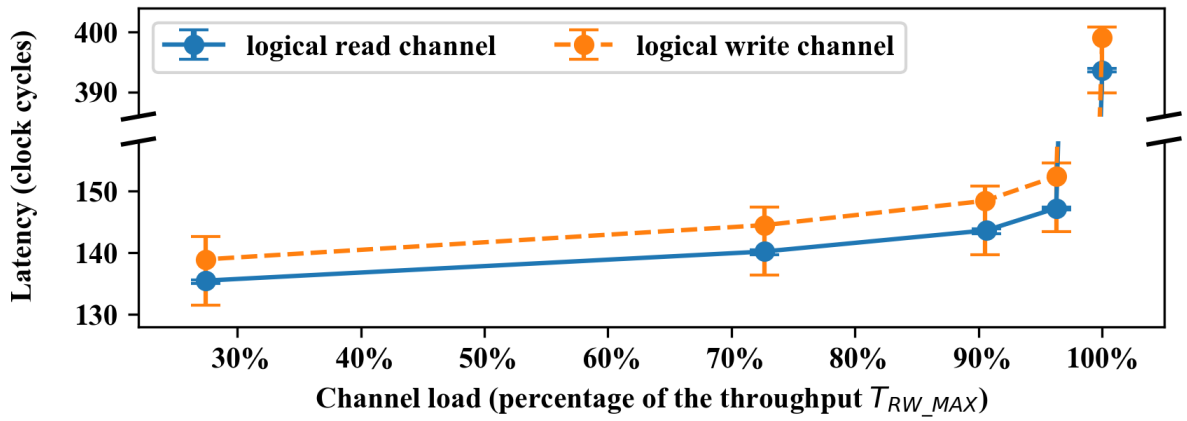


Fig.A.VI.21. Load-latency curve for RingNet with one 1st level ring and 7 PGs connected at the ring, and one ring used at the network root (F=1 × G=7, R=1).

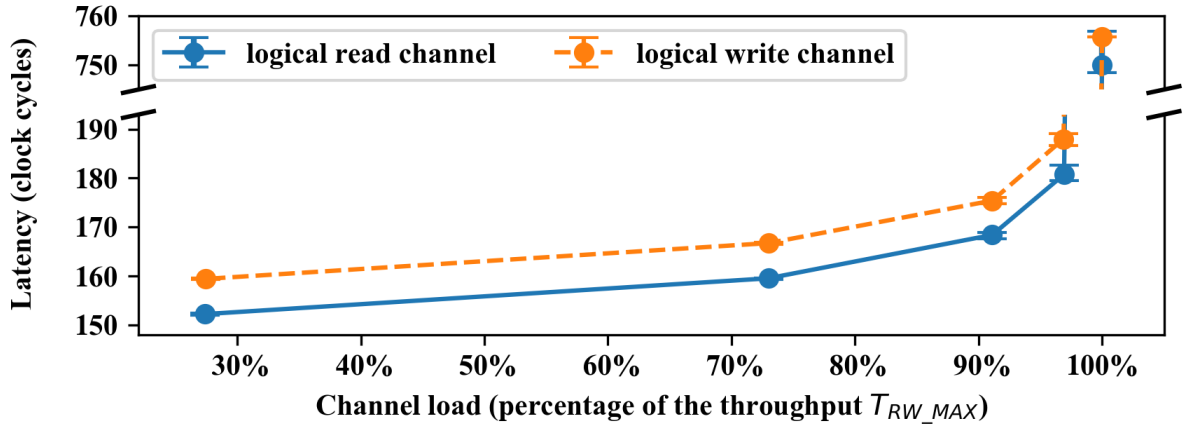


Fig.A.VI.22. Load-latency curve for RingNet with 2 1st level rings and 7 PGs connected at each ring, and one ring used at the network root (F=2 × G=7, R=1).

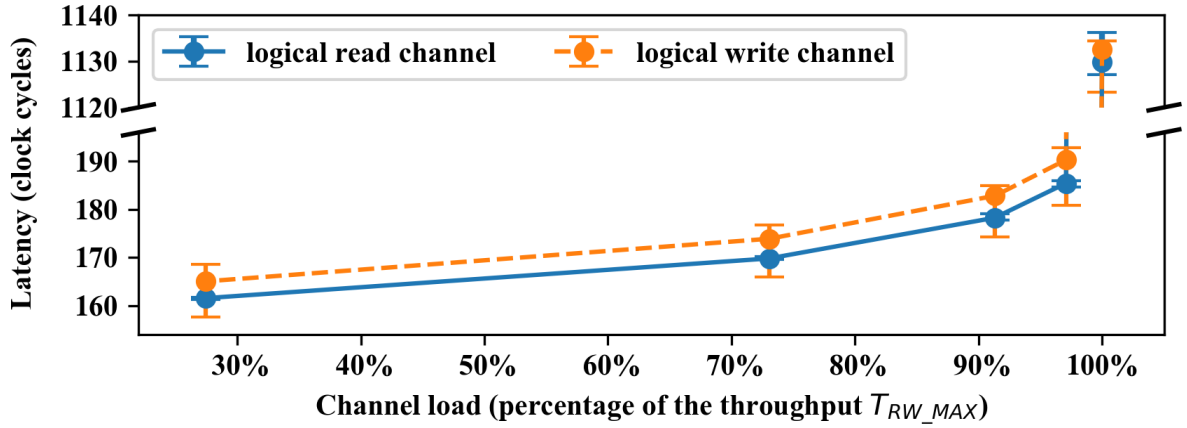


Fig.A.VI.23. Load-latency curve for RingNet with 3 1st level rings and 7 PGs connected at each ring, and one ring used at the network root (F=3 × G=7, R=1).

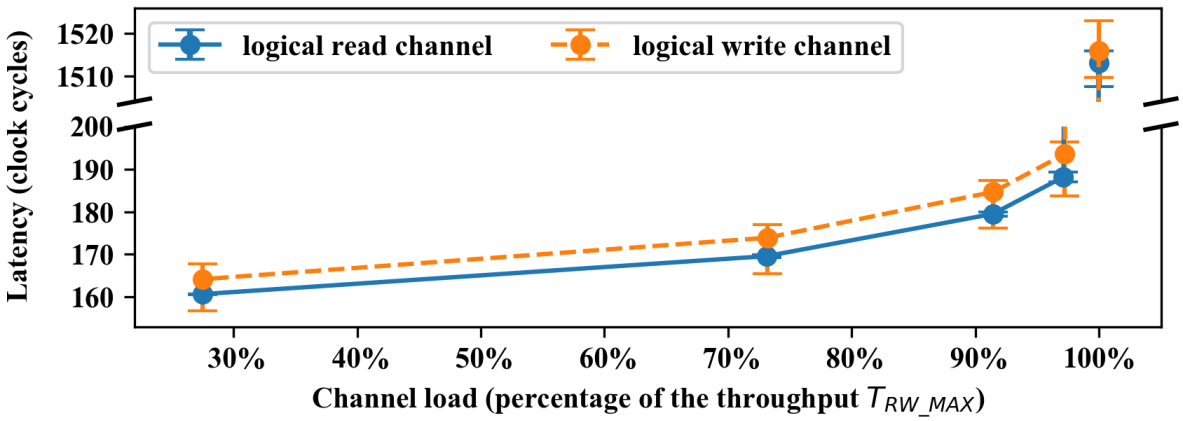


Fig.A.VI.24. Load-latency curve for RingNet with 4 1st level rings and 7 PGs connected at each ring, and one ring used at the network root (F=4 × G=7, R=1).

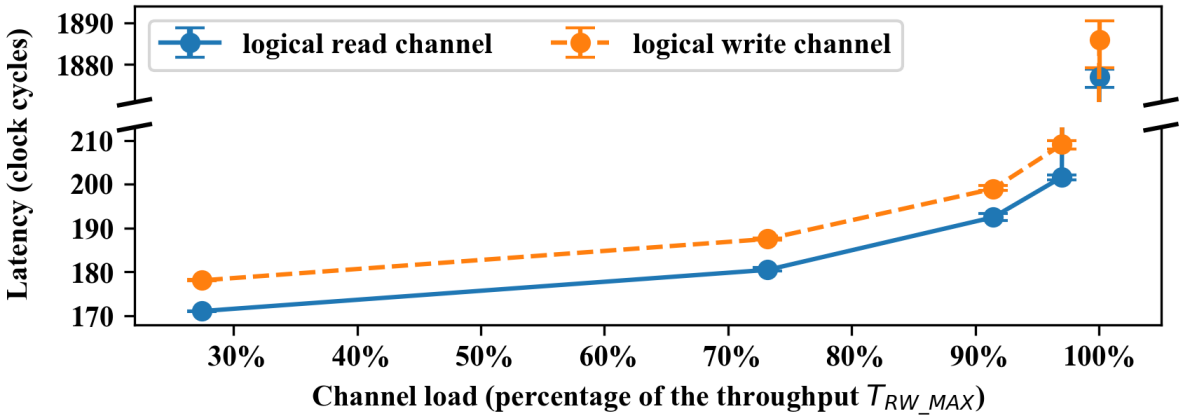


Fig.A.VI.25. Load-latency curve for RingNet with 5 1st level rings and 7 PGs connected at each ring, and one ring used at the network root (F=5 × G=7, R=1).

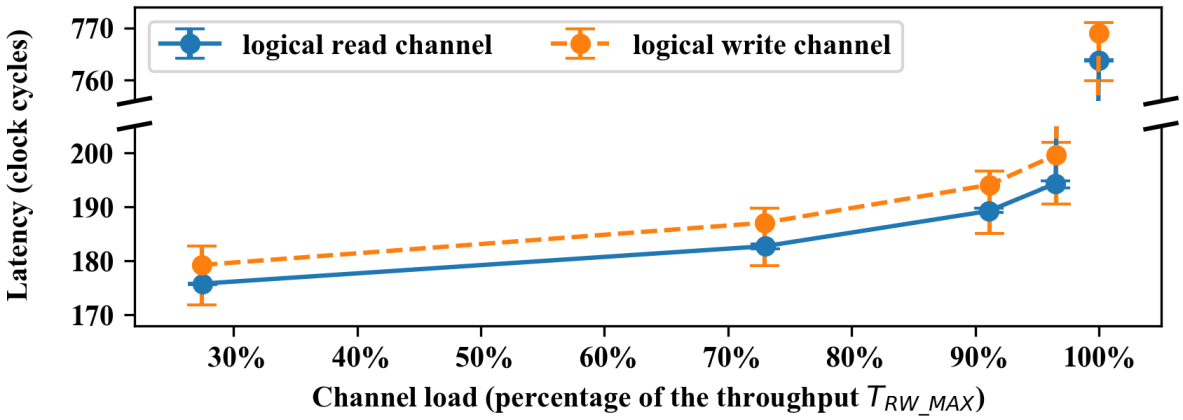


Fig.A.VI.26. Load-latency curve for RingNet with one 1st level ring and 15 PGs connected at the ring, and one ring used at the network root (F=1 × G=15, R=1).

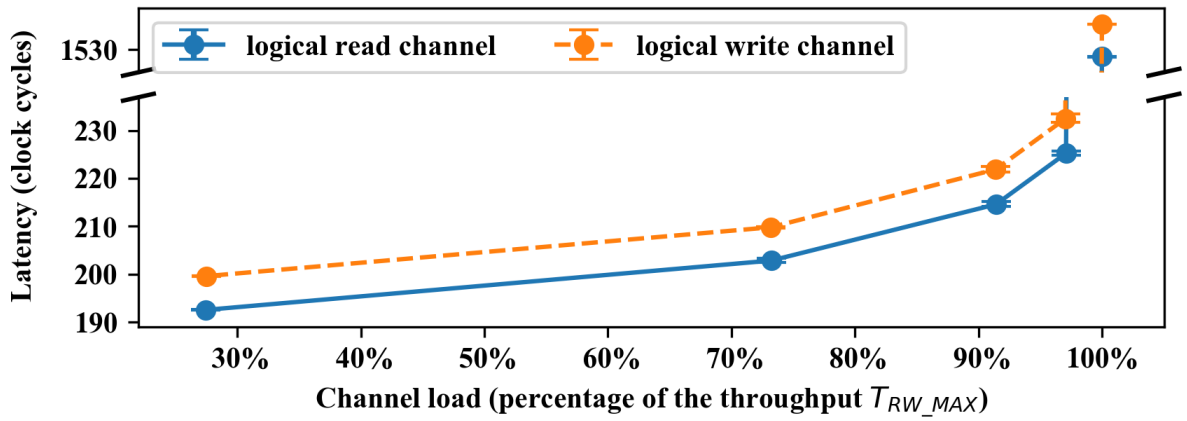


Fig.A.VI.27. Load-latency curve for RingNet with 2 1st level rings and 15 PGs connected at each ring, and one ring used at the network root (F=2 × G=15, R=1).

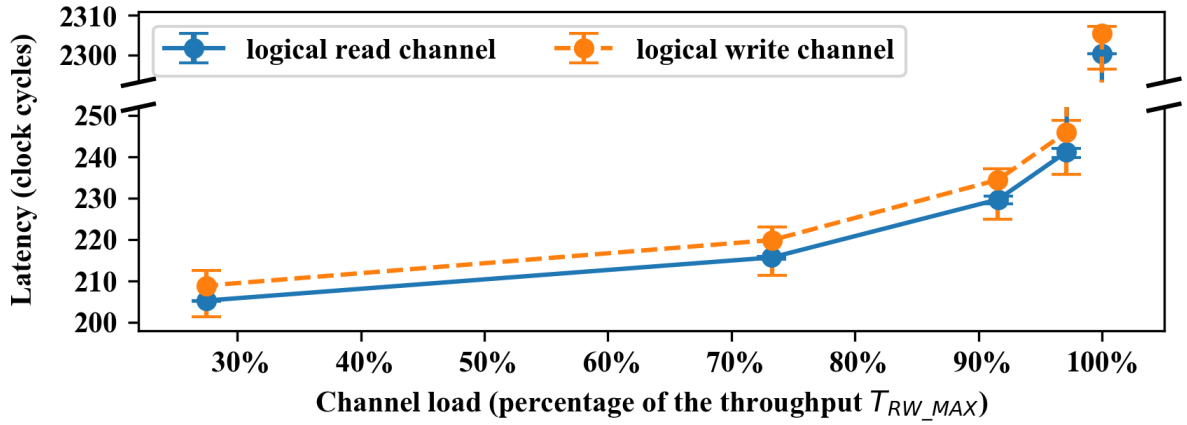


Fig.A.VI.28. Load-latency curve for RingNet with 3 1st level rings and 15 PGs connected at each ring, and one ring used at the network root (F=3 × G=15, R=1).

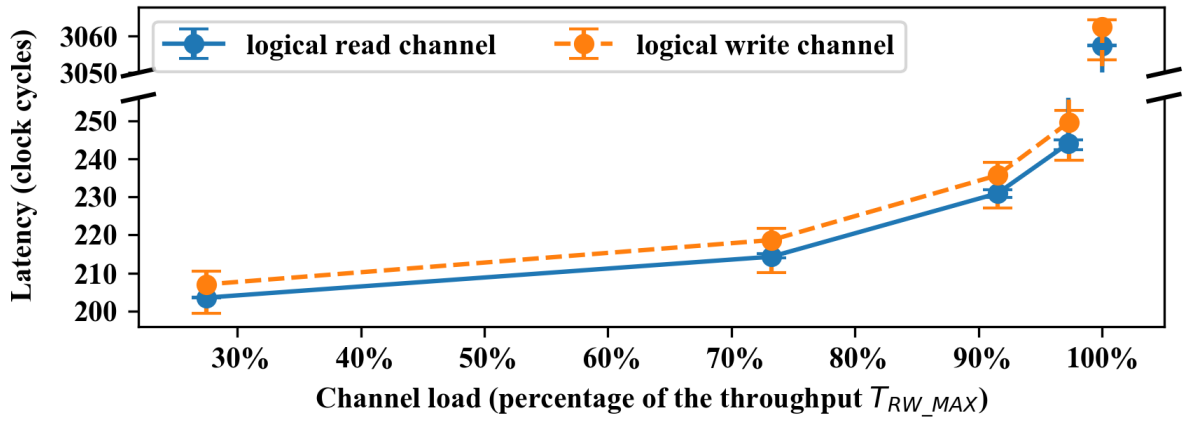


Fig.A.VI.29. Load-latency curve for RingNet with 4 1st level rings and 15 PGs connected at each ring, and one ring used at the network root (F=4 × G=15, R=1).

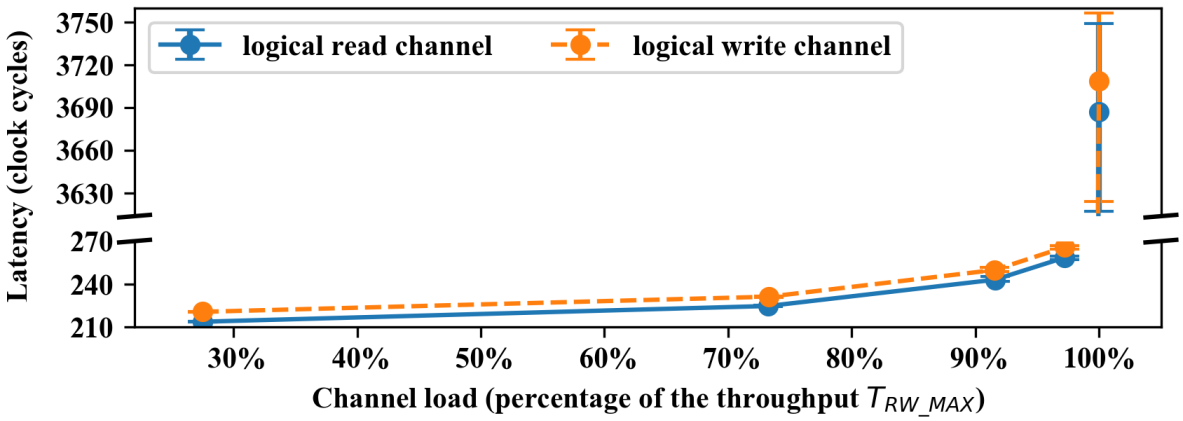


Fig.A.VI.30. Load-latency curve for RingNet with 5 1st level rings and 15 PGs connected at each ring, and one ring used at the network root (F=5 × G=15, R=1).

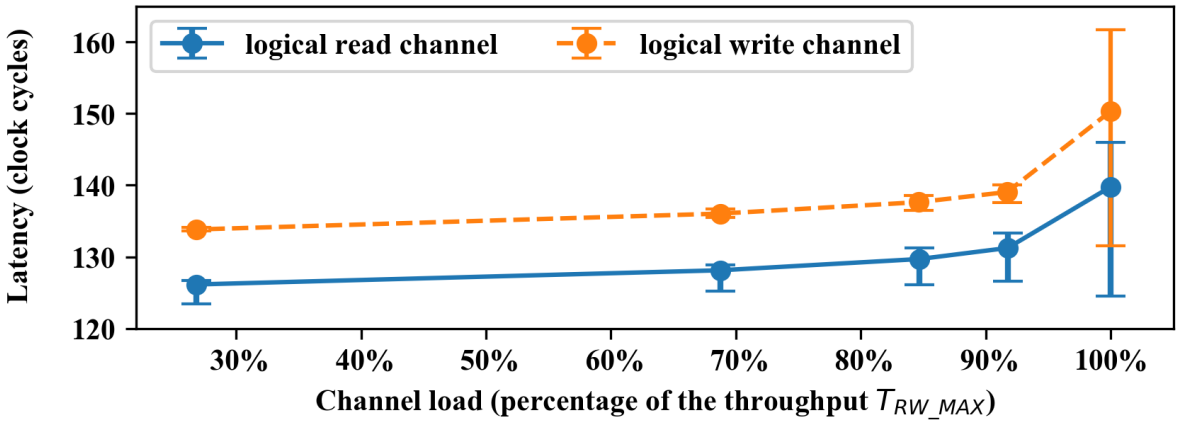


Fig.A.VI.31. Load-latency curve for RingNet with 2 1st level rings and 1 PG connected at each ring, and 2 parallel rings used at the network root (F=2 × G=1, R=2).

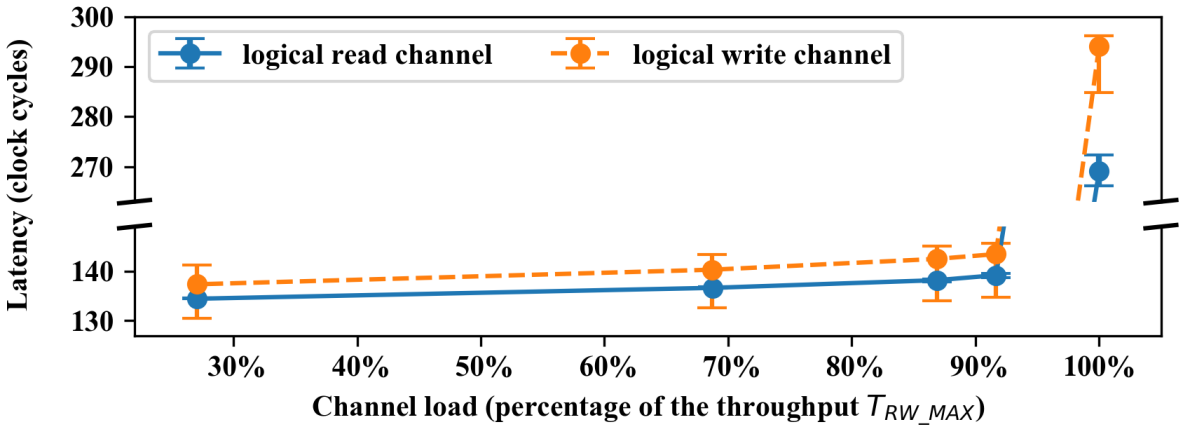


Fig.A.VI.32. Load-latency curve for RingNet with 3 1st level rings and 1 PG connected at each ring, and 2 parallel rings used at the network root (F=3 × G=1, R=2).

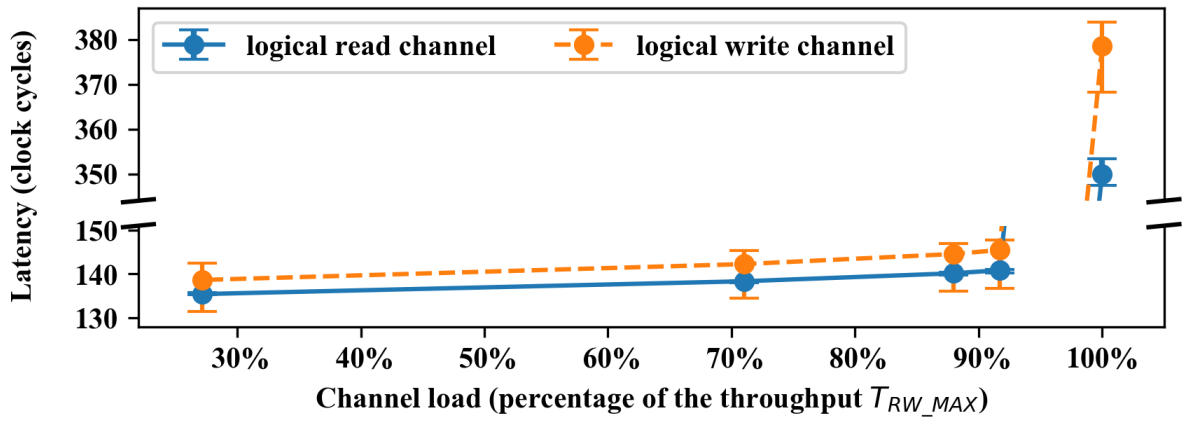


Fig.A.VI.33. Load-latency curve for RingNet with 4 1st level rings and 1 PG connected at each ring, and 2 parallel rings used at the network root (F=4 × G=1, R=2).

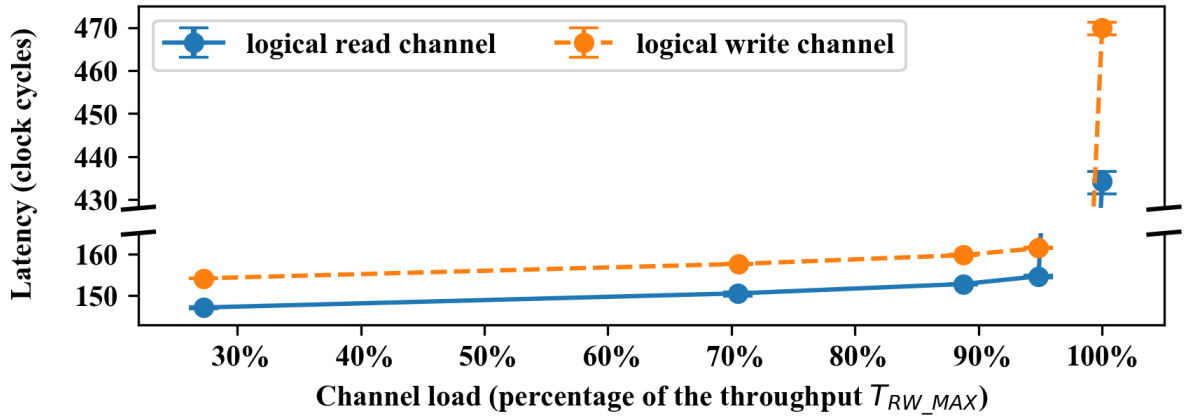


Fig.A.VI.34. Load-latency curve for RingNet with 5 1st level rings and 1 PG connected at each ring, and 2 parallel rings used at the network root (F=5 × G=1, R=2).

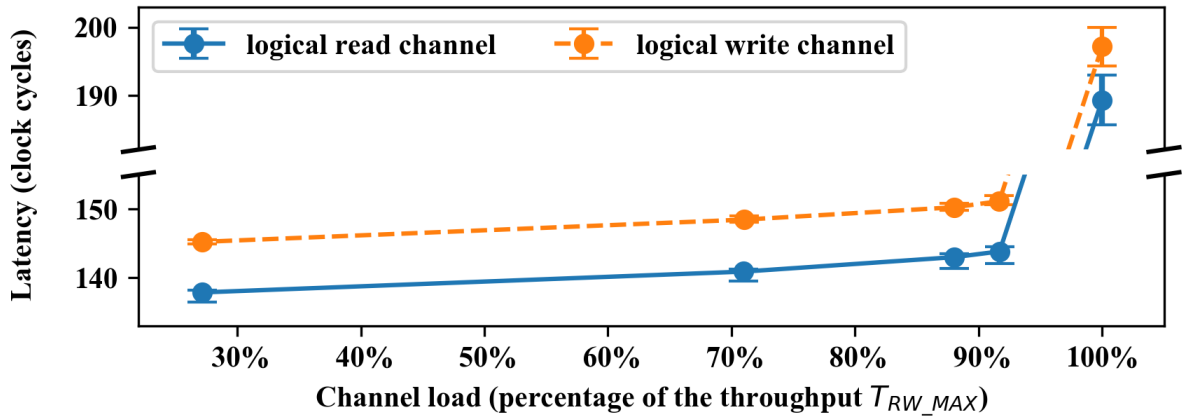


Fig.A.VI.35. Load-latency curve for RingNet with 2 1st level rings and 2 PGs connected at each ring, and 2 parallel rings used at the network root (F=2 × G=2, R=2).

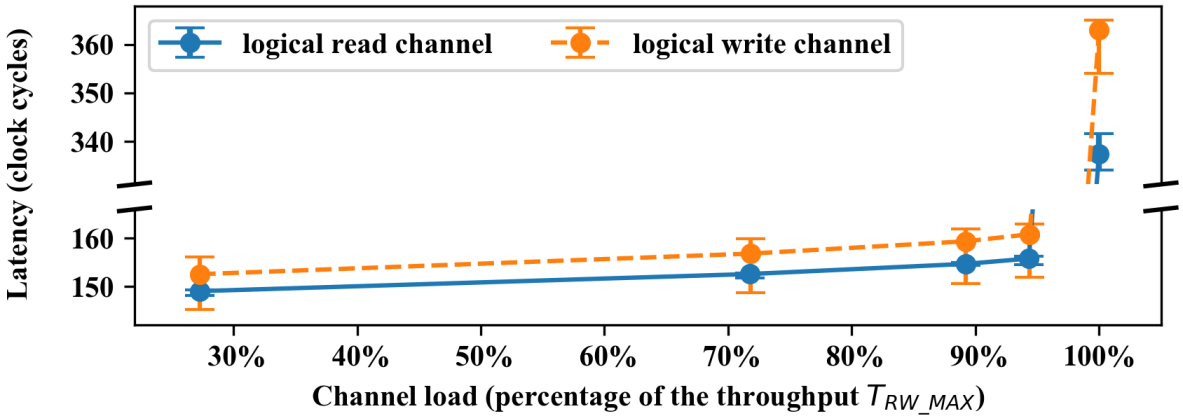


Fig.A.VI.36. Load-latency curve for RingNet with 3 1st level rings and 2 PGs connected at each ring, and 2 parallel rings used at the network root (F=3 × G=2, R=2).

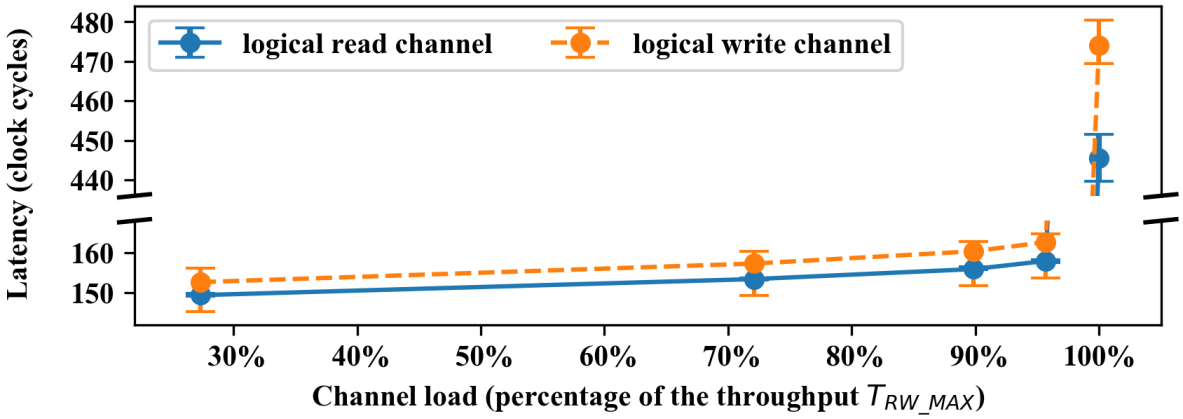


Fig.A.VI.37. Load-latency curve for RingNet with 4 1st level rings and 2 PGs connected at each ring, and 2 parallel rings used at the network root (F=4 × G=2, R=2).

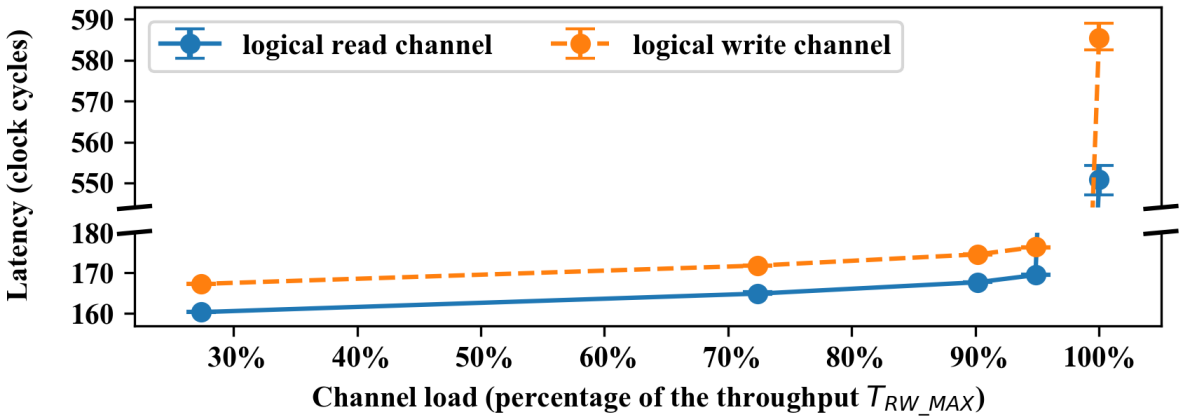


Fig.A.VI.38. Load-latency curve for RingNet with 5 1st level rings and 2 PGs connected at each ring, and 2 parallel rings used at the network root (F=5 × G=2, R=2).

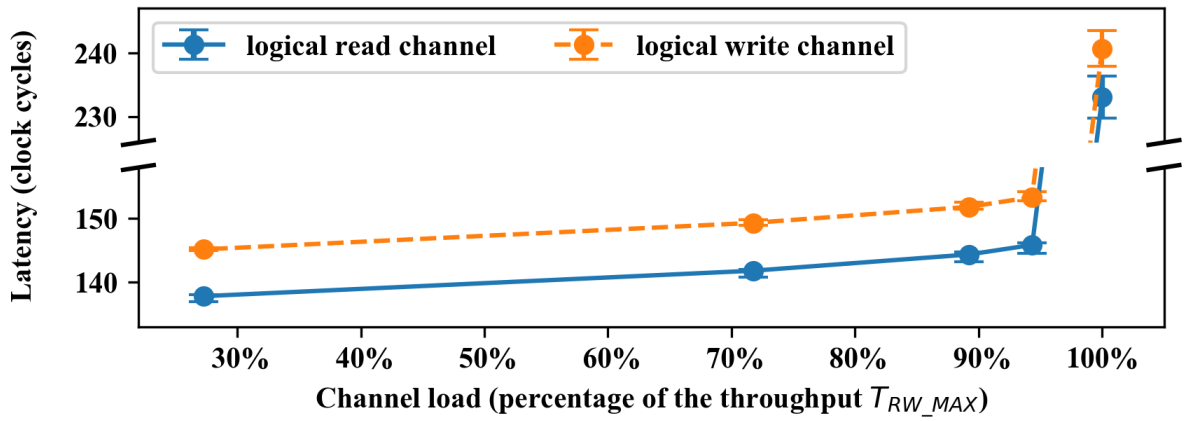


Fig.A.VI.39. Load-latency curve for RingNet with 2 1st level rings and 3 PGs connected at each ring, and 2 parallel rings used at the network root (F=2 × G=3, R=2).

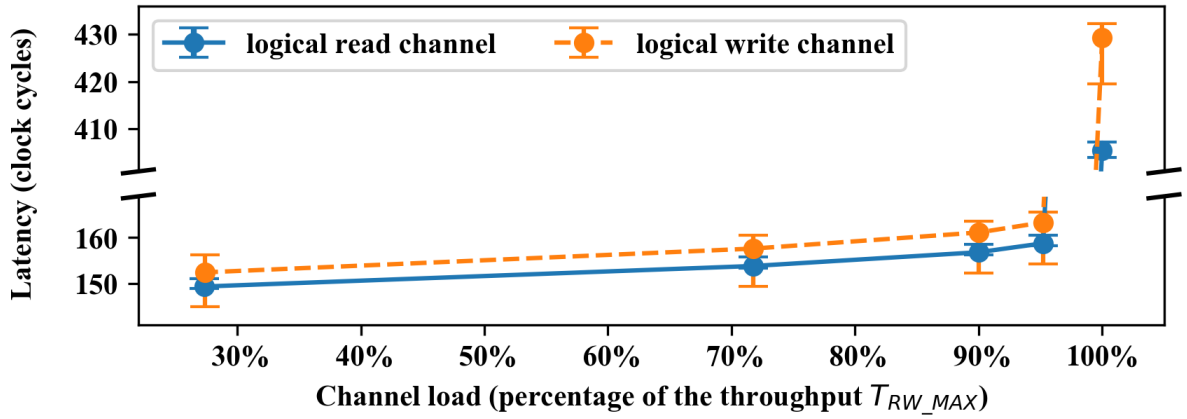


Fig.A.VI.40. Load-latency curve for RingNet with 3 1st level rings and 3 PGs connected at each ring, and 2 parallel rings used at the network root (F=3 × G=3, R=2).

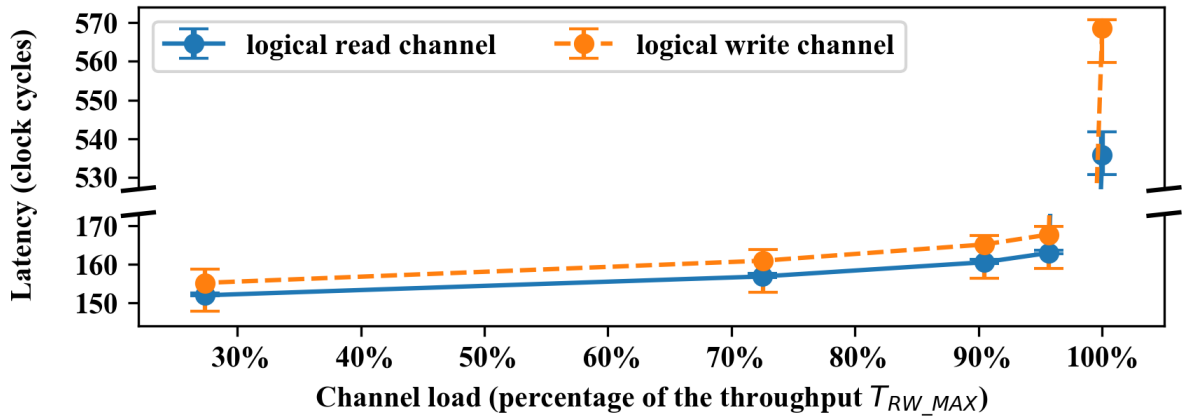


Fig.A.VI.41. Load-latency curve for RingNet with 4 1st level rings and 3 PGs connected at each ring, and 2 parallel rings used at the network root (F=4 × G=3, R=2).

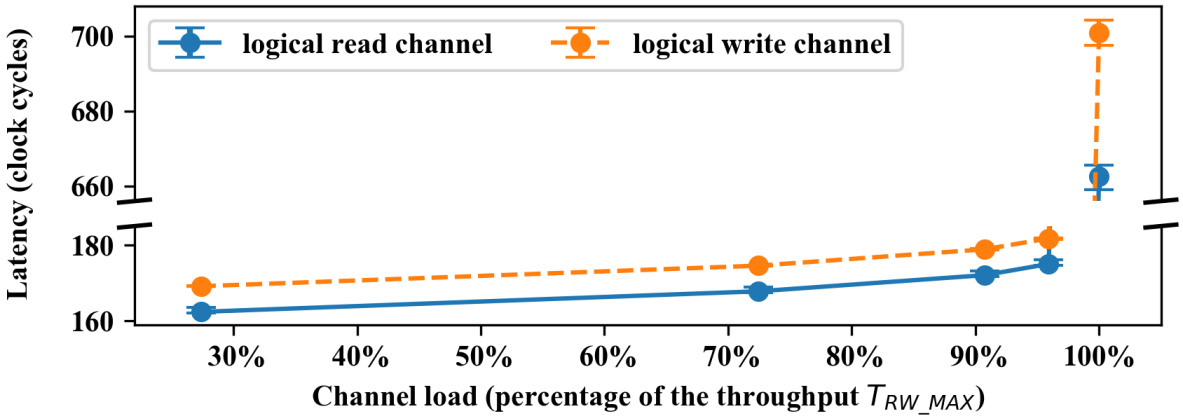


Fig.A.VI.42. Load-latency curve for RingNet with 5 1st level rings and 3 PGs connected at each ring, and 2 parallel rings used at the network root (F=5 × G=3, R=2).

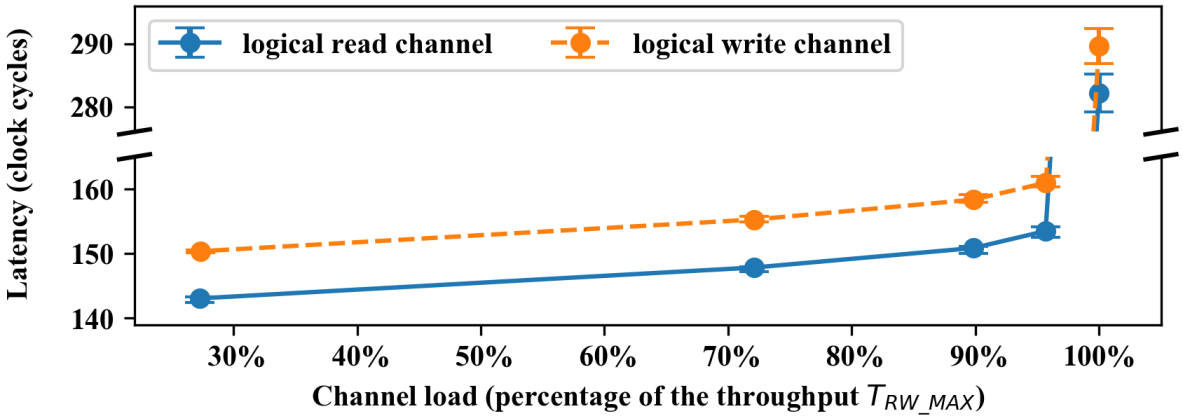


Fig.A.VI.43. Load-latency curve for RingNet with 2 1st level rings and 4 PGs connected at each ring, and 2 parallel rings used at the network root (F=2 × G=4, R=2).

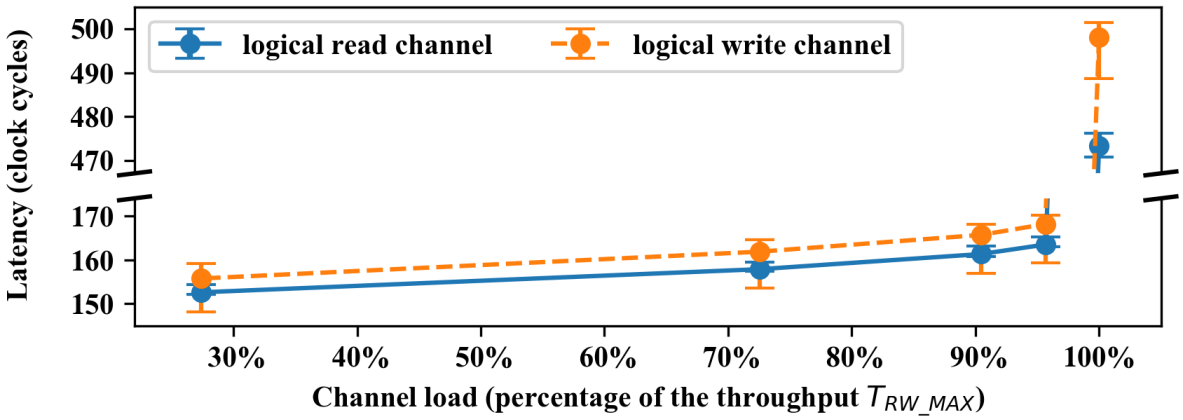


Fig.A.VI.44. Load-latency curve for RingNet with 3 1st level rings and 4 PGs connected at each ring, and 2 parallel rings used at the network root (F=3 × G=4, R=2).

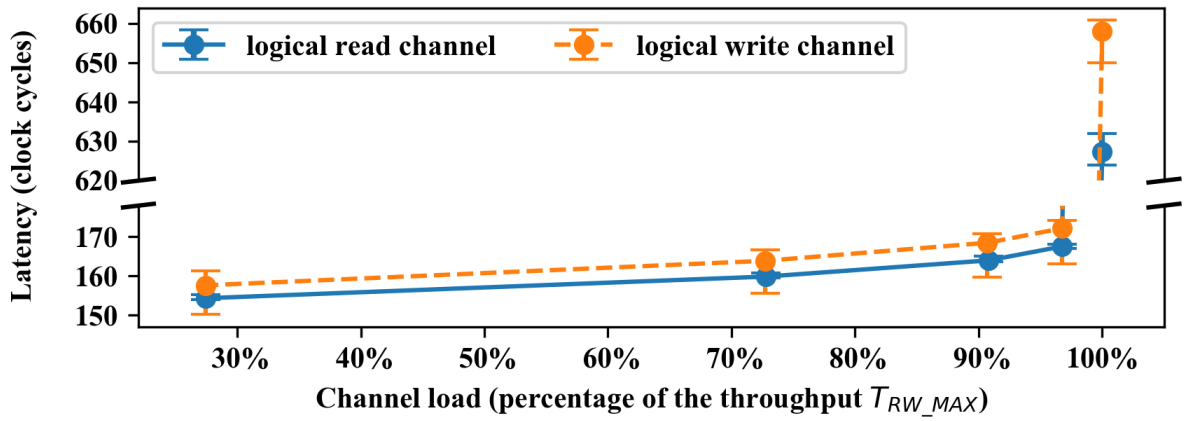


Fig.A.VI.45. Load-latency curve for RingNet with 4 1st level rings and 4 PGs connected at each ring, and 2 parallel rings used at the network root (F=4 × G=4, R=2).

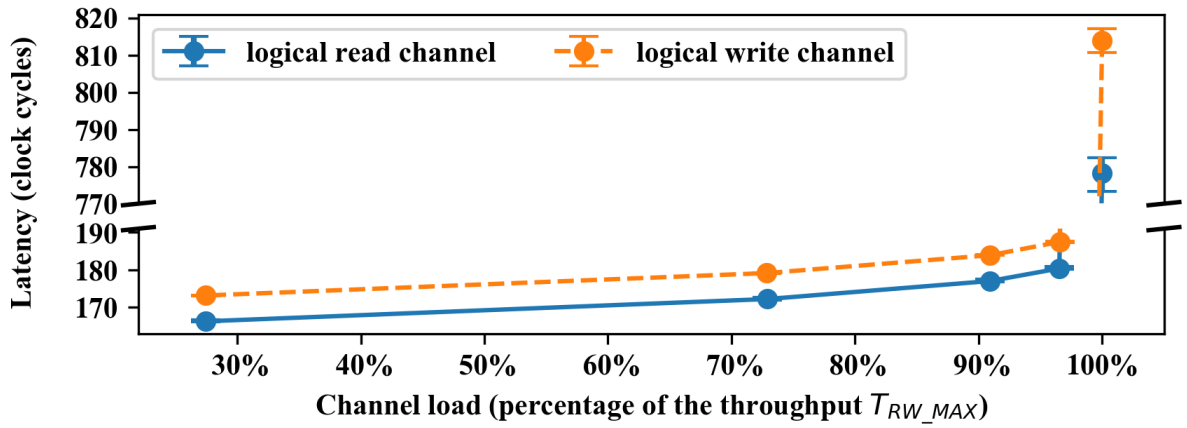


Fig.A.VI.46. Load-latency curve for RingNet with 5 1st level rings and 4 PGs connected at each ring, and 2 parallel rings used at the network root (F=5 × G=4, R=2).

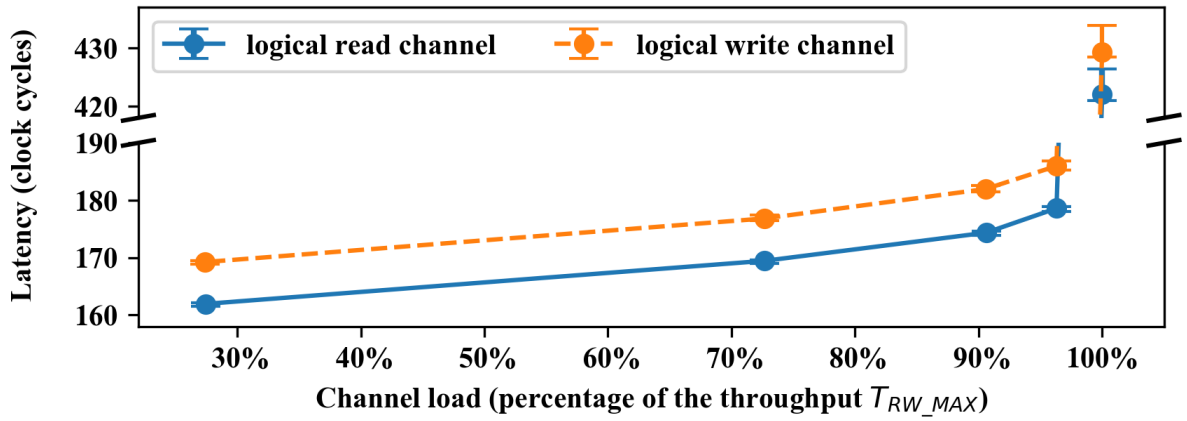


Fig.A.VI.47. Load-latency curve for RingNet with 2 1st level rings and 7 PGs connected at each ring, and 2 parallel rings used at the network root (F=2 × G=7, R=2).

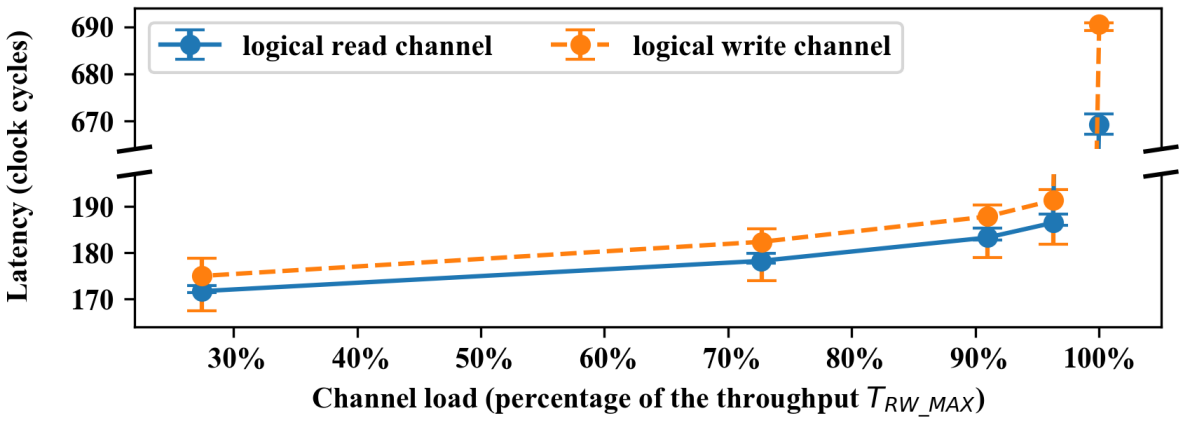


Fig.A.VI.48. Load-latency curve for RingNet with 3 1st level rings and 7 PGs connected at each ring, and 2 parallel rings used at the network root (F=3 × G=7, R=2).

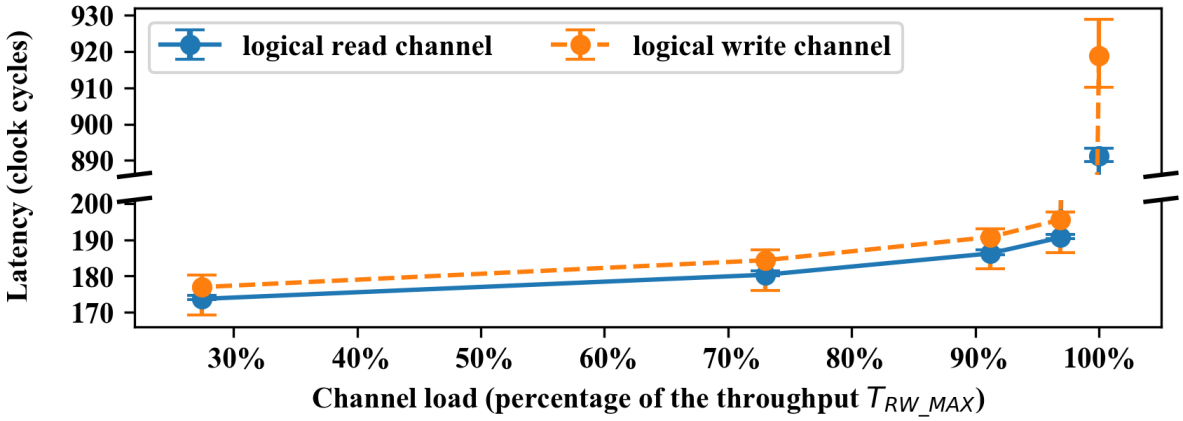


Fig.A.VI.49. Load-latency curve for RingNet with 4 1st level rings and 7 PGs connected at each ring, and 2 parallel rings used at the network root (F=4 × G=7, R=2).

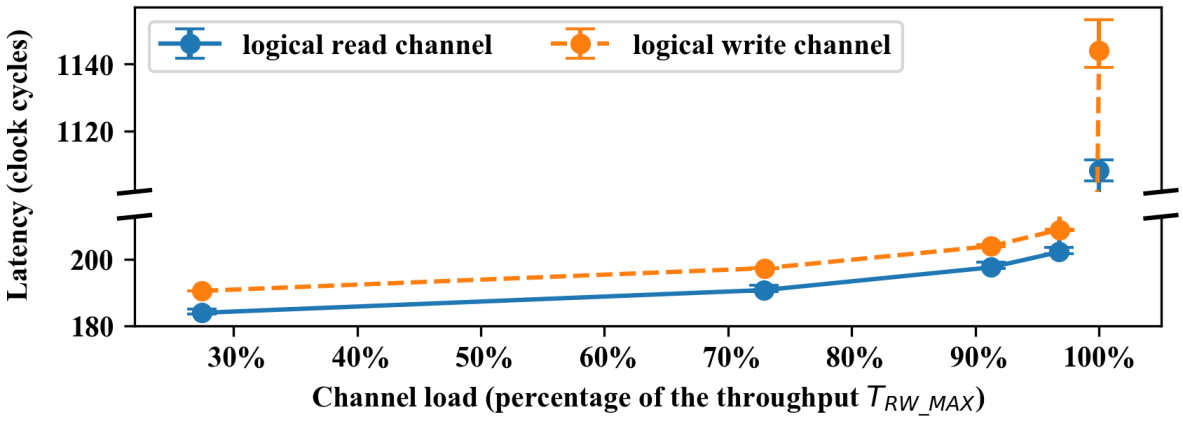


Fig.A.VI.50. Load-latency curve for RingNet with 5 1st level rings and 7 PGs connected at each ring, and 2 parallel rings used at the network root (F=5 × G=7, R=2).

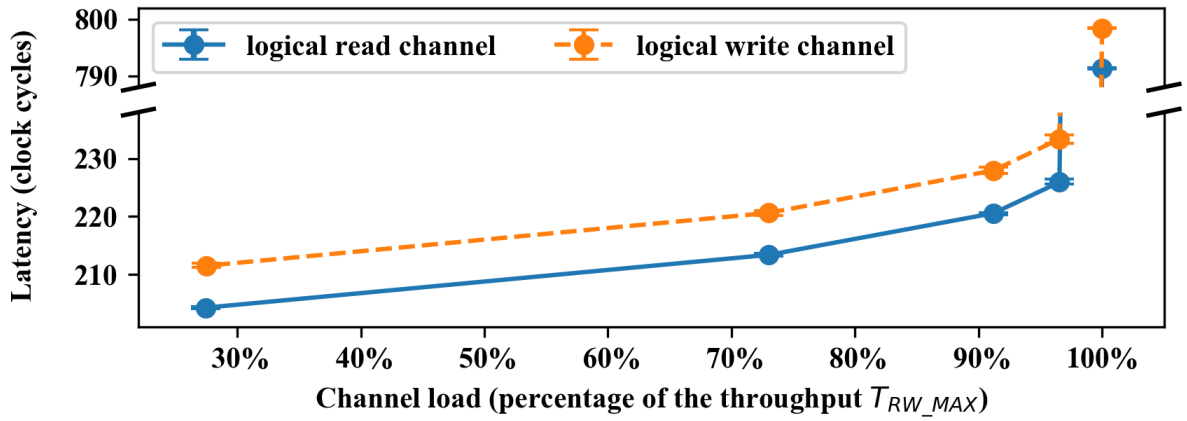


Fig.A.VI.51. Load-latency curve for RingNet with 2 1st level rings and 15 PGs connected at each ring, and 2 parallel rings used at the network root (F=2 × G=15, R=2).

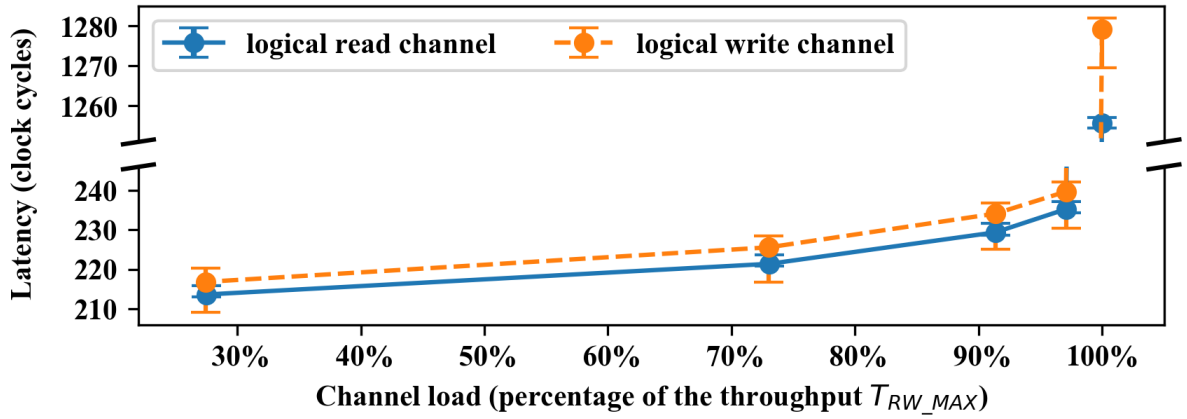


Fig.A.VI.52. Load-latency curve for RingNet with 3 1st level rings and 15 PGs connected at each ring, and 2 parallel rings used at the network root (F=3 × G=15, R=2).

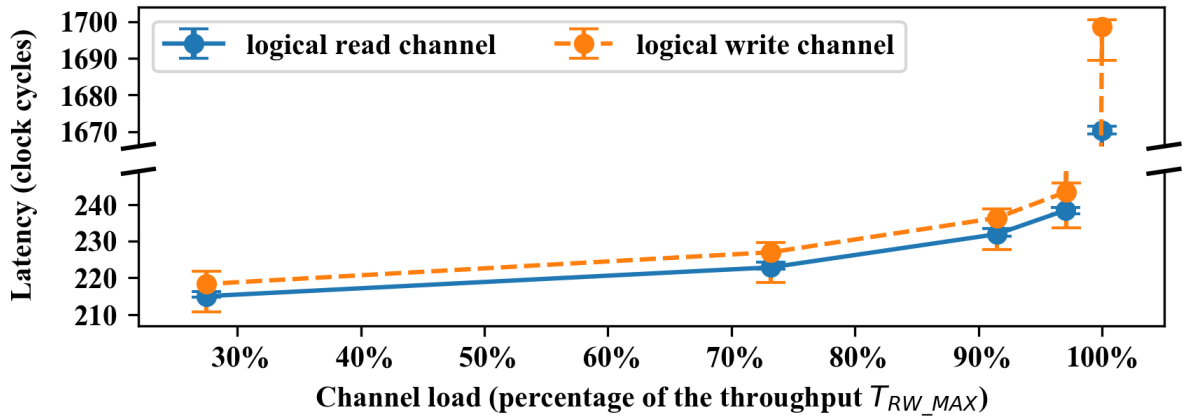


Fig.A.VI.53. Load-latency curve for RingNet with 4 1st level rings and 15 PGs connected at each ring, and 2 parallel rings used at the network root (F=4 × G=15, R=2).

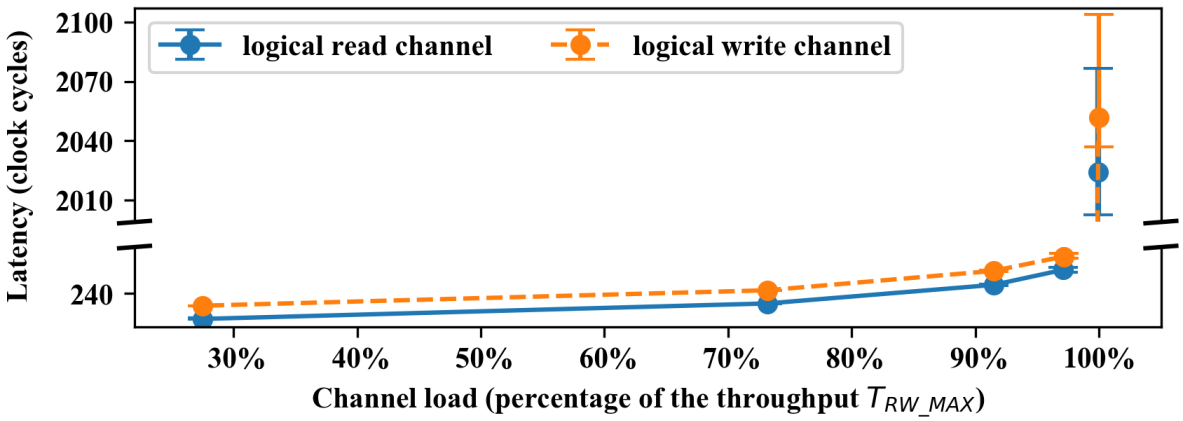


Fig.A.VI.54. Load-latency curve for RingNet with 5 1st level rings and 15 PGs connected at each ring, and 2 parallel rings used at the network root (F=5 × G=15, R=2).

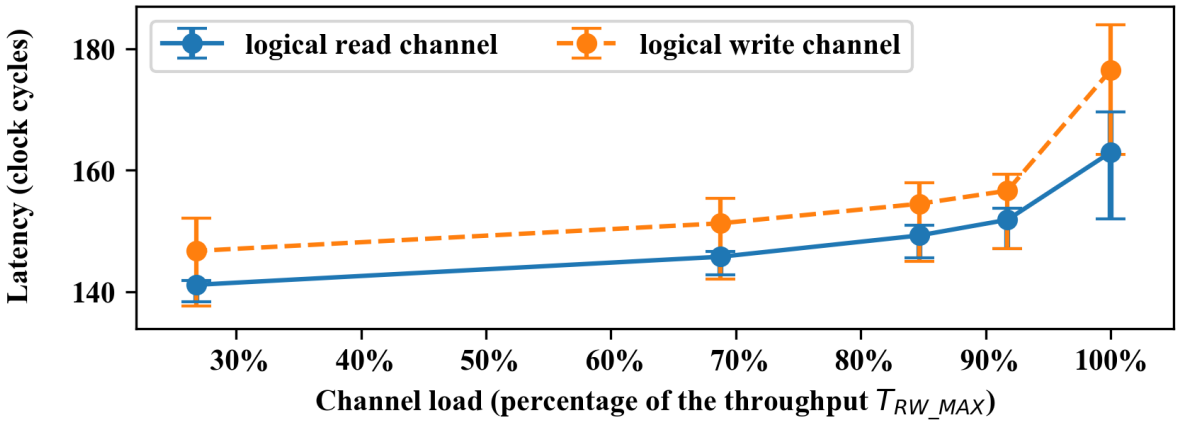


Fig.A.VI.55. Load-latency curve for RingNet with 3 1st level rings and 1 PG connected at each ring, and 3 parallel rings used at the network root (F=3 × G=1, R=3).

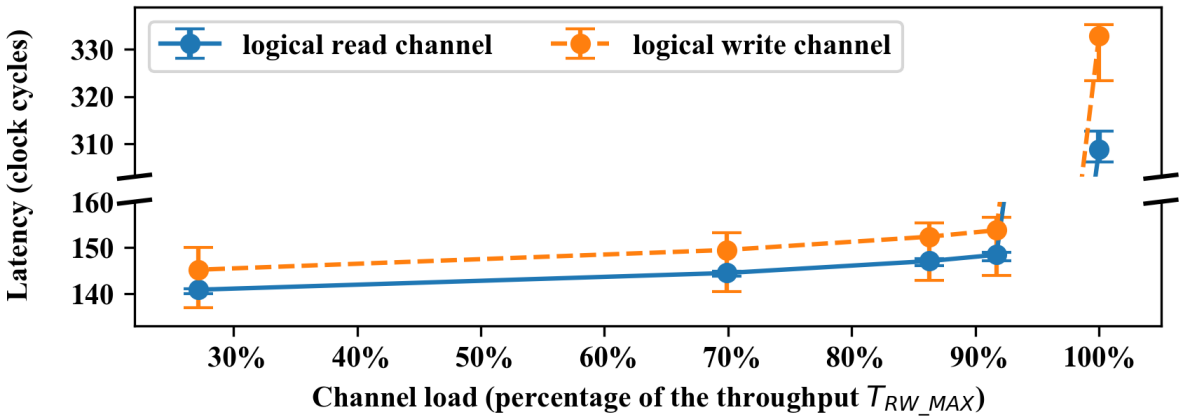


Fig.A.VI.56. Load-latency curve for RingNet with 4 1st level rings and 1 PG connected at each ring, and 3 parallel rings used at the network root (F=4 × G=1, R=3).

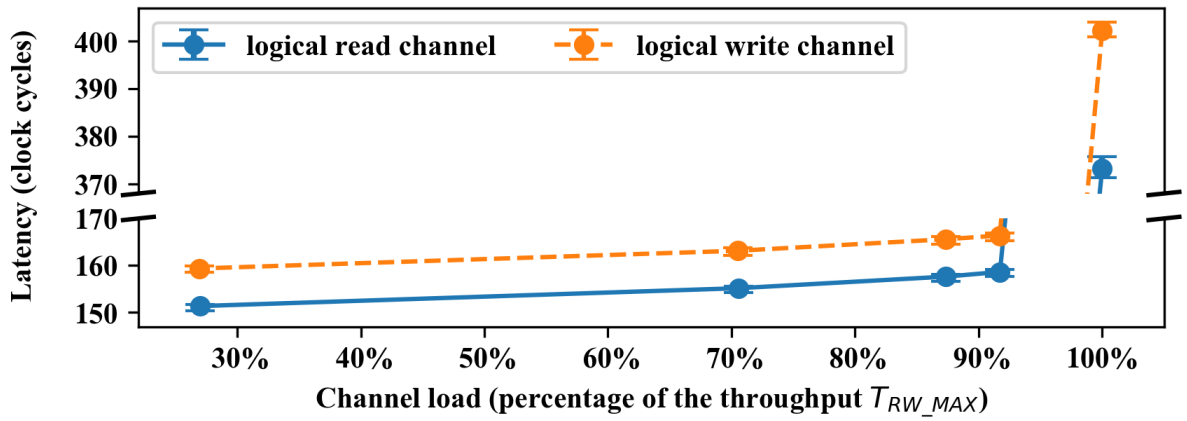


Fig.A.VI.57. Load-latency curve for RingNet with 5 1st level rings and 1 PG connected at each ring, and 3 parallel rings used at the network root (F=5 × G=1, R=3).

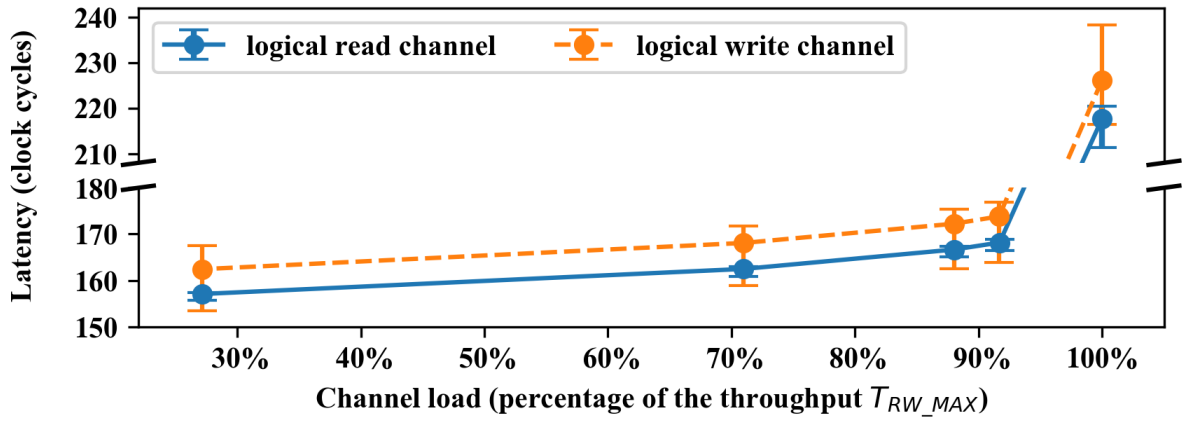


Fig.A.VI.58. Load-latency curve for RingNet with 3 1st level rings and 2 PGs connected at each ring, and 3 parallel rings used at the network root (F=3 × G=2, R=3).

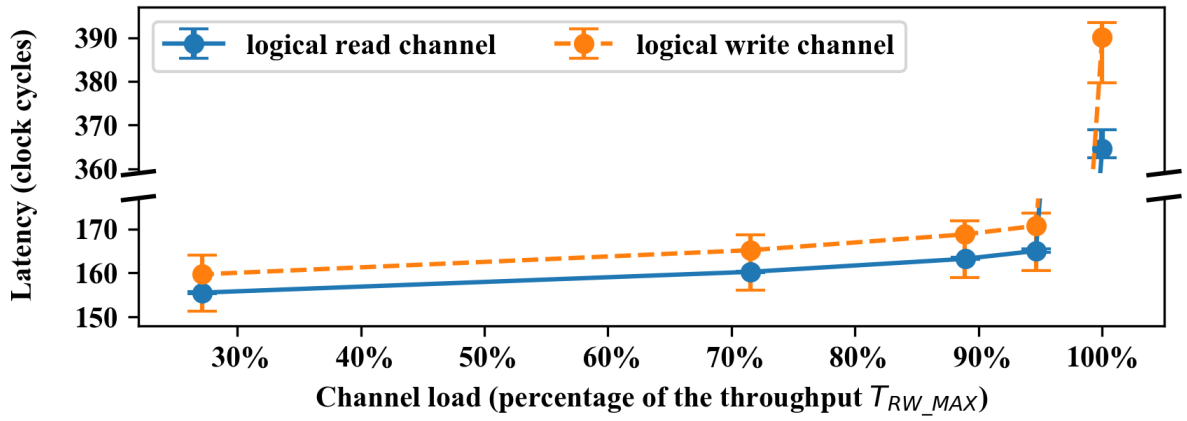


Fig.A.VI.59. Load-latency curve for RingNet with 4 1st level rings and 2 PGs connected at each ring, and 3 parallel rings used at the network root (F=4 × G=2, R=3).

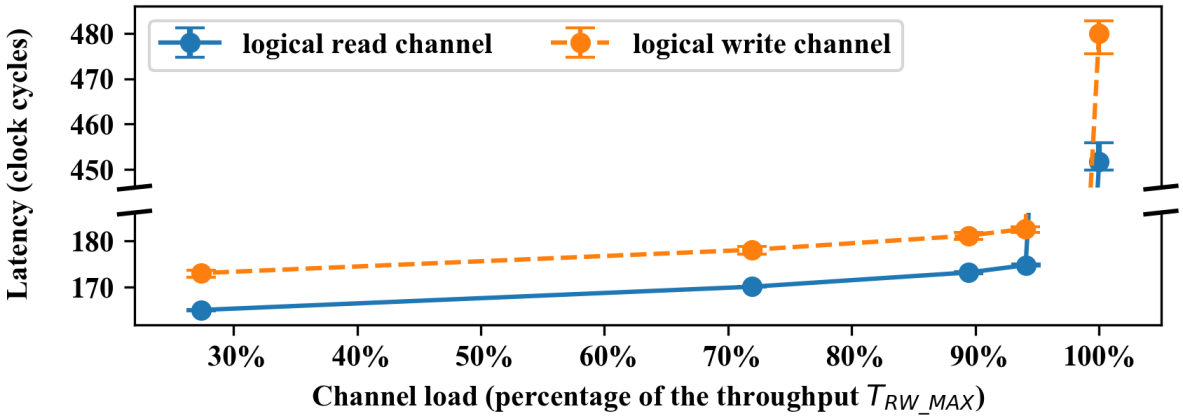


Fig.A.VI.60. Load-latency curve for RingNet with 5 1st level rings and 2 PGs connected at each ring, and 3 parallel rings used at the network root (F=5 × G=2, R=3).

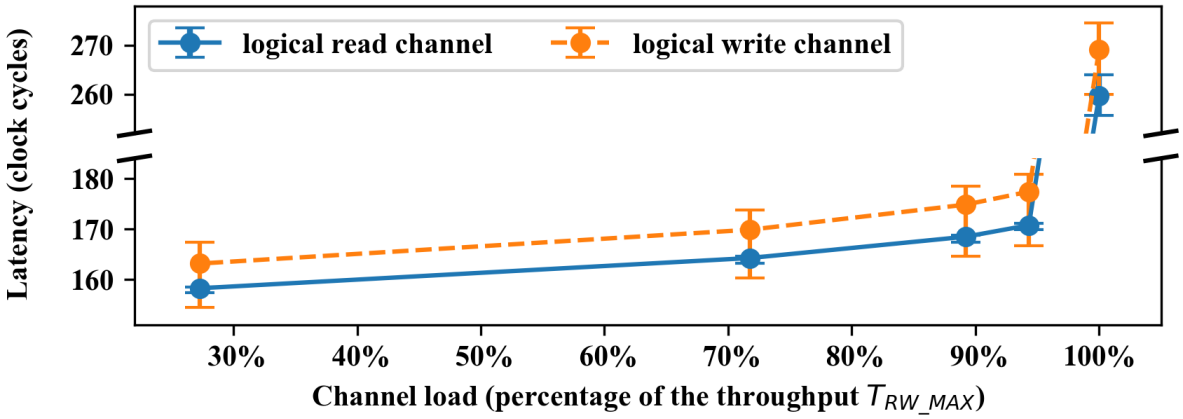


Fig.A.VI.61. Load-latency curve for RingNet with 3 1st level rings and 3 PGs connected at each ring, and 3 parallel rings used at the network root (F=3 × G=3, R=3).

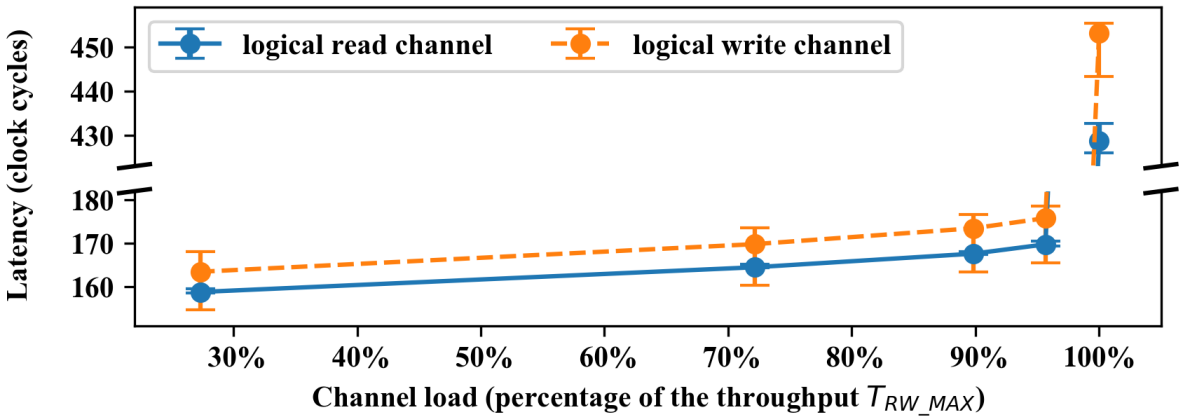


Fig.A.VI.62. Load-latency curve for RingNet with 4 1st level rings and 3 PGs connected at each ring, and 3 parallel rings used at the network root (F=4 × G=3, R=3).

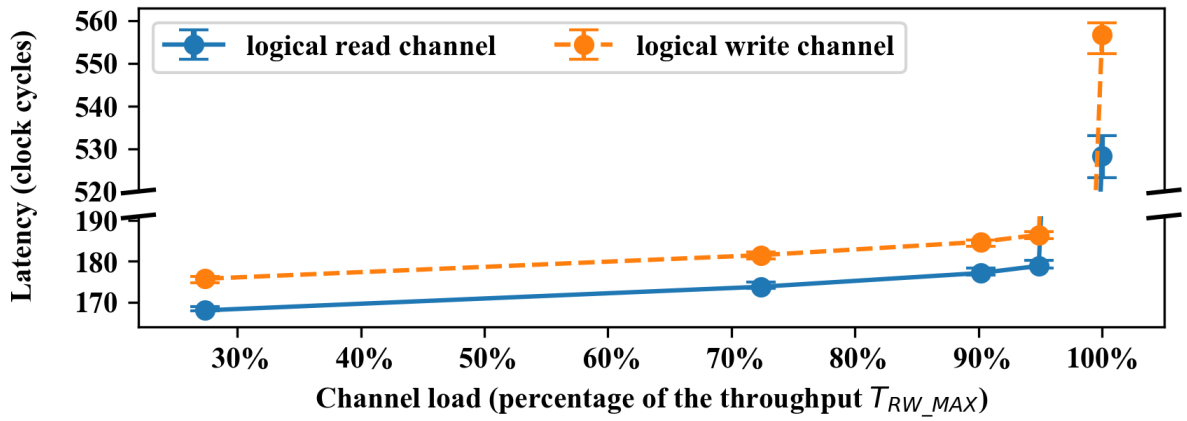


Fig.A.VI.63. Load-latency curve for RingNet with 5 1st level rings and 3 PGs connected at each ring, and 3 parallel rings used at the network root (F=5 × G=3, R=3).

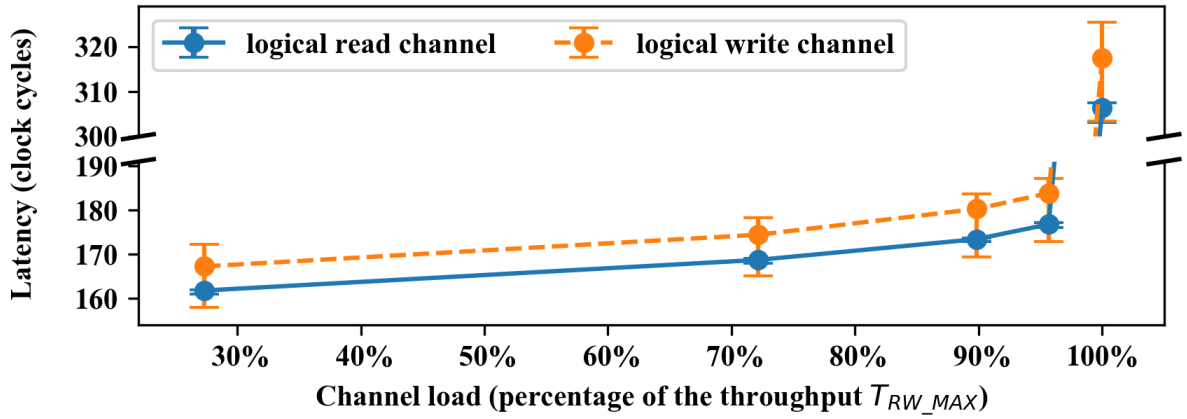


Fig.A.VI.64. Load-latency curve for RingNet with 3 1st level rings and 4 PGs connected at each ring, and 3 parallel rings used at the network root (F=3 × G=4, R=3).

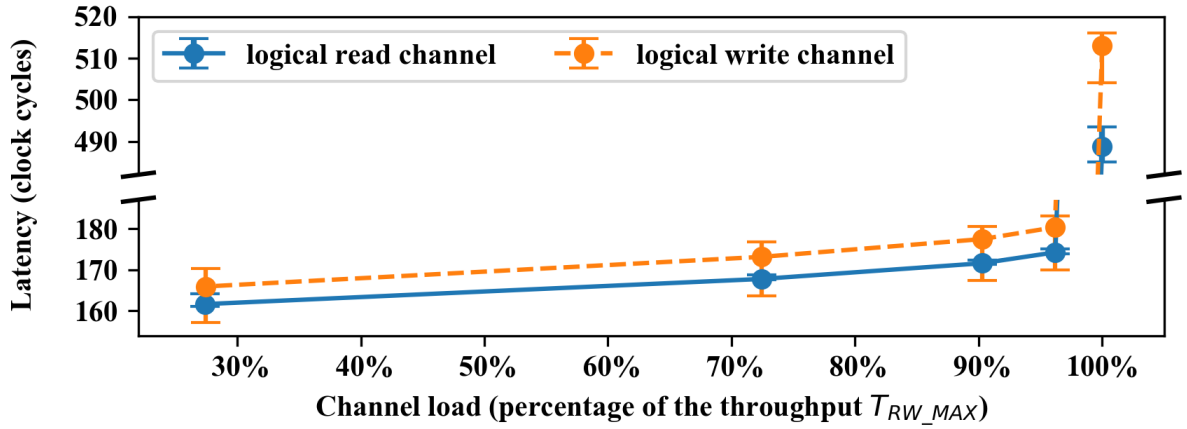


Fig.A.VI.65. Load-latency curve for RingNet with 4 1st level rings and 4 PGs connected at each ring, and 3 parallel rings used at the network root (F=4 × G=4, R=3).

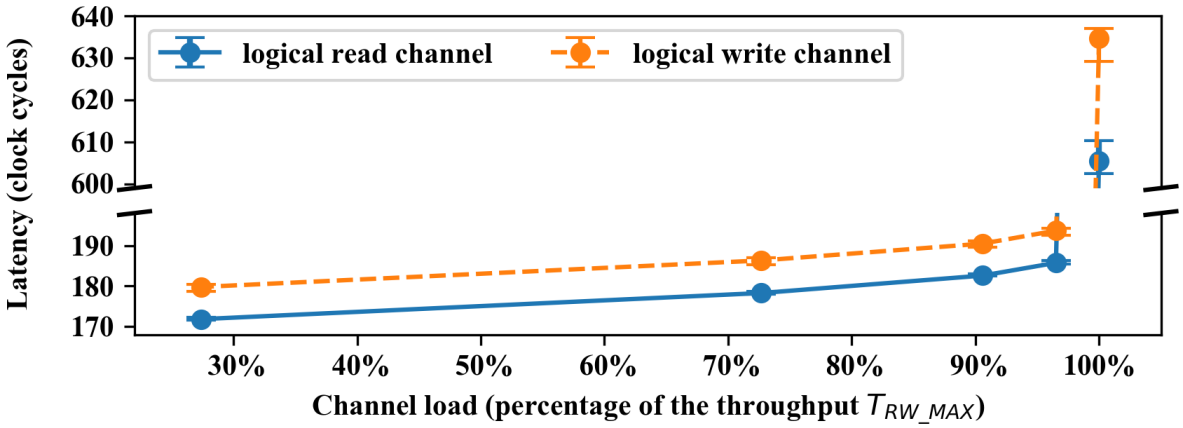


Fig.A.VI.66. Load-latency curve for RingNet with 5 1st level rings and 4 PGs connected at each ring, and 3 parallel rings used at the network root (F=5 × G=4, R=3).

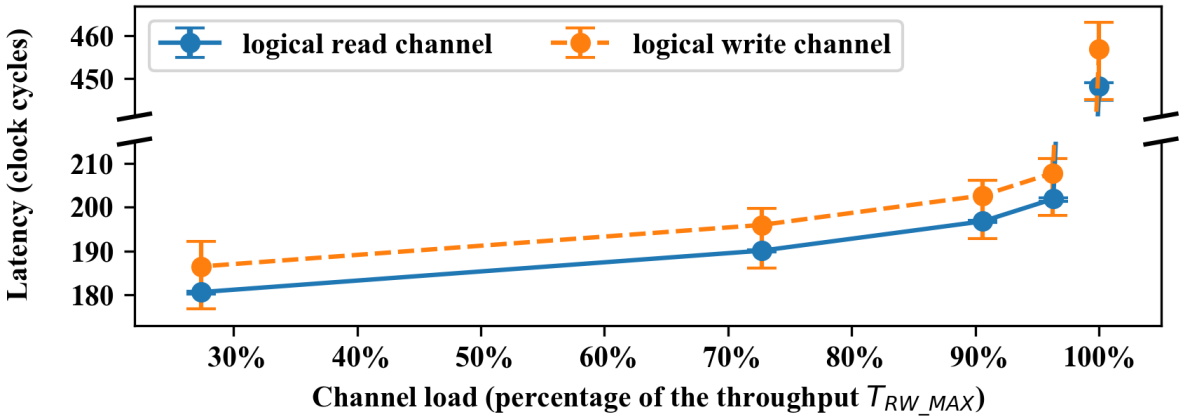


Fig.A.VI.67. Load-latency curve for RingNet with 3 1st level rings and 7 PGs connected at each ring, and 3 parallel rings used at the network root (F=3 × G=7, R=3).

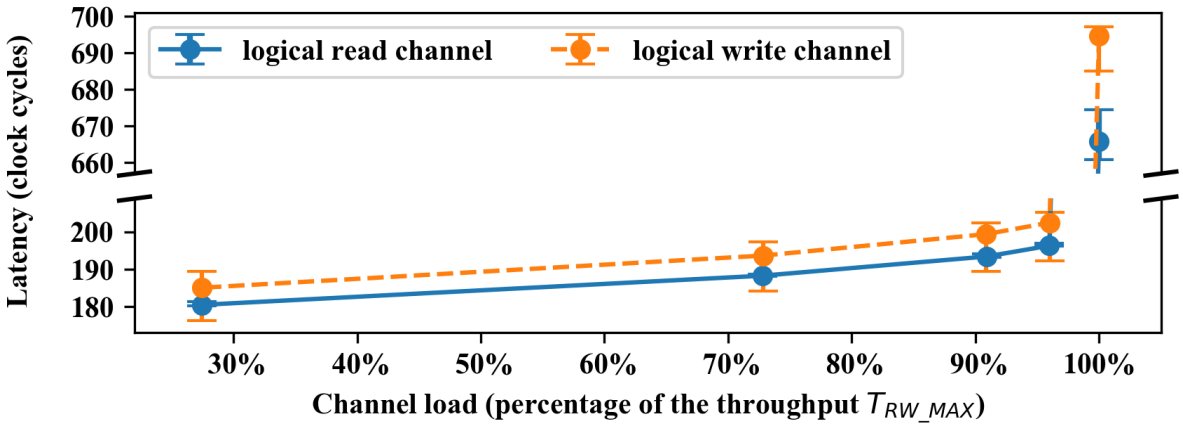


Fig.A.VI.68. Load-latency curve for RingNet with 4 1st level rings and 7 PGs connected at each ring, and 3 parallel rings used at the network root (F=4 × G=7, R=3).

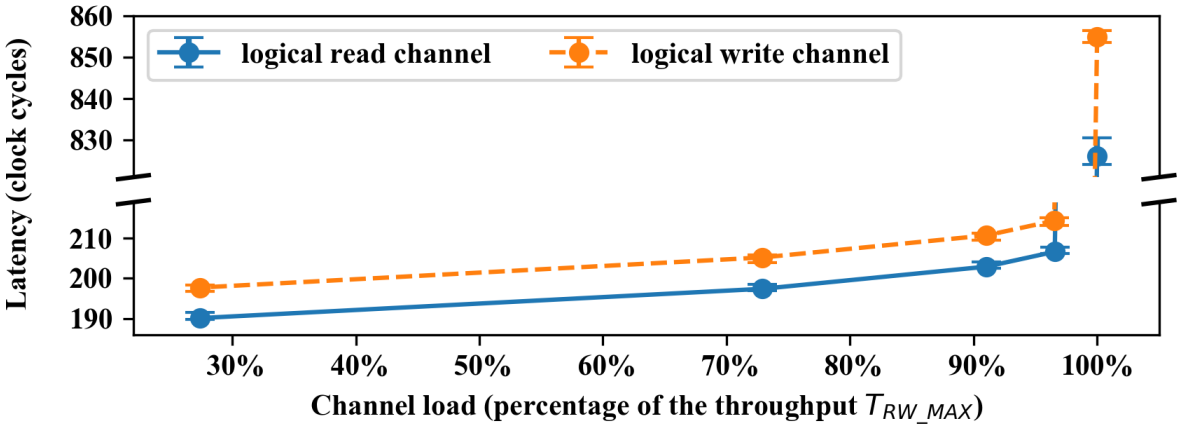


Fig.A.VI.69. Load-latency curve for RingNet with 5 1st level rings and 7 PGs connected at each ring, and 3 parallel rings used at the network root (F=5 × G=7, R=3).

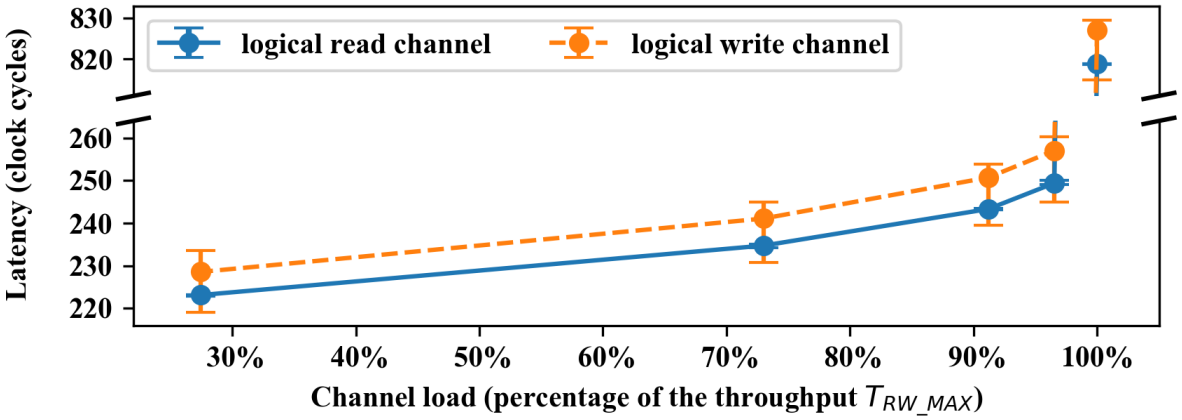


Fig.A.VI.70. Load-latency curve for RingNet with 3 1st level rings and 15 PGs connected at each ring, and 3 parallel rings used at the network root (F=3 × G=15, R=3).

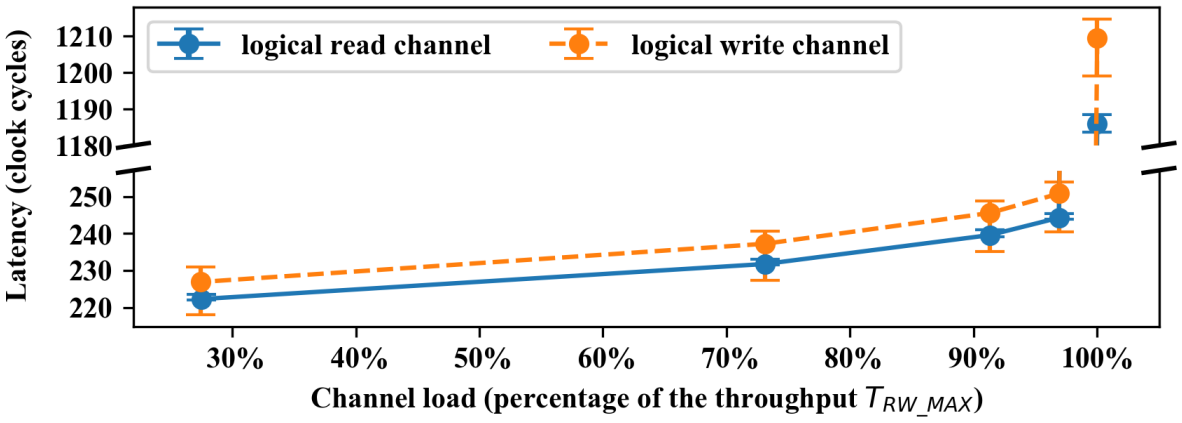


Fig.A.VI.71. Load-latency curve for RingNet with 4 1st level rings and 15 PGs connected at each ring, and 3 parallel rings used at the network root (F=4 × G=15, R=3).

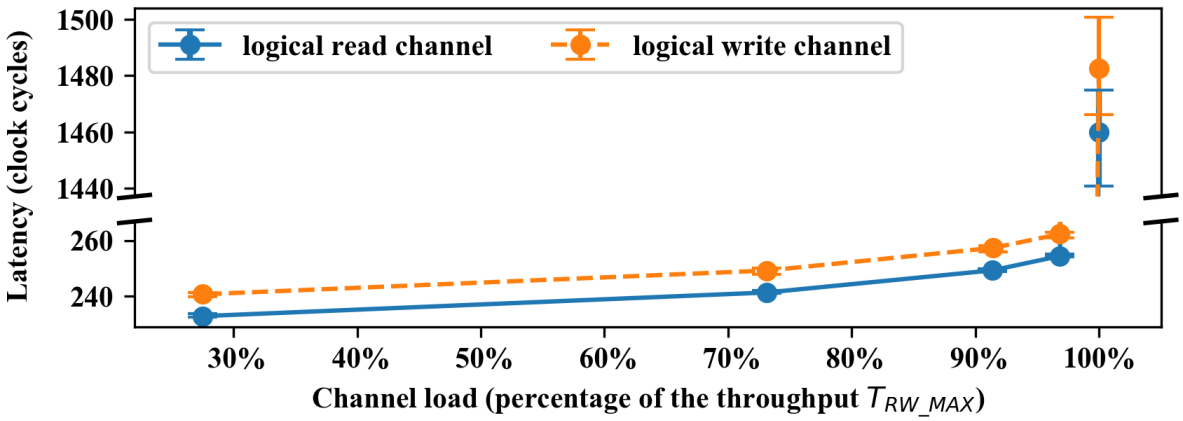


Fig.A.VI.72. Load-latency curve for RingNet with 5 1st level rings and 15 PGs connected at each ring, and 3 parallel rings used at the network root (F=5 × G=15, R=3).

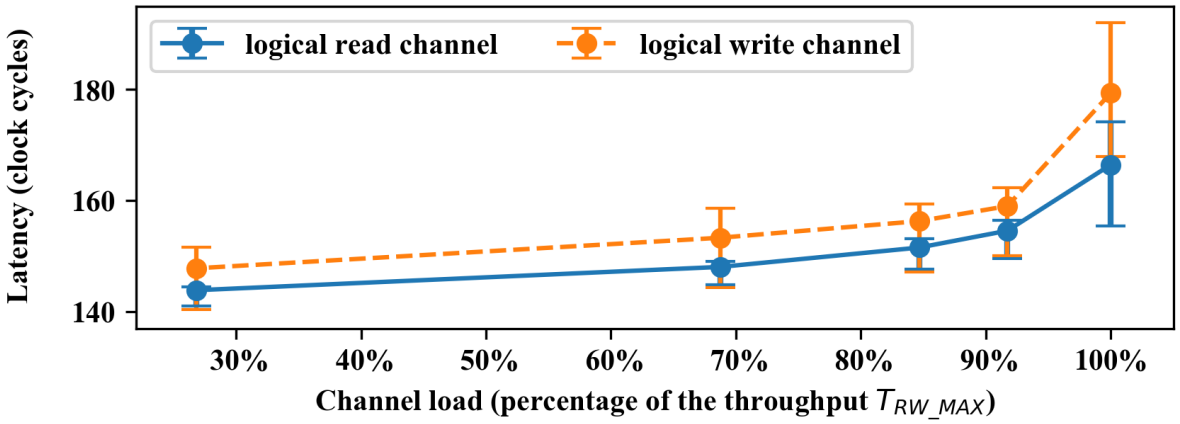


Fig.A.VI.73. Load-latency curve for RingNet with 4 1st level rings and 1 PG connected at each ring, and 4 parallel rings used at the network root (F=4 × G=1, R=4).

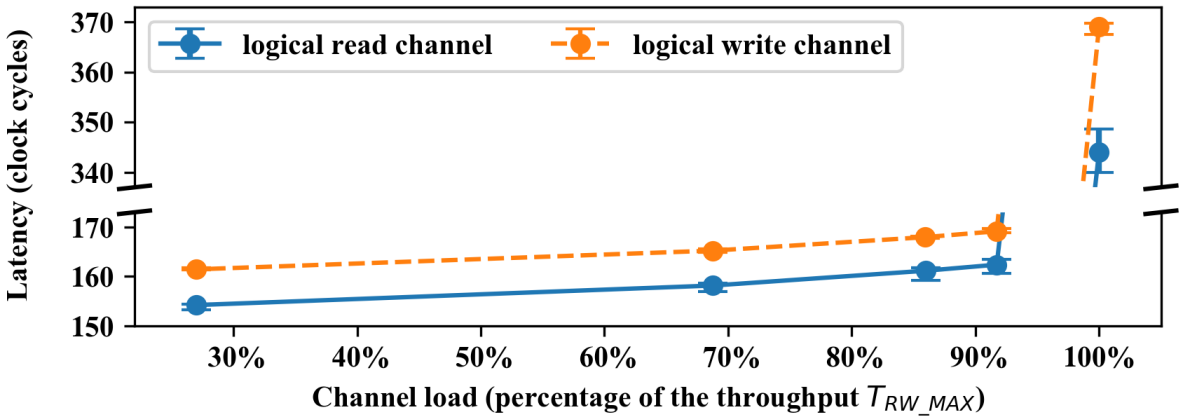


Fig.A.VI.74. Load-latency curve for RingNet with 5 1st level rings and 1 PG connected at each ring, and 4 parallel rings used at the network root (F=5 × G=1, R=4).

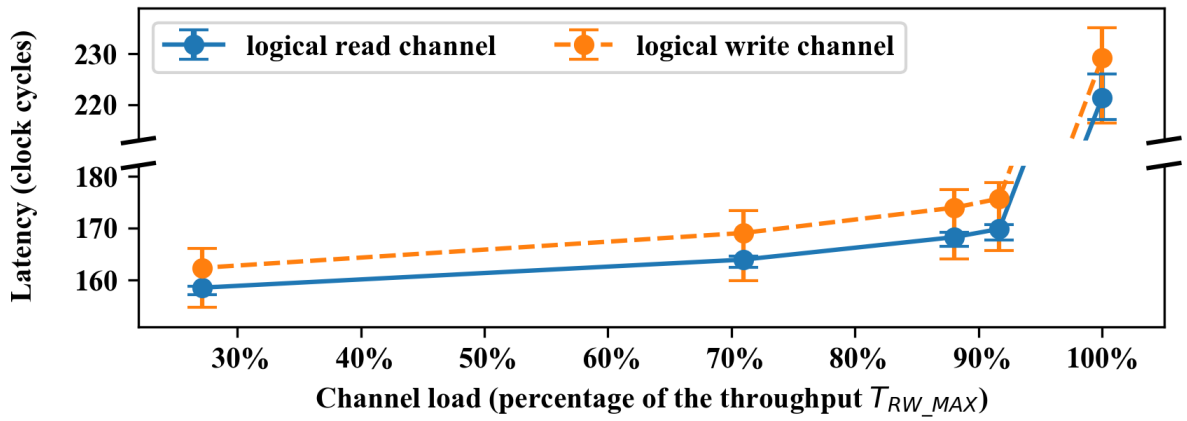


Fig.A.VI.75. Load-latency curve for RingNet with 4 1st level rings and 2 PGs connected at each ring, and 4 parallel rings used at the network root (F=4 × G=2, R=4).

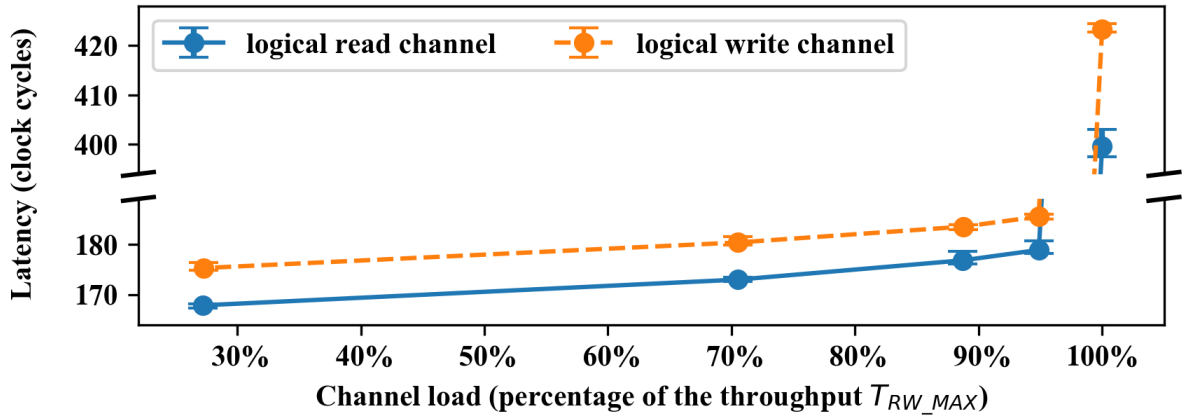


Fig.A.VI.76. Load-latency curve for RingNet with 5 1st level rings and 2 PGs connected at each ring, and 4 parallel rings used at the network root (F=5 × G=2, R=4).

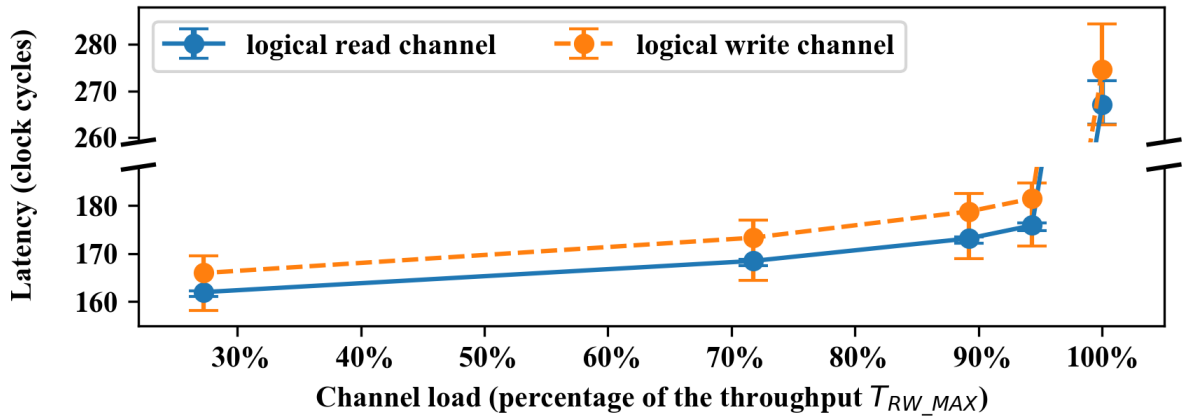


Fig.A.VI.77. Load-latency curve for RingNet with 4 1st level rings and 3 PGs connected at each ring, and 4 parallel rings used at the network root (F=4 × G=3, R=4).

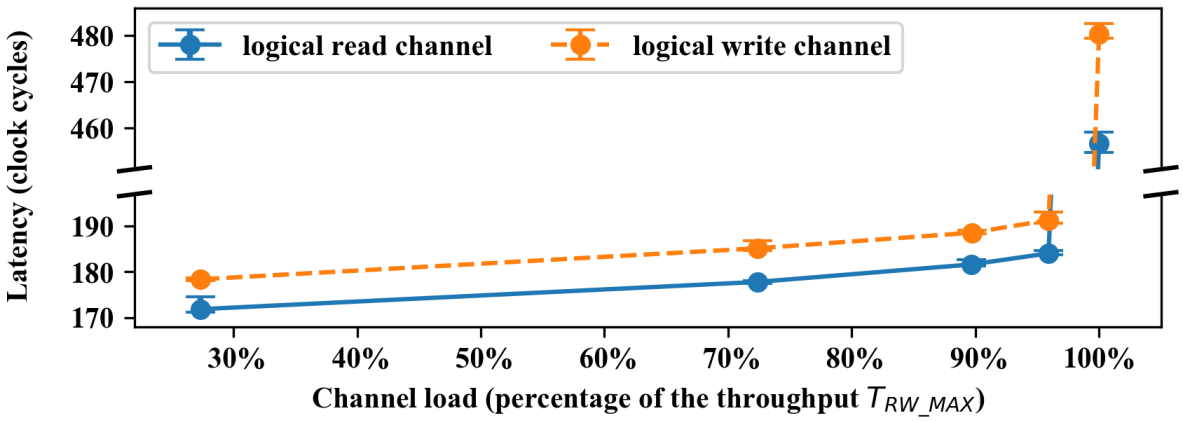


Fig.A.VI.78. Load-latency curve for RingNet with 5 1st level rings and 3 PGs connected at each ring, and 4 parallel rings used at the network root (F=5 × G=3, R=4).

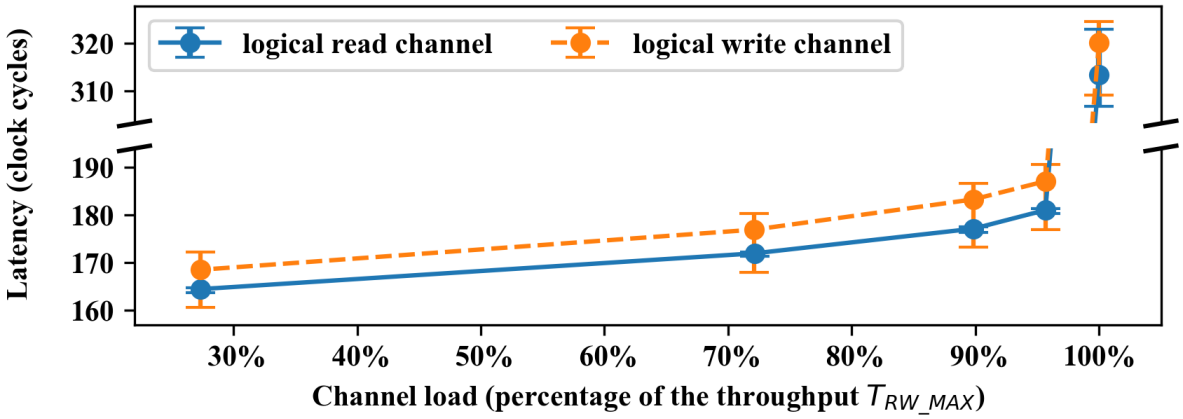


Fig.A.VI.79. Load-latency curve for RingNet with 4 1st level rings and 4 PGs connected at each ring, and 4 parallel rings used at the network root (F=4 × G=4, R=4).

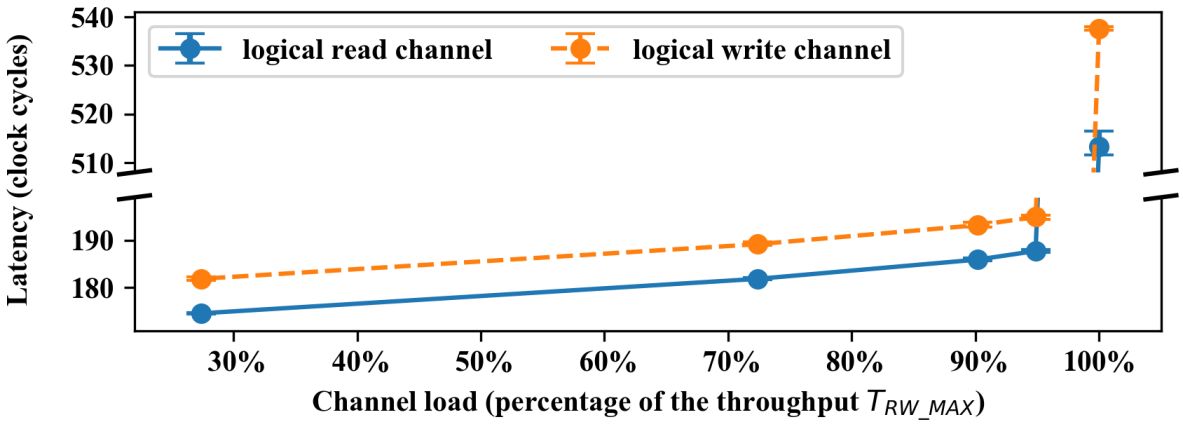


Fig.A.VI.80. Load-latency curve for RingNet with 5 1st level rings and 4 PGs connected at each ring, and 4 parallel rings used at the network root (F=5 × G=4, R=4).

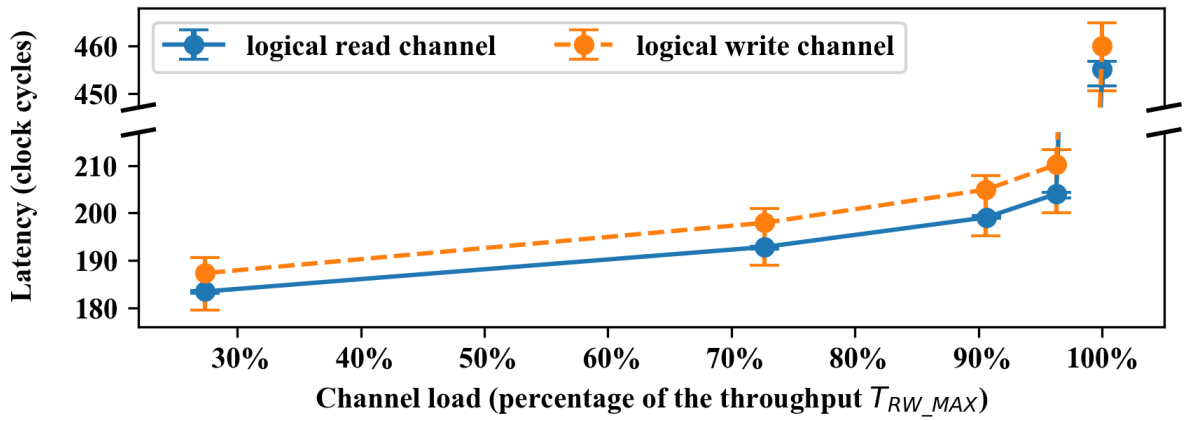


Fig.A.VI.81. Load-latency curve for RingNet with 4 1st level rings and 7 PGs connected at each ring, and 4 parallel rings used at the network root (F=4 × G=7, R=4).

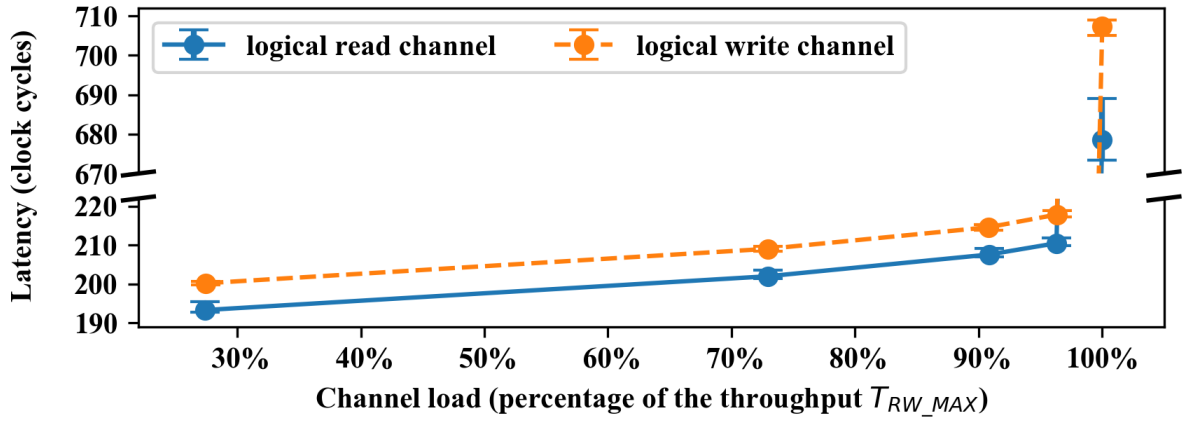


Fig.A.VI.82. Load-latency curve for RingNet with 5 1st level rings and 7 PGs connected at each ring, and 4 parallel rings used at the network root (F=5 × G=7, R=4).

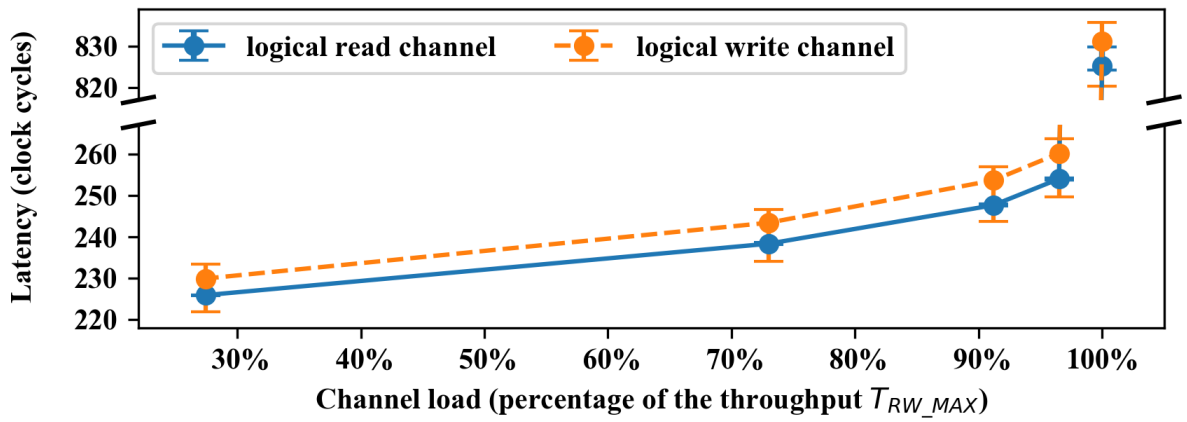


Fig.A.VI.83. Load-latency curve for RingNet with 4 1st level rings and 15 PGs connected at each ring, and 4 parallel rings used at the network root (F=4 × G=15, R=4).

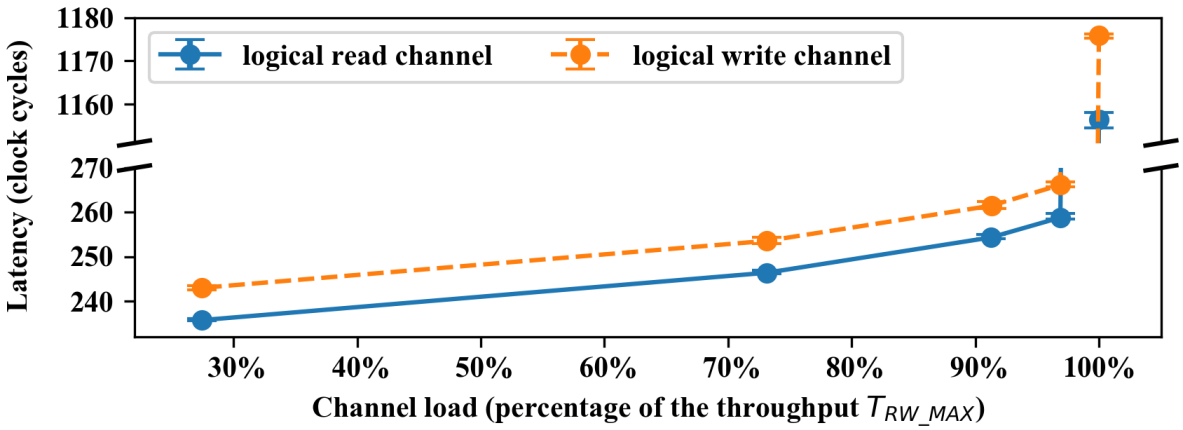


Fig.A.VI.84. Load-latency curve for RingNet with 5 1st level rings and 15 PGs connected at each ring, and 4 parallel rings used at the network root (F=5 × G=15, R=4).

VII. Simulation results of the test for RingNet access fairness

In Appendix VII, the results of the fairness experiment are presented. The results illustrate the conclusions of Section 7.3.

The parameters of the test are:

- *R*: Multiplication degree of the root level, i.e., the number of parallel rings used at the root level, set in the range of 1 – 4.
- *F*: The number of 1st level rings, set in the range of 2 – 5. A RingNet network with fewer 1st level rings (*F*) than there are parallel rings used at the root level (*R*) is not tested, as it cannot generate a 100% load.
- *G*: The number of packet generators (PGs) connected to a single 1st level ring, chosen from the set {1, 2, 3, 4, 7, 15}.
- Logical channel load. The aggregated load generated by all PGs is set in the range of 27% – 100% of the theoretical throughput T_{RW_MAX} (1). The same logical channel load is set for the read and write channels.

PEs are simulated using Packet Generators (PGs). For each individual PG, the author collects the values of the average latency L_{PG} expressed in clock cycles, and the values of the throughput T_{PG} expressed in bits per clock cycle, for both logical channels.

In tables, the values of L_{PG}, and the values of T_{PG}, calculated over all PGs ($\overline{L_{PG}}$ and $\overline{T_{PG}}$, respectively), are presented together with a standard deviation for those variables ($\sigma_{\overline{L_{PG}}}$ and $\sigma_{\overline{T_{PG}}}$, respectively).

Fairness test results for each combination of *R*, *F*, and *G* parameters are presented in separate tables. 83 tables are presented in the Appendix. In Table A.VII.i, the ranges of tables are presented that correspond to a given value of the multiplication degree of the root level (*R*) and a given number of PGs connected to a single 1st level ring (*G*). Each range presented in Table A.VII.i comprises tables with fairness test results for a given *R* and *G* value and for an increasing number of 1st level rings (*F*).

TABLE A.VII.i
SUMMARY FOR THE TABLES PRESENTED IN APPENDIX VII.

		Multiplication degree of the root level (R)			
		1	2	3	4
Number of PGs connected to a 1st level ring (G)	1	A.VII.1 – A.VII.4	A.VII.30 – A.VII.33	A.VII.54 – A.VII.56	A.VII.72 – A.VII.73
	2	A.VII.5 – A.VII.9	A.VII.34 – A.VII.37	A.VII.57 – A.VII.59	A.VII.74 – A.VII.75
	3	A.VII.10 – A.VII.14	A.VII.38 – A.VII.41	A.VII.60 – A.VII.62	A.VII.76 – A.VII.77
	4	A.VII.15 – A.VII.19	A.VII.42 – A.VII.45	A.VII.63 – A.VII.65	A.VII.78 – A.VII.79
	7	A.VII.20 – A.VII.24	A.VII.46 – A.VII.49	A.VII.66 – A.VII.68	A.VII.80 – A.VII.81
	15	A.VII.25 – A.VII.29	A.VII.50 – A.VII.53	A.VII.69 – A.VII.71	A.VII.82 – A.VII.83

In order to assess latency fairness, relative standard deviation is calculated according to the following formulas:

$$c_L = \frac{\sigma_{\overline{L_{PG}}}}{\overline{L_{PG}}} \qquad \text{(A.1)}$$

$$c_T = \frac{\sigma_{\overline{T_{PG}}}}{\overline{T_{PG}}} \qquad \text{(A.2)}$$

High values of c_L and c_T mean that the standard deviation is relatively big, in comparison with the average value of a parameter. The highest observed values of c_L are reported in Table A.VII.ii together with the network configuration they are observed for. The source table is also specified. Cases that are reported in Table A.VII.ii are marked in red in their source tables for easy identification.

TABLE A.VII.ii
THE HIGHEST OBSERVED RELATIVE STANDARD DEVIATIONS FOR THE AVERAGE LATENCY CALCULATED OVER ALL PGS.

	Low load (i.e., 26% – 27% of the theoretical throughput T_{RW_MAX})		Moderate load (i.e., 70% – 72% of the theoretical throughput T_{RW_MAX})		High load (i.e., 88% – 91% of the theoretical throughput T_{RW_MAX})		Near saturation (i.e., 92% – 97% of the theoretical throughput T_{RW_MAX})		Saturation (i.e., 100% of the theoretical throughput T_{RW_MAX})	
	c_L	Network configuration (table)	c_L	Network configuration (table)	c_L	Network configuration (table)	c_L	Network configuration (table)	c_L	Network configuration (table)
Read	**4.2%**	F=2 × G=1, R=1 (A.VII.1)	**3.2%**	F=2 × G=1, R=1 (A.VII.1)	**2.8%**	F=2 × G=1, R=1 (A.VII.1)	**2.9%**	F=2 × G=1, R=2 (A.VII.30)	**6.8%**	F=4 × G=1, R=4 (A.VII.72)
Write	**4.1%**	F=2 × G=1, R=1 (A.VII.1)	**3.4%**	F=2 × G=3, R=1 (A.VII.11)	**3.6%**	F=2 × G=3, R=1 (A.VII.11)	**3.6%**	F=2 × G=3, R=1 (A.VII.11)	**7.4%**	F=4 × G=1, R=4 (A.VII.72)

For all tested network loads and for both logical channels, the highest values of c_L are observed for small networks, with 6 PGs at most. For those cases, $\overline{\sigma_{L_{PG}}}$ is below 6 clock cycles if the network is not in saturation, or 13 cycles for a network in saturation. Differences in latency of this magnitude are negligible and latency throughput is proved for the tested RingNet configurations.

The highest values of c_T are reported in Table A.VII.iii together with the network configuration they are observed for. The source table is also specified. Cases that are reported in Table A.VII.iii are marked in red in their source tables for easy identification.

TABLE A.VII.iii
THE HIGHEST OBSERVED RELATIVE STANDARD DEVIATIONS FOR THE AVERAGE THROUGHPUT CALCULATED OVER ALL PGS.

	Low load (i.e., 26% – 27% of the theoretical throughput T_{RW_MAX})		Moderate load (i.e., 70% – 72% of the theoretical throughput T_{RW_MAX})		High load (i.e., 88% – 91% of the theoretical throughput T_{RW_MAX})		Near saturation (i.e., 92% – 97% of the theoretical throughput T_{RW_MAX})		Saturation (i.e., 100% of the theoretical throughput T_{RW_MAX})	
	c_T	Network configuration (table)	c_T	Network configuration (table)	c_T	Network configuration (table)	c_T	Network configuration (table)	c_T	Network configuration (table)
Read	**1.9%**	F=4 × G=15, R=1 (A.VII.28)	**1.2%**	F=5 × G=15, R=1 (A.VII.29)	**1.0%**	F=5 × G=15, R=1 (A.VII.29)	**1.0%**	F=5 × G=15, R=1 (A.VII.29)	**0.5%**	F=4 × G=15, R=1 (A.VII.28)
Write	**2.0%**	F=4 × G=15, R=1 (A.VII.28)	**1.1%**	F=5 × G=15, R=1 (A.VII.29)	**1.1%**	F=5 × G=15, R=1 (A.VII.29)	**1.2%**	F=5 × G=15, R=1 (A.VII.29)	**0.5%**	F=4 × G=15, R=1 (A.VII.28)

The highest values of c_T are observed for the largest network configurations tested, i.e., with 60 and 75 connected PGs. Nevertheless, the highest c_T value of just 2% is reported. The author concludes that throughput fairness is proved for the tested RingNet configurations.

TABLE A.VII.1
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=1, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	110	4.65	117	4.75	6.3	0.015	6.3	0.010
71%	111	3.60	118	3.55	16.5	0.005	16.5	0.015
88%	113	3.15	119	3.10	20.4	0.025	20.5	0.010
92%	113	2.95	120	2.90	21.3	0.010	21.3	0.005
100%	210	5.50	217	5.50	23.3	0.000	23.3	0.000

Results marked in red correspond to the highest observed values of c_L, which are reported in Table A.VII.ii.

TABLE A.VII.2
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=1, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	123	4.57	123	4.41	4.2	0.000	4.2	0.012
72%	126	3.50	131	3.15	11.1	0.052	11.1	0.029
89%	127	3.20	134	3.19	13.8	0.041	13.8	0.014
94%	129	3.06	135	3.11	14.6	0.012	14.6	0.005
100%	318	4.68	325	4.68	15.5	0.000	15.5	0.005

TABLE A.VII.3
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=1, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	122	4.57	121	4.61	3.2	0.008	3.2	0.005
72%	125	3.50	130	3.48	8.4	0.011	8.4	0.015
90%	128	3.26	134	3.18	10.5	0.021	10.5	0.019
96%	130	3.10	136	3.11	11.1	0.024	11.1	0.008
100%	417	5.27	425	4.13	11.6	0.000	11.6	0.000

TABLE A.VII.4
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=1, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	132	4.56	139	4.64	2.5	0.014	2.5	0.012
72%	136	3.55	142	3.71	6.7	0.022	6.7	0.010
90%	138	3.31	145	3.18	8.4	0.020	8.4	0.019
95%	140	3.05	147	3.17	8.8	0.016	8.8	0.020
100%	514	5.94	521	5.90	9.3	0.000	9.3	0.000

TABLE A.VII.5
TRAFFIC STATISTICS FOR RINGNET WITH ONE 1ST LEVEL RING AND 2 PGS CONNECTED AT THE RING, AND ONE RING USED AT THE NETWORK ROOT (F=1 × G=2, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	115	0.35	114	0.45	6.3	0.015	6.3	0.010
71%	116	0.25	122	0.15	16.5	0.005	16.5	0.015
88%	118	0.20	124	0.10	20.4	0.025	20.5	0.010
92%	118	0.20	125	0.15	21.3	0.010	21.3	0.005
100%	166	0.50	172	0.50	23.3	0.000	23.3	0.000

TABLE A.VII.6
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=2, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	127	0.89	134	0.91	3.2	0.008	3.2	0.005
72%	130	1.98	137	1.95	8.4	0.011	8.4	0.015
90%	132	2.44	139	2.50	10.5	0.021	10.5	0.019
96%	134	2.64	141	2.63	11.1	0.024	11.1	0.008
100%	297	4.46	304	0.50	11.6	0.004	11.6	0.000

TABLE A.VII.7
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=2, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	138	0.97	138	1.19	2.1	0.007	2.1	0.005
73%	142	2.09	148	2.53	5.6	0.022	5.6	0.011
90%	145	2.60	152	2.70	7.0	0.017	7.0	0.022
96%	148	2.84	155	2.96	7.4	0.027	7.4	0.021
100%	459	6.89	469	3.68	7.8	0.000	7.8	0.000

TABLE A.VII.8
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=2, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	138	0.88	138	0.91	1.6	0.007	1.6	0.007
73%	143	2.28	149	2.49	4.2	0.012	4.2	0.015
91%	147	2.74	154	2.75	5.3	0.014	5.3	0.011
97%	151	2.84	158	3.03	5.6	0.014	5.6	0.015
100%	602	21.06	611	20.46	5.8	0.003	5.8	0.000

TABLE A.VII.9
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=2, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	147	3.57	154	3.77	1.3	0.008	1.3	0.005
73%	153	2.85	160	3.15	3.4	0.012	3.4	0.008
91%	158	3.08	165	3.15	4.2	0.021	4.2	0.011
97%	163	3.03	169	3.19	4.5	0.012	4.5	0.014
100%	750	19.04	762	15.14	4.7	0.000	4.7	0.004

TABLE A.VII.10
TRAFFIC STATISTICS FOR RINGNET WITH ONE 1ST LEVEL RING AND 3 PGS CONNECTED AT THE RING, AND ONE RING USED AT THE NETWORK ROOT (F=1 × G=3, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	115	0.78	114	0.67	4.2	0.012	4.2	0.009
72%	117	0.66	123	0.45	11.1	0.012	11.1	0.012
89%	119	0.53	126	0.42	13.8	0.009	13.8	0.028
94%	120	0.45	127	0.37	14.6	0.005	14.6	0.016
100%	209	0.82	216	0.82	15.5	0.005	15.5	0.005

TABLE A.VII.11
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=3, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	132	4.72	139	5.00	2.1	0.005	2.1	0.013
73%	136	3.44	143	4.84	5.6	0.012	5.6	0.015
90%	139	3.19	146	5.22	7.0	0.007	7.0	0.014
96%	142	2.70	149	5.41	7.4	0.020	7.4	0.021
100%	390	5.55	397	3.40	7.8	0.000	7.8	0.000

Results marked in red correspond to the highest observed values of c_L, which are reported in Table A.VII.ii.

TABLE A.VII.12
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=3, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	141	4.45	141	4.44	1.4	0.008	1.4	0.008
73%	147	3.52	153	3.84	3.8	0.010	3.8	0.011
91%	151	3.20	158	3.91	4.7	0.016	4.7	0.015
96%	155	3.05	162	3.93	5.0	0.009	5.0	0.016
100%	594	13.56	601	9.40	5.2	0.000	5.2	0.000

TABLE A.VII.13
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=3, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	141	4.18	140	4.16	1.1	0.008	1.1	0.009
73%	147	3.53	153	3.78	2.8	0.012	2.8	0.013
91%	153	3.37	159	3.45	3.5	0.015	3.5	0.009
96%	157	3.34	163	3.62	3.7	0.014	3.7	0.018
100%	780	14.11	790	9.92	3.9	0.000	3.9	0.000

TABLE A.VII.14
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=3, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	151	3.89	158	3.92	0.9	0.008	0.9	0.008
73%	158	3.24	165	3.60	2.3	0.011	2.3	0.011
91%	165	3.42	172	3.44	2.8	0.009	2.8	0.013
97%	169	3.00	177	3.52	3.0	0.016	3.0	0.010
100%	980	28.97	979	6.81	3.1	0.004	3.1	0.002

TABLE A.VII.15
TRAFFIC STATISTICS FOR RINGNET WITH ONE 1ST LEVEL RING AND 4 PGS CONNECTED AT THE RING, AND ONE RING USED AT THE NETWORK ROOT (F=1 × G=4, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	115	0.94	114	0.95	3.2	0.011	3.2	0.013
72%	118	0.82	124	0.65	8.4	0.022	8.4	0.018
90%	120	0.67	127	0.78	10.5	0.013	10.5	0.011
96%	122	0.66	129	0.67	11.1	0.018	11.1	0.028
100%	252	1.12	260	1.12	11.6	0.000	11.6	0.000

TABLE A.VII.16
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=4, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	132	5.00	138	4.95	1.6	0.008	1.6	0.009
73%	137	3.64	144	3.51	4.2	0.015	4.2	0.012
91%	141	3.04	147	3.08	5.3	0.017	5.3	0.020
97%	145	2.82	152	2.45	5.6	0.019	5.6	0.011
100%	478	5.61	485	5.63	5.8	0.000	5.8	0.000

TABLE A.VII.17
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=4, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	144	4.68	144	4.29	1.1	0.008	1.1	0.009
73%	151	3.54	157	3.65	2.8	0.012	2.8	0.013
91%	157	3.25	164	3.07	3.5	0.015	3.5	0.009
96%	161	3.34	167	2.89	3.7	0.014	3.7	0.018
100%	729	21.70	736	19.26	3.9	0.000	3.9	0.000

TABLE A.VII.18
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=4, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	143	4.89	143	4.81	0.8	0.007	0.8	0.007
73%	150	3.77	156	3.71	2.1	0.009	2.1	0.012
91%	157	3.35	164	3.33	2.7	0.012	2.7	0.008
97%	163	3.36	170	3.06	2.8	0.009	2.8	0.010
100%	958	19.52	969	22.79	2.9	0.000	2.9	0.000

TABLE A.VII.19
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=4, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
28%	153	4.60	160	4.78	0.6	0.004	0.6	0.005
73%	160	3.75	168	3.69	1.7	0.010	1.7	0.007
91%	168	3.56	176	3.49	2.1	0.007	2.1	0.007
97%	175	3.08	183	3.21	2.3	0.008	2.3	0.009
100%	1193	22.65	1201	40.66	2.3	0.002	2.3	0.003

TABLE A.VII.20
TRAFFIC STATISTICS FOR RINGNET WITH ONE 1ST LEVEL RING AND 7 PGS CONNECTED AT THE RING, AND ONE RING USED AT THE NETWORK ROOT (F=1 × G=7, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	136	1.86	135	1.92	1.8	0.011	1.8	0.007
73%	140	1.72	146	1.64	4.8	0.016	4.8	0.006
91%	144	1.55	151	1.53	6.0	0.021	6.0	0.018
96%	147	1.56	155	1.49	6.4	0.021	6.4	0.012
100%	394	2.00	401	2.00	6.6	0.000	6.6	0.000

TABLE A.VII.21
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=7, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	152	5.04	159	5.26	0.9	0.007	0.9	0.006
73%	160	3.65	167	3.59	2.4	0.012	2.4	0.006
91%	169	2.14	175	2.36	3.0	0.010	3.0	0.010
97%	181	3.18	187	3.19	3.2	0.011	3.2	0.012
100%	749	16.24	756	17.24	3.3	0.005	3.3	0.005

TABLE A.VII.22
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=7, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	162	4.91	162	4.92	0.6	0.004	0.6	0.006
73%	170	3.94	175	4.17	1.6	0.008	1.6	0.011
91%	178	3.26	185	3.49	2.0	0.009	2.0	0.010
97%	185	3.41	193	3.37	2.2	0.011	2.2	0.009
100%	1127	3.42	1134	3.38	2.2	0.000	2.2	0.002

TABLE A.VII.23
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=7, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
28%	161	4.84	161	4.51	0.5	0.005	0.5	0.006
73%	170	3.74	175	3.98	1.2	0.008	1.2	0.009
91%	179	3.67	186	3.72	1.5	0.010	1.5	0.010
97%	189	3.75	196	3.67	1.6	0.009	1.6	0.008
100%	1516	15.14	1523	18.88	1.7	0.000	1.7	0.000

TABLE A.VII.24
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=7, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	171	4.24	178	4.43	0.4	0.006	0.4	0.006
73%	181	3.79	188	3.94	1.0	0.007	1.0	0.008
91%	192	4.00	200	3.82	1.2	0.008	1.2	0.009
97%	201	3.68	210	3.75	1.3	0.007	1.3	0.008
100%	1877	33.38	1879	25.62	1.3	0.000	1.3	0.000

TABLE A.VII.25
TRAFFIC STATISTICS FOR RINGNET WITH ONE 1ST LEVEL RING AND 15 PGS CONNECTED AT THE RING, AND ONE RING USED AT THE NETWORK ROOT (F=1 × G=15, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	176	4.13	176	4.06	0.9	0.008	0.9	0.008
73%	183	3.84	189	3.87	2.3	0.011	2.3	0.011
91%	189	3.90	196	3.95	2.8	0.009	2.8	0.013
97%	194	3.70	202	3.76	3.0	0.016	3.0	0.010
100%	764	4.32	771	4.32	3.1	0.000	3.1	0.000

TABLE A.VII.26
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=2 × G=15, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
28%	192	6.34	199	6.37	0.4	0.005	0.4	0.005
73%	203	5.27	210	5.08	1.1	0.009	1.1	0.006
91%	214	4.64	222	4.91	1.4	0.008	1.4	0.008
97%	225	4.12	232	4.43	1.5	0.010	1.5	0.008
100%	1528	6.99	1536	6.99	1.5	0.000	1.5	0.000

TABLE A.VII.27
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=3 × G=15, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	205	6.14	205	6.14	0.3	0.005	0.3	0.005
73%	216	5.57	221	5.42	0.8	0.007	0.8	0.008
92%	230	5.00	237	4.53	0.9	0.007	0.9	0.006
97%	241	4.96	249	5.07	1.0	0.007	1.0	0.009
100%	2300	6.75	2307	6.75	1.0	0.004	1.0	0.004

TABLE A.VII.28
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=4 × G=15, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	204	6.40	204	6.27	0.2	0.004	0.2	0.004
73%	214	5.65	220	5.36	0.6	0.006	0.6	0.006
92%	231	5.62	237	5.47	0.7	0.006	0.7	0.007
97%	245	5.33	252	5.59	0.8	0.007	0.8	0.006
100%	3058	6.99	3064	6.99	0.8	0.004	0.8	0.004

Results marked in red correspond to the highest observed values of c_T, which are reported in Table A.VII.iii.

TABLE A.VII.29
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND ONE RING USED AT THE NETWORK ROOT (F=5 × G=15, R=1).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	213	6.44	220	6.42	0.2	0.003	0.2	0.003
73%	224	5.78	231	5.39	0.5	0.005	0.5	0.005
92%	242	5.37	249	5.74	0.6	0.006	0.6	0.006
97%	258	5.01	266	5.57	0.6	0.006	0.6	0.007
100%	3672	40.46	3712	33.86	0.6	0.000	0.6	0.000

Results marked in red correspond to the highest observed values of c_T, which are reported in Table A.VII.iii.

TABLE A.VII.30
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=1, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	127	0.70	134	0.60	12.5	0.020	12.5	0.000
69%	129	2.55	136	1.95	32.0	0.070	32.0	0.005
85%	130	3.55	138	2.60	39.4	0.010	39.4	0.010
92%	132	3.75	140	2.60	42.7	0.085	42.7	0.015
100%	124	3.50	132	3.50	46.5	0.005	46.5	0.000

Results marked in red correspond to the highest observed values of c_L, which are reported in Table A.VII.ii.

TABLE A.VII.31
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=1, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	134	4.54	134	4.44	8.4	0.012	8.4	0.019
69%	137	3.53	142	3.23	21.4	0.033	21.3	0.045
87%	138	3.18	145	3.11	26.9	0.014	27.0	0.014
92%	139	3.02	146	2.91	28.5	0.012	28.4	0.000
100%	272	5.21	296	5.19	31.0	0.000	31.0	0.000

TABLE A.VII.32
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=1, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	136	4.12	135	3.93	6.3	0.011	6.3	0.008
71%	138	3.44	144	3.01	16.5	0.015	16.5	0.012
88%	140	3.22	147	3.03	20.5	0.015	20.5	0.008
92%	141	3.02	148	2.94	21.3	0.031	21.3	0.015
100%	353	5.00	379	4.76	23.3	0.000	23.3	0.000

TABLE A.VII.33
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=1, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	147	3.82	154	3.84	5.1	0.014	5.1	0.011
71%	150	3.25	158	3.22	13.1	0.034	13.1	0.009
89%	153	2.87	160	3.13	16.5	0.020	16.5	0.011
95%	155	2.91	162	3.05	17.7	0.021	17.7	0.030
100%	436	5.42	469	4.58	18.6	0.000	18.6	0.000

TABLE A.VII.34
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=2, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	138	1.05	145	0.95	6.3	0.011	6.3	0.008
71%	141	2.78	148	2.97	16.5	0.015	16.5	0.012
88%	143	3.73	151	4.55	20.5	0.015	20.5	0.008
92%	144	3.68	152	4.85	21.3	0.031	21.3	0.015
100%	193	5.52	200	5.52	23.3	0.000	23.3	0.000

TABLE A.VII.35
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=2, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	149	0.96	149	1.13	4.2	0.017	4.2	0.007
72%	153	2.52	159	2.67	11.1	0.022	11.1	0.011
89%	155	2.98	162	2.82	13.8	0.024	13.8	0.018
94%	156	3.11	163	2.80	14.6	0.020	14.6	0.017
100%	341	2.64	365	1.77	15.5	0.005	15.5	0.004

TABLE A.VII.36
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=2, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	149	1.02	149	1.00	3.2	0.012	3.2	0.008
72%	153	2.48	159	2.61	8.4	0.017	8.4	0.019
90%	156	2.95	163	2.98	10.5	0.019	10.5	0.015
96%	158	3.09	165	3.00	11.1	0.016	11.1	0.013
100%	449	8.59	478	9.68	11.6	0.000	11.6	0.000

TABLE A.VII.37
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=2, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	160	1.06	167	0.87	2.5	0.010	2.6	0.007
72%	165	2.41	172	2.47	6.7	0.020	6.7	0.019
90%	168	2.83	175	2.80	8.4	0.012	8.4	0.020
95%	169	3.16	176	3.02	8.8	0.020	8.8	0.012
100%	550	8.48	589	14.32	9.3	0.000	9.3	0.000

TABLE A.VII.38
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=3, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	138	1.16	145	1.12	4.2	0.010	4.2	0.015
72%	142	2.76	149	3.01	11.1	0.018	11.1	0.020
89%	145	3.80	152	4.64	13.8	0.013	13.8	0.015
94%	146	4.00	154	5.28	14.7	0.017	14.6	0.024
100%	236	5.56	238	0.82	15.5	0.005	15.5	0.004

TABLE A.VII.39
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=3, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	149	1.10	149	1.22	2.8	0.011	2.8	0.008
72%	153	2.61	160	2.82	7.4	0.021	7.4	0.013
90%	156	2.93	163	3.08	9.3	0.031	9.3	0.020
95%	159	3.01	165	3.01	9.9	0.021	9.8	0.017
100%	407	4.15	430	6.16	10.3	0.003	10.3	0.003

TABLE A.VII.40
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=3, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	152	4.19	152	4.07	2.1	0.009	2.1	0.013
73%	157	3.38	163	3.36	5.6	0.019	5.6	0.017
90%	160	3.18	167	3.31	7.0	0.016	7.0	0.013
96%	163	3.07	170	3.29	7.4	0.014	7.4	0.016
100%	541	5.94	571	6.71	7.8	0.000	7.8	0.000

TABLE A.VII.41
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=3, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	162	3.84	169	3.89	1.7	0.007	1.7	0.011
72%	168	3.31	175	3.31	4.5	0.012	4.5	0.014
91%	172	3.09	179	3.38	5.6	0.012	5.6	0.012
96%	175	3.08	182	3.29	6.0	0.013	6.0	0.022
100%	665	7.40	699	8.64	6.2	0.004	6.2	0.004

TABLE A.VII.42
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=4, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	143	4.75	150	4.64	3.2	0.010	3.2	0.017
72%	148	2.91	155	2.56	8.4	0.016	8.4	0.014
90%	151	1.94	159	1.13	10.5	0.018	10.5	0.019
96%	154	1.41	162	0.67	11.1	0.015	11.1	0.035
100%	280	5.61	287	5.61	11.6	0.003	11.6	0.000

TABLE A.VII.43
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=4, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	152	4.53	152	4.35	2.1	0.009	2.1	0.013
73%	158	3.27	164	3.26	5.6	0.019	5.6	0.017
90%	161	2.96	168	3.04	7.0	0.016	7.0	0.013
96%	163	3.05	170	3.11	7.4	0.014	7.4	0.016
100%	476	6.04	500	5.41	7.8	0.000	7.8	0.000

TABLE A.VII.44
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=4, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	154	4.81	154	4.70	1.6	0.007	1.6	0.010
73%	160	3.58	166	3.55	4.2	0.015	4.2	0.009
91%	164	3.29	171	3.30	5.3	0.014	5.3	0.016
97%	167	3.14	174	3.16	5.6	0.012	5.6	0.015
100%	632	12.20	661	12.42	5.8	0.000	5.8	0.000

TABLE A.VII.45
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=4, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	166	4.68	173	4.75	1.3	0.006	1.3	0.010
73%	172	3.68	179	3.60	3.4	0.011	3.4	0.011
91%	177	3.34	184	3.24	4.2	0.016	4.2	0.015
97%	180	3.22	187	3.24	4.5	0.012	4.5	0.010
100%	780	11.03	812	17.41	4.7	0.004	4.7	0.004

TABLE A.VII.46
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=7, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	162	3.93	169	3.93	1.8	0.010	1.8	0.009
73%	170	3.09	177	2.93	4.8	0.012	4.8	0.016
91%	175	2.40	183	1.85	6.0	0.017	6.0	0.015
96%	179	2.22	187	1.42	6.4	0.015	6.4	0.012
100%	422	5.85	428	5.85	6.7	0.000	6.7	0.000

TABLE A.VII.47
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=7, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	171	3.53	171	3.45	1.2	0.007	1.2	0.008
73%	178	3.44	184	3.40	3.2	0.012	3.2	0.009
91%	183	3.18	190	3.37	4.0	0.016	4.0	0.011
96%	186	3.46	193	3.31	4.3	0.012	4.3	0.015
100%	667	11.84	691	11.18	4.4	0.000	4.4	0.000

TABLE A.VII.48
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=7, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	173	4.52	174	4.70	0.9	0.006	0.9	0.007
73%	180	3.95	186	3.72	2.4	0.010	2.4	0.011
91%	186	3.79	193	3.62	3.0	0.010	3.0	0.013
97%	190	3.74	197	3.62	3.2	0.015	3.2	0.015
100%	892	7.94	916	8.07	3.3	0.005	3.3	0.005

TABLE A.VII.49
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=7, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	184	4.45	191	4.44	0.7	0.006	0.7	0.006
73%	190	3.83	197	3.80	1.9	0.011	1.9	0.010
91%	197	3.69	204	3.55	2.4	0.010	2.4	0.011
97%	202	3.74	209	3.59	2.6	0.013	2.6	0.012
100%	1108	12.34	1153	5.75	2.7	0.000	2.7	0.000

TABLE A.VII.50
TRAFFIC STATISTICS FOR RINGNET WITH 2 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=2 × G=15, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	204	6.22	211	6.06	0.9	0.008	0.9	0.007
73%	213	4.63	221	4.45	2.3	0.010	2.3	0.007
91%	221	4.12	228	3.94	2.8	0.012	2.8	0.010
97%	226	3.90	234	3.82	3.0	0.011	3.0	0.014
100%	792	6.99	798	6.99	3.1	0.000	3.1	0.000

TABLE A.VII.51
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=15, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	213	6.04	213	6.14	0.6	0.005	0.6	0.006
73%	221	5.22	227	5.01	1.5	0.008	1.5	0.008
91%	229	5.09	236	5.21	1.9	0.010	1.9	0.009
97%	235	5.03	242	4.78	2.0	0.008	2.0	0.012
100%	1257	15.60	1281	13.84	2.1	0.000	2.1	0.000

TABLE A.VII.52
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=15, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
28%	215	6.29	215	6.22	0.4	0.006	0.4	0.005
73%	223	5.51	229	5.41	1.1	0.006	1.1	0.007
91%	232	5.15	238	5.23	1.4	0.009	1.4	0.009
97%	238	4.91	246	5.00	1.5	0.009	1.5	0.009
100%	1671	28.19	1700	25.02	1.6	0.000	1.6	0.000

TABLE A.VII.53
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 2 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=15, R=2).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	227	6.31	234	6.37	0.3	0.005	0.3	0.005
73%	235	5.47	242	5.42	0.9	0.006	0.9	0.007
91%	244	5.19	252	5.24	1.1	0.009	1.1	0.007
97%	253	5.06	261	5.00	1.2	0.009	1.2	0.008
100%	2003	6.91	2037	9.90	1.2	0.000	1.2	0.000

TABLE A.VII.54
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=1, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	142	4.57	142	4.54	12.5	0.033	12.5	0.019
69%	147	3.29	153	3.07	32.0	0.019	32.0	0.019
85%	150	2.94	157	2.57	39.4	0.034	39.4	0.037
92%	152	2.71	159	2.03	42.7	0.037	42.7	0.050
100%	152	7.79	163	9.72	46.5	0.005	46.5	0.000

TABLE A.VII.55
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=1, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	141	4.02	141	3.80	9.5	0.022	9.5	0.004
70%	145	2.81	151	2.68	24.4	0.039	24.4	0.025
86%	147	2.37	155	2.19	30.1	0.040	30.1	0.032
92%	149	2.41	157	2.20	32.0	0.034	32.0	0.029
100%	311	4.15	334	5.50	34.9	0.000	34.9	0.000

TABLE A.VII.56
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=1, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	152	3.67	159	3.67	7.5	0.026	7.5	0.019
71%	156	2.56	163	2.85	19.7	0.030	19.7	0.031
87%	158	2.30	166	2.70	24.4	0.029	24.4	0.023
92%	159	2.21	167	2.64	25.6	0.014	25.6	0.029
100%	374	4.40	402	7.57	27.9	0.000	27.9	0.000

TABLE A.VII.57
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=2, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	157	0.92	158	1.05	6.3	0.016	6.3	0.009
71%	163	2.40	170	2.55	16.5	0.026	16.5	0.018
88%	167	3.40	175	2.86	20.5	0.012	20.5	0.026
92%	169	3.33	177	3.01	21.3	0.016	21.3	0.024
100%	220	9.00	224	5.20	23.3	0.000	23.3	0.000

TABLE A.VII.58
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=2, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	155	2.64	155	2.55	4.7	0.015	4.7	0.012
72%	160	2.07	167	2.01	12.5	0.026	12.5	0.014
89%	163	2.25	171	1.98	15.5	0.031	15.5	0.027
95%	165	2.38	173	1.97	16.5	0.020	16.5	0.025
100%	365	3.22	393	4.79	17.4	0.000	17.5	0.005

TABLE A.VII.59
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=2, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	165	3.34	172	3.21	3.8	0.007	3.8	0.018
72%	170	2.85	178	2.87	10.0	0.018	10.0	0.018
89%	173	2.94	181	2.97	12.5	0.019	12.5	0.024
94%	174	2.95	183	2.89	13.1	0.014	13.1	0.021
100%	451	1.73	475	3.60	14.0	0.005	14.0	0.003

TABLE A.VII.60
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=3, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	158	1.86	159	1.97	4.2	0.009	4.2	0.016
72%	164	2.83	172	2.93	11.1	0.014	11.1	0.017
89%	169	3.55	178	3.10	13.8	0.019	13.8	0.024
94%	171	3.77	181	3.06	14.6	0.016	14.6	0.020
100%	264	9.02	275	7.40	15.5	0.005	15.5	0.005

TABLE A.VII.61
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=3, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	159	1.79	159	1.76	3.2	0.010	3.2	0.012
72%	164	1.91	172	1.90	8.4	0.021	8.4	0.014
90%	167	2.15	176	2.09	10.5	0.020	10.5	0.020
96%	170	2.30	178	2.16	11.1	0.018	11.1	0.024
100%	432	3.89	455	6.16	11.6	0.003	11.6	0.000

TABLE A.VII.62
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=3, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	168	3.68	175	3.69	2.5	0.009	2.6	0.010
72%	174	3.09	182	3.22	6.7	0.016	6.7	0.014
90%	177	3.09	185	3.00	8.4	0.022	8.4	0.016
95%	179	2.98	187	2.98	8.8	0.018	8.8	0.015
100%	533	8.59	558	10.69	9.3	0.000	9.3	0.000

TABLE A.VII.63
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=4, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	162	4.40	162	4.13	3.2	0.010	3.2	0.012
72%	169	3.25	177	2.91	8.4	0.021	8.4	0.014
90%	174	2.77	183	2.51	10.5	0.020	10.5	0.020
96%	177	3.01	187	2.33	11.1	0.018	11.1	0.024
100%	308	9.05	318	10.42	11.6	0.003	11.6	0.000

TABLE A.VII.64
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=4, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	161	3.61	162	3.51	2.4	0.010	2.4	0.011
72%	168	2.63	175	2.45	6.3	0.018	6.3	0.010
90%	171	2.29	180	2.07	7.9	0.020	7.9	0.015
96%	174	2.37	183	2.11	8.4	0.021	8.4	0.023
100%	494	4.89	515	4.89	8.7	0.002	8.7	0.000

TABLE A.VII.65
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=4, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	172	3.41	179	3.29	1.9	0.011	1.9	0.011
73%	178	2.86	186	2.41	5.1	0.016	5.1	0.014
91%	182	2.69	191	2.14	6.3	0.019	6.3	0.018
97%	186	2.59	194	2.01	6.7	0.016	6.7	0.017
100%	610	4.85	633	7.90	7.0	0.000	7.0	0.000

TABLE A.VII.66
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=7, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	181	3.97	181	3.91	1.8	0.008	1.8	0.009
73%	190	3.66	198	3.38	4.8	0.017	4.8	0.014
91%	197	3.38	206	3.12	6.0	0.017	6.0	0.016
96%	202	3.42	211	2.45	6.4	0.019	6.4	0.019
100%	449	9.20	456	9.20	6.7	0.000	6.7	0.000

TABLE A.VII.67
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=7, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	180	3.41	181	3.23	1.4	0.010	1.4	0.007
73%	188	3.03	196	2.91	3.6	0.009	3.6	0.011
91%	193	2.84	202	2.67	4.5	0.014	4.5	0.012
96%	196	2.82	205	2.82	4.8	0.017	4.8	0.014
100%	674	3.69	696	2.51	5.0	0.003	5.0	0.003

TABLE A.VII.68
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=7, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	190	4.02	197	3.85	1.1	0.007	1.1	0.007
73%	197	3.44	205	3.23	2.9	0.011	2.9	0.013
91%	203	3.37	211	2.99	3.6	0.012	3.6	0.013
97%	207	3.25	215	2.90	3.9	0.014	3.8	0.013
100%	830	7.14	856	6.31	4.0	0.000	4.0	0.000

TABLE A.VII.69
TRAFFIC STATISTICS FOR RINGNET WITH 3 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=3 × G=15, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	223	5.94	224	5.74	0.9	0.006	0.9	0.007
73%	235	4.80	243	4.68	2.3	0.011	2.3	0.007
91%	243	4.62	254	4.52	2.8	0.010	2.8	0.012
97%	249	5.01	260	4.29	3.0	0.012	3.0	0.010
100%	819	9.97	830	11.64	3.1	0.002	3.1	0.000

TABLE A.VII.70
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=15, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	222	5.50	223	5.60	0.6	0.005	0.6	0.006
73%	232	4.68	240	4.67	1.7	0.009	1.7	0.010
91%	239	4.50	248	4.45	2.1	0.009	2.1	0.010
97%	244	4.70	254	4.28	2.3	0.009	2.3	0.010
100%	1188	11.59	1210	15.96	2.3	0.001	2.3	0.001

TABLE A.VII.71
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 3 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=15, R=3).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	233	5.42	240	5.21	0.5	0.005	0.5	0.006
73%	241	4.89	249	4.75	1.4	0.009	1.4	0.008
91%	249	4.86	258	4.59	1.7	0.009	1.7	0.009
97%	254	4.72	263	4.48	1.8	0.009	1.8	0.010
100%	1441	7.66	1466	6.17	1.9	0.000	1.9	0.000

TABLE A.VII.72
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=1, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	145	4.36	144	4.43	12.5	0.029	12.5	0.025
69%	149	3.43	154	3.08	32.0	0.034	32.0	0.029
85%	152	3.13	159	2.68	39.4	0.020	39.4	0.011
92%	155	2.88	162	2.27	42.7	0.043	42.7	0.038
100%	156	10.59	168	12.39	46.5	0.005	46.5	0.004

Results marked in red correspond to the highest observed values of c_L, which are reported in Table A.VII.ii.

TABLE A.VII.73
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 1 PG CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=1, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	154	4.55	161	4.65	10.0	0.012	10.1	0.012
69%	158	3.80	166	4.34	25.6	0.014	25.6	0.029
86%	162	3.80	168	4.00	32.0	0.046	32.0	0.057
92%	163	3.45	170	3.93	34.2	0.038	34.1	0.022
100%	349	4.80	368	4.46	37.2	0.005	37.2	0.005

TABLE A.VII.74
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=2, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	159	3.05	158	3.35	6.3	0.026	6.3	0.017
71%	164	3.19	171	2.82	16.5	0.020	16.5	0.025
88%	169	3.01	177	2.78	20.5	0.031	20.5	0.031
92%	170	3.28	179	2.64	21.3	0.034	21.3	0.040
100%	223	9.13	230	12.72	23.3	0.000	23.3	0.000

TABLE A.VII.75
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 2 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=2, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	168	4.40	175	4.47	5.1	0.015	5.1	0.016
71%	173	4.34	180	4.02	13.1	0.014	13.1	0.021
89%	177	4.02	184	4.26	16.5	0.021	16.5	0.025
95%	179	4.02	185	4.01	17.7	0.025	17.7	0.024
100%	401	4.33	423	5.01	18.6	0.000	18.6	0.000

TABLE A.VII.76
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=3, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	162	2.82	162	2.57	4.2	0.009	4.2	0.014
72%	169	3.12	175	2.90	11.1	0.018	11.1	0.024
89%	174	2.96	182	2.49	13.8	0.025	13.8	0.023
94%	176	2.91	185	2.03	14.6	0.023	14.6	0.024
100%	272	9.49	279	9.49	15.5	0.005	15.5	0.004

TABLE A.VII.77
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 3 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=3, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	171	4.36	178	4.22	3.4	0.010	3.4	0.007
72%	178	4.01	185	4.03	9.0	0.013	9.0	0.015
90%	181	3.84	188	3.80	11.1	0.019	11.1	0.014
96%	184	3.88	191	3.63	11.9	0.025	11.9	0.017
100%	459	6.88	481	6.18	12.4	0.000	12.4	0.000

TABLE A.VII.78
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=4, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	165	4.15	165	4.23	3.2	0.011	3.2	0.012
72%	172	3.36	179	2.90	8.4	0.021	8.4	0.023
90%	177	3.25	186	2.58	10.5	0.026	10.5	0.032
96%	181	2.97	191	2.13	11.1	0.019	11.1	0.020
100%	316	10.11	323	15.35	11.6	0.002	11.6	0.002

TABLE A.VII.79
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 4 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=4, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	175	4.34	182	4.24	2.5	0.013	2.6	0.010
72%	182	3.91	189	3.64	6.7	0.016	6.7	0.017
90%	186	3.61	194	3.42	8.4	0.017	8.4	0.016
95%	188	3.73	195	3.46	8.8	0.020	8.8	0.021
100%	515	8.75	538	7.64	9.3	0.000	9.3	0.000

TABLE A.VII.80
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=7, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	184	3.90	184	3.60	1.8	0.009	1.8	0.008
73%	193	3.51	200	2.99	4.8	0.013	4.8	0.014
91%	199	3.32	207	2.88	6.0	0.017	6.0	0.018
96%	204	3.14	213	2.00	6.4	0.015	6.4	0.019
100%	454	12.46	462	12.46	6.7	0.000	6.7	0.000

TABLE A.VII.81
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 7 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=7, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	193	4.67	200	4.61	1.5	0.008	1.5	0.008
73%	202	4.31	209	4.10	3.9	0.014	3.9	0.016
91%	208	4.04	215	3.80	4.8	0.013	4.8	0.015
96%	211	4.15	218	3.75	5.1	0.011	5.1	0.017
100%	689	6.11	708	6.42	5.3	0.000	5.3	0.000

TABLE A.VII.82
TRAFFIC STATISTICS FOR RINGNET WITH 4 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=4 × G=15, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{L_{PG}}$	$\sigma_{L_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$	$\overline{T_{PG}}$	$\sigma_{T_{PG}}$
27%	226	5.86	226	5.85	0.9	0.008	0.9	0.007
73%	238	5.19	246	4.64	2.3	0.009	2.3	0.011
91%	247	4.93	257	4.39	2.8	0.013	2.8	0.014
97%	254	4.57	263	3.97	3.0	0.011	3.0	0.014
100%	824	13.04	834	14.96	3.1	0.002	3.1	0.002

TABLE A.VII.83
TRAFFIC STATISTICS FOR RINGNET WITH 5 1ST LEVEL RINGS AND 15 PGS CONNECTED AT EACH RING, AND 4 PARALLEL RINGS USED AT THE NETWORK ROOT (F=5 × G=15, R=4).

Load (percentage of the throughput T_{RW_MAX} (1))	Average latency (clock cycles)				Average throughput (bits per cycle)			
	Read		Write		Read		Write	
	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{L_{PG}}$	$\sigma_{\overline{L_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$	$\overline{T_{PG}}$	$\sigma_{\overline{T_{PG}}}$
27%	236	5.97	243	5.92	0.7	0.007	0.7	0.006
73%	246	5.67	254	5.29	1.8	0.010	1.8	0.009
91%	254	5.25	262	5.03	2.3	0.012	2.3	0.009
97%	259	5.37	267	5.00	2.4	0.011	2.4	0.010
100%	1157	6.93	1175	8.54	2.5	0.000	2.5	0.000

Bibliography

Author's contributions

[Dom11a] M. Domański, T. Grajek, D. Karwowski, K. Klimaszewski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "Multiview HEVC – experimental results," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, MPEG 2011 / M22147, Geneva, Switzerland, 28 Nov. - 02 Dec. 2011.

[Dom11b] M. Domański, T. Grajek, D. Karwowski, K. Klimaszewski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "Technical Desciption of Poznan University of Technology proposal for Call on 3D Video Coding Technology," ISO/IEC JTC1/SC29/WG11, MPEG 2011 / M22697, Geneva, Switzerland, 28 Nov.-02 Dec. 2011.

[Dom12a] M. Domański, T. Grajek, D. Karwowski, K. Klimaszewski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "New Coding Technology for 3D Video with Depth Maps as Proposed for Standardization within MPEG," in *Proc. 19th Int. Conf. on Syst., Signals and Image Processing (IWSSIP)*, 2012, pp. 401-404.

[Dom12b] M. Domański, T. Grajek, D. Karwowski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "Coding of multiple video+depth using HEVC technology and reduced representations of side views and depth maps," in *Proc. Picture Coding Symposium (PCS)*, 2012, pp. 5-8.

[Dom12c] M. Domański, J. Konieczny, M. Kurc, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "3D Video Compression by Coding of Disoccluded Regions," in *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 2012, pp. 1317-1320.

[Dom13a] M. Domański, T. Grajek, D. Karwowski, K. Klimaszewski, J. Konieczny, M. Kurc, A. Łuczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "Poznański kodek obrazów trójwymiarowych," *Przegląd Telekomunikacyjny*, no. 2-3, pp. 81-83, February/March 2013.

[Dom13b] M. Domański, O. Stankiewicz, K. Wegner, M. Kurc, J. Konieczny, J. Siast, J. Stankowski, R. Ratajczak and T. Grajek, "High Efficiency 3D Video Coding Using New Tools Based on View Synthesis," *IEEE Trans. on Image Processing*, vol.22, no. 9, pp. 3517-3527, 2013.

[Dom14a] M. Domański A. Dziembowski, A. Kuehn, M. Kurc, A. Łuczak, D. Mieloch, J. Siast, O. Stankiewicz and K. Wegner, "*Experiments on acquisition and processing of video for free-viewpoint television*," in *Proc. 3DTV Conf. 2014*, Budapest, Hungary, 2-4 July 2014, pp. 1-4.

[Dom14b] M. Domański, A. Dziembowski, A. Kuehn, M. Kurc, A. Łuczak, D. Mieloch, J. Siast, O. Stankiewicz and K. Wegner, "Poznan Blocks – a multiview video test sequence and camera parameters for Free Viewpoint Television*,"* ISO/IEC JTC1/SC 29/WG11, MPEG2014, Doc. m32243, San Jose, USA, 2014.

[Dom14c] M. Domański, K. Klimaszewski, J. Konieczny, M. Kurc, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski and K. Wegner, "Image coding method," Patent office: USPTO, Status: granted, Application number: US 13/680652, Filling date: 19.11.2012, Publication number: US 2013/0129235A1, Publication date: 23.05.2013, Patent number: US 8761527, Date of Patent: 24.06.2014.

[Dom15] M. Domański, J. Konieczny, M. Kurc, A. Łuczak, J. Siast, O. Stankiewicz and K. Wegner, "Fast depth estimation on mobile platforms and FPGA devices," in *Proc. 2015 3DTV-Conf.: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, Lisbon, 2015.

[Łucz10] A. Łuczak, M. Kurc, M. Stępniewska and J. Siast, "Interfejs komunikacyjny dla układów FPGA serii Virtex," *Pomiary Automatyka Kontrola*, PAK, pp. 749-751, 2010.

[Łucz11] A. Łuczak, M. Stępniewska, J. Siast, M. Domański, O. Stankiewicz, M. Kurc and J. Konieczny, "Network-on-multi-chip (NoMC) with monitoring and debugging support," *J. Telecommun. and Inform. Techno.*, no. 3, pp. 81, 2011.

[Łucz14] A. Łuczak, S. Maćkowiak and J. Siast; "*Depth Map's 2D Histogram Assisted Occlusion Handling in Video Object Tracking*," in *Proc. Int. Conf. on Computer Vision and Graphics ICCVG 2014*, Warsaw, Poland, 15-17 September 2014, pp. 400-408.

[Łucz17] A. Łuczak, S. Maćkowiak, M. Domański, J. Siast and T. Grajek, "A system and a method for tracking objects," Patent office: USPTO, Status: granted, Application number: US 14/664862, Filling date: 22.03.2015, Patent number: US 9672634, Date of Patent: 06.06.2017.

[Sia12] J. Siast, O. Stankiewicz, K. Wegner and M. Domański, "Independent intra-period coding in 3D-HTM," Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, a0091, Stockholm, Sweden, 16-20 July 2012.

[Sia13] J. Siast, J. Stankowski, T. Grajek and M. Domański, "Digital Watermarking with Local Strength Adjustment for AVC-Compressed HDTV Bitstreams," in *Proc. 30th Picture Coding Symposium, PCS 2013*, San Jose, CA, USA, 8-11 December 2013, pp. 53-56.

[Sia14] J. Siast and M. Domański, "A method and a system for video signal encoding and decoding with motion estimation," Patent office: EPO, Status: granted, Application number: EP.13180012, Filling date: 12.08.2013, Publication number: EP 2699001A1, Publication date: 19.02.2014, Patent number: EP.2699001, Date of Patent: 06.05.2015.

[Sia16] J. Siast, J. Stankowski and M. Domański, "Hierarchical fast selection of intraframe prediction mode in HEVC," *Int. Journal of Electronics and Telecommunications*, vol. 62, no. 2, Electronics and Telecommunications Committee of Polish Academy of Sciences, Warsaw, Poland, pp. 147-151, 2016.

[Sia19] J. Siast, A. Łuczak and M. Domański, "RingNet: A Memory-Oriented Network-On-Chip Designed for FPGA," in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1284-1297, 2019.

[Stanki12a] O. Stankiewicz, K. Wegner and J. Siast,"3D-CE2h results on Adaptive Depth Quantization combined with Nonlinear Depth Representation," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25022, Geneva, Switzerland, 30 April-4 May 2012.

[Stanki12b] O. Stankiewicz, K. Wegner and J. Siast, "3D-CE2h results on Adaptive Depth Quantization combined with Nonlinear Depth Representation," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25022, Geneva, Switzerland, 30 April-4 May 2012.

[Stanki12c] O. Stankiewicz, K. Wegner and J. Siast, "3D-CE2a results on Nonlinear Depth Representation," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25017, Geneva, Switzerland, 30 April-4 May 2012.

[Stanko12] J. Stankowski, M. Domański, O. Stankiewicz, J. Konieczny, J. Siast and K. Wegner, "Extensions of the HEVC Technology for Efficient Multiview Video Coding," in *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 2012, pp. 225-228.

[Stanko14] J. Stankowski, D. Karwowski, T. Grajek, K. Wegner, J. Siast, K. Klimaszewski, O. Stankiewicz and M. Domański, "Bitrate Distribution of Syntax Elements in the HEVC Encoded Video," in *Proc. Int. Conf. on Signal and Electronics Syst., ICSES 2014*, Poznań, Poland, 11-13 September 2014, pp. 1-4.

[Stanko15] J. Stankowski, D. Karwowski, T. Grajek, K. Wegner, J. Siast, K. Klimaszewski, O. Stankiewicz and M. Domański, "Analysis of Compressed Data Stream Content in HEVC Video Encoder," *Int. Journal of Electronics and Telecommunications*, vol. 61, no. 2, Electronics and Telecommunications Committee of Polish Academy of Sciences, Warsaw, Poland, pp. 121-127, 2015.

[Stę10a] M. Stępniewska, O. Stankiewicz, A. Łuczak and J. Siast, "Embedded debugging for NoCs," in *Proc. 17th Int. Conf. on Mixed Design of Integrated Circuits and Syst.*, Wrocław, Poland, 24-26 June 2010, pp. 601-606.

[Stę10b] M. Stępniewska, A. Łuczak and J. Siast, "Network-on-Multi-Chip (NoMC) for multi-FPGA multimedia systems," in *Proc. 13th Euromicro Conf. on Digital Syst. Design*, Lille, France, 1-3 September 2010, pp. 475-481.

[Weg12a] K. Wegner, J. Siast, J. Konieczny, O. Stankiewicz and M. Domański, "Poznan University of Technology tools for 3DV coding integrated into 3D-HTM," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m23783, San Jose, USA, 6-10 February 2012.

[Weg12b] K. Wegner, O. Stankiewicz and J. Siast, "3D-CE2h results on Nonlinear Depth Representation," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25020, Geneva, Switzerland, 30 April-4 May 2012.

[Weg12c] K. Wegner, O. Stankiewicz and J. Siast, "3D-CE1h results on Depth Map Disocclusion Coding by Poznan University of Technology," ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25014, Geneva, Switzerland, 30 April-4 May 2012.

[Weg12d] K. Wegner, O. Stankiewicz and J. Siast, “3D-CE1h cross check of RWTH University proposal on Warping Based Prediction by Poznan University of Technology,” ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25187, Geneva, Switzerland, 30 April-4 May 2012.

[Weg12e] K. Wegner, O. Stankiewicz and J. Siast, “3D-CE2a cross check of Samsung proposal on Adaptive Depth Quantization by Poznan University of Technology,” ISO/IEC JTC1/SC29/WG11 MPEG 2012, Doc. m25018, Geneva, Switzerland, 30 April-4 May 2012.

[Weg12f] K. Wegner, O. Stankiewicz J. Siast and M. Domański, “Independent intra-period coding in HEVC,” ISO/IEC JTC1/SC 29/WG11, Doc. JTCVC-K0332 11th Meeting: Shanghai, CN, 10–19, 2012.

Other references

(not co-authored by Jakub Siast)

[Abb14] S. Abba and Jeong-A Lee, “A parametric-based performance evaluation and design trade-offs for interconnect architectures using FPGAs for networks-on-chip,” in *Microprocessors and Microsystems*, vol. 38, no. 5, pp. 375-398, July 2014.

[Abd14] M. S. Abdelfattah and V. Betz, “The case for embedded networks on chip on field-programmable gate arrays,” in *IEEE Micro*, vol. 34, no. 1, pp. 80-89, Jan.-Feb. 2014.

[Abd16a] M. S. Abdelfattah and V. Betz, “Power analysis of embedded NoCs on FPGAs and comparison with custom buses,” in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 165-177, Jan. 2016.

[Abd16b] M. S. Abdelfattah and V. Betz, “LYNX: CAD for FPGA-based networks-on-chip,” in *Proc. 26th Int. Conf. on Field Programmable Logic and Applications (FPL)*, Lausanne, 2016, pp. 1-10.

[Abd16c] M. Abdelfattah, V. Betz, “Embedded Networks-on-Chip for FPGA,” in *Reconfigurable Logic: Architecture, Tools, and Applications,* Boca Raton, FL, USA: CRC Press, 2016.

[Abd17] M. S. Abdelfattah, A. Bitar and V. Betz, “Design and applications for embedded networks-on-chip on FPGAs,” in *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 1008-1021, June 2017.

[Add17] M. Adda and A. Peratikou, “Routing and Fault Tolerance in Z-Fat Tree,” in *IEEE Trans. Parallel and Distrib. Syst.*, vol. 28, no. 8, pp. 2373-2386, 1 Aug. 2017.

[Ahm18] R. Ahmed, H. Mostafa and A. H. Khalil, “Impact of dynamic partial reconfiguration on CONNECT Network-on-Chip for FPGAs, ” in *Proc. 2018 13th Int. Conf. on Design & Technol. of Integrated Systems In Nanoscale Era* (*DTIS*), Taormina, 2018, pp. 1-5.

[AlF12] M. A. Al Faruque, T. Ebi and J. Henkel, “AdNoC: Runtime adaptive network-on-chip architecture,” in *IEEE Trans. Very Large Integr. (VLSI) Syst.*, vol. 20, no. 2, pp. 257-269, Feb. 2012.

[Alt09] *Quartus II Handbook, Recommended HDL Coding Styles*, QII51007-9.1.0, Altera Co., Nov. 2009.

[Alt11] *Stratix V device handbook*, SV51002, v1.3, vol. 2, Altera Co., Nov. 2011.

[Alt15a] *Stratix V device overview*, SV51001, Altera Co., Oct. 2015.

[Alt15b] *Devices: 28nm device portfolio*, Altera Product Catalog, Altera Co., 2015.

[Alt16] *Arria 10 device overview*, A10-overwiev, Altera Co., Oct. 2016.

[Ben02] L. Benini and G. D. Micheli, “Networks on chips: A new SoC paradigm,” in *IEEE Computers*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[Ben06] L. Benini and G. D. Micheli, *Networks on chips: Technology and Tools*, San Francisco, CA, USA: Elsevier, 2006.

[Ber04] D. Bertozzi and L. Benini, “Xpipes: A network-on-chip architecture for gigascale systems-on-chip,” in *IEEE Circuits and Syst. Magazine*, vol. 4, no. 2, pp. 18-31, 2004.

[Buk17] A. V. Bukit and Wirawan, “3D video coding development based on FPGA platform Xilinx Zynq-7000,” in *Proc. 2017 Int. Seminar on Intelligent Technol. and Its Applications (ISITIA)*, Surabaya, 2017, pp. 29-34.

[Bre17] M. F. Brejza, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo, “A High-Throughput FPGA Architecture for Joint Source and Channel Decoding,” in *IEEE Access*, vol. 5, pp. 2921-2944, 2017.

[Cha15] H. Y. Chang, I. H. R. Jiang, H. P. Hofstee, D. Jamsek and G. J. Nam, “Feature detection for image analytics via FPGA acceleration,” in *IBM Journal of Research and Development*, vol. 59, no. 2/3, pp. 8:1-8:10, March-May 2015.

[Che12] S.-J. Chen, L. Ying-Cherng, W.-C. Tsai, Y.-H. Hu, *Reconfigurable Networks-on-Chip*, New York, USA: Springer-Verlag, 2012.

[Cle16] J. A. Clemente, R. Gran, A. Chocano, C. del Prado and J. Resano, "Hardware Architectural Support for Caching Partitioned Reconfigurations in Reconfigurable Systems," in *IEEE Trans. Very Large Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 530-543, Feb. 2016.

[Cot12] É. Cota, A. de Morais Amory, M. Soares Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*, Boston, MA, USA: Springer, 2012.

[Dal99] W. J. Dally, "Interconnect-limited VLSI architecture," in *Proc. of the IEEE 1999 Inter. Interconnect Technol. Conf.*, San Francisco, CA, USA, 1999, pp. 15-17.

[Dal01] W. J. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Proc. of the 38th Design Automation Conf. (IEEE Cat. No.01CH37232)*, 2001, pp. 684-689.

[Dal03] W. Dally and B. Towles, *Principles and Practices of Interconnection Network*, San Francisco, USA: Morgan Kaufmann, 2003.

[Dim15] G. Dimitrakopoulos, A. Psarras, I. Seitanidis, *Microarchitecture of Network-on-Chip Routers: A Designer's Perspective*, New York, USA: Springer-Verlag, 2015.

[Gar06] P. Garstecki, A. Łuczak and T. Żernicki, "A bit-serial architecture for H.264/AVC interframe decoding," in *Proc. European Signal Processing Conf. EUSIPCO2006,* Florence, Italy, 4-8 September 2006, pp. 1-5.

[Hel15] K. A. Helal, S. Attia, T. Ismail and H. Mostafa, "Comparative review of NoCs in the context of ASICs and FPGAs," in *Proc. 2015 IEEE Int. Symp. Circuits and Syst. (ISCAS)*, Lisbon, pp. 1866-1869.

[Hry07] E. Hrynkiewicz and S. Kolodzinski, "Decomposition of Logic Functions in Reed-Muller Spectral Domain," in *Proc. 2007 IEEE Design and Diagnostics of Electronic Circuits and Syst.*, Kraków, 11-13 April 2007, pp. 219-222.

[Hud16] S. Huda, J. Anderson, "Circuits and Embedded Networks-on-Chip for FPGA," in *Reconfigurable Logic: Architecture, Tools, and Applications*, Boca Raton, FL, USA: CRC Press, 2016.

[Int16] *Stratix 10 GX/SX device overview*, S10-overview, Intel Co., Oct. 2016.

[Int17] *Intel Cyclone 10 GX device overview*, C10GX51001, Intel Co., Nov. 2017.

[Int18] *Intel Stratix 10 Logic Array Blocks and Adaptive Logic Modules*, UG-S10LAB, Intel Co., Sep. 2018.

[Kam18] H. M. Kamali, K. Z. Azar and S. Hessabi, "DuCNoC: A high-throughput FPGA-based NoC simulator using dual-clock lightweight router micro-architecture," in *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 208-221, Feb. 2018.

[Kan18] S. Kanakala, K. Ashok Kumar and P. Dananjayan, "High Reliability NoC switch using Modified Hamming Code with Transient Faults, " in *Proc. 2018 IEEE Int. Conf. on System, Computation, Automation and Networking* (*ICSCA*), Pondicherry, 2018, pp. 1-5.

[Kap15] N. Kapre and J. Gray, "Hoplite: Building austere overlay NoCs for FPGAs," in *Proc. Int. Conf. Field Programmable Logic and Applications (FPL)*, London, 2015, pp. 1-8.

[Kap17a] N. Kapre, "On bit-serial NoCs for FPGAs," in *Proc. IEEE Int. Symp. Field-Programmable Custom Computing Machines (FCCM)*, Napa, CA, 2017, pp. 32-39.

[Kap17b] N. Kapre, "Implementing FPGA overlay NoCs using the Xilinx UltraScale memory cascades," in *Proc. IEEE Annu. Int. Symp. Field-Programmable Custom Computing Machines (FCCM)*, Napa, CA, 2017, pp. 40-47.

[Kap17c] N. Kapre, "Deflection-routed butterfly fat trees on FPGAs," in *Proc. Int. Conf. Field Programmable Logic and Applications (FPL)*, Ghent, 2017, pp. 1-8.

[Ker79] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," in *Computer Networks*, vol. 3, no. 4, pp. 267-286, Sep. 1979.

[Kil00] J. S. Kilby, "The integrated circuit's early history," in *Proc. of the IEEE*, vol. 88, no. 1, pp. 109-111, Jan. 2000.

[Kum16] R. Kumar and A. Gordon-Ross, "MACS: A highly customizable low-latency communication architecture," in *IEEE Trans. Parallel and Distrib. Syst.*, vol. 27, no. 1, pp. 237-249, Jan. 2016.

[Lat13] *LatticeECP2/M family data sheet*, DS1006, v04.1, Lattice Semiconductor Co., Sept. 2013.

[Lat15a] *LatticeECP3 family data sheet*, DS1021, v02.8EA, Lattice Semiconductor Co., Mar. 2015.

[Lat15b] *ECP5 and ECP5-5G Memory Usage Guide*, TN1264, v1.2, Lattice Semiconductor Co., Nov. 2015.

[Lat16] *ECP5 and ECP5-5G family*, DS1044, v1.6, Lattice Semiconductor Co., Feb. 2016.

[Lei85] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," in *IEEE Trans. on Computers*, vol. C-34, no. 10, pp. 892-901, Oct. 1985.

[Lic18] G. D. Licciardo, C. Cappetta and L. D. Benedetto, "Design of a Gabor Filter HW Accelerator for Applications in Medical Imaging," in *IEEE Trans. on Components, Packaging and Manufacturing Technol.*, pp. 1-8, April, 2018.

[LiuS13] S. Liu, A. Jantsch and Z. Lu, "Analysis and Evaluation of Circuit Switched NoC and Packet Switched NoC," in *Proc. 2013 Euromicro Conf. on Digital Syst. Design*, Los Alamitos, CA, 2013, pp. 21-28.

[LiuS14] S. Liu, A. Jantsch and Z. Lu, "A fair and maximal allocator for single-cycle on-chip homogeneous resource allocation," in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2230-2234, Oct. 2014.

[LiuT16] T. Liu, N. K. Dumpala and R. Tessier, "Hybrid hard NoCs for efficient FPGA communication," in *Proc. 2016 Int. Conf. on Field-Programmable Technol. (FPT)*, Xi'an, 2016, pp. 157-164.

[Luu16] J. Luu, "Adaptive packing for design space exploration of logic block architectures," in *Reconfigurable Logic: Architecture, Tools, and Applications*, Boca Raton, FL, USA: CRC Press, 2016.

[Łub95] T. Łuba and H. Selvaraj, "A General Approach to Boolean Function Decomposition and its Application in FPGA-Based Synthesis," in *VLSI Design*, vol. 3, no. 3-4, pp. 289-300, 1995.

[Łucz08] A. Łuczak, P. Garstecki, O. Stankiewicz and M. Stępniewska, "Network-On-Chip Based Architecture of H. 264 Video Decoder," in *Proc. Int. Conf. on Signals and Electronic Syst.*, Krakow, Poland, 14-17 September 2008, pp. 419-422.

[Łucz09] A. Łuczak, M. Kurc, M. Stępniewska and K. Wegner, "Platforma przetwarzania rozproszonego bazująca na sieci NoC," in *Pomiary Automatyka Kontrola*, PAK, pp. 690-692, 2009.

[Mai15] P. Maidee and A. Kaviani, "Improving FPGA NoC performance using virtual cut-through switching technique," in *Proc. Int. Conf. ReConFigurable Computing and FPGAs (ReConFig)*, Mexico City, 2015, pp. 1-6.

[Mai17] P. Maidee, A. Kaviani and K. Zeng, “LinkBlaze: Efficient global data movement for FPGAs,” in *Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, 2017, pp. 1-8.

[Man14] A. Mandal, S. P. Khatri, R. Mahapatra, *Source-Synchronous Networks-On-Chip*, New York, USA: Springer-Verlag, 2014.

[Mar16] Z. Marrakchi, H. Mehrez, “Tree-based FPGA routing architectures,” in *Reconfigurable Logic: Architecture, Tools, and Applications*, Boca Raton, FL, USA: CRC Press, 2016.

[Mat09] H. Matsutani, M. Koibuchi, Y. Yamada, D. F. Hsu and H. Amano, “Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network,” in *IEEE Trans. Parallel and Distrib. Syst.*, vol. 20, no. 8, pp. 1126-1141, Aug. 2009.

[Mea80] C. A. Mead, L. A. Conway, “Introduction to VLSI systems,” New York, USA: Addison-Wesley, 1980.

[Mic18a] *PolarFire FPGA Fabric*, UG0680, Microsemi, Mar. 2018.

[Mic18b] *IGLOO2 FPGA and SmartFusion2 SoC FPGA*, DS0128, Microsemi, Aug. 2018.

[MP11] “Call for Proposals on 3D Video Coding Technology,” ISO/IEC JTC1/SC29/WG1, Doc. N12036, Geneva, Switzerland, March 2011.

[Mur07] S. Murali *et al.*, “Synthesis of predictable networks-on-chip-based interconnect architectures for chip multiprocessors,” in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 8, pp. 869-880, Aug. 2007.

[Nay97] D. Naylor, S. Johnes, *VHDL: A Logic Synthesis Approach*, London, UK: Chapman & Hall, 1997.

[Pal03] S. Palnitkar, P. Goel, *Verilog HDL: A Guide to Digital Design and Synthesis*, Second Edition, Mountain View, CA, USA: Prentice Hall, 2003.

[Pap12] M. K. Papamichael and J. C. Hoe, “CONNECT: Re-examining conventional wisdom for designing NoCs in the context of FPGAs,” in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, New York, 2012, pp. 37-46.

[Pap15] M. K. Papamichael and J. C. Hoe, “The CONNECT network-on-chip generator,” in *Computer*, vol. 48, no. 12, pp. 72-79, Dec. 2015.

[Pos13] J. Postman, T. Krishna, C. Edmonds, L. S. Peh and P. Chiang, “SWIFT: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects,” in *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1432-1446, Aug. 2013.

[Red19] K. S. Reddy and K. Vipin, “OpenNoC: An Open-Source NoC infrastructure for FPGA-based Hardware Acceleration, ” in *IEEE Embedded Systems Letters*, Early Access, Mar. 2019.

[Ret14] J. Rettkowski and D. Göhringer, “RAR-NoC: A reconfigurable and adaptive routable network-on-chip for FPGA-based multiprocessor systems,” in *Proc. Int. Conf. ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, 2014, pp. 1-6.

[Sed06] P. Sedcole, B. Blodget, T. Becker, J. Anderson and P. Lysaght, "Modular dynamic reconfiguration in Virtex FPGAs," in *IEE Proc. - Computers and Digital Techniques*, vol. 153, no. 3, pp. 157-164, 2 May 2006.

[She14] S. N. Shelke and P. B. Patil, "Low-latency, low-area overhead and high throughput NoC architecture for FPGA based computing system," in *Proc. Int. Conf. Electronic Syst., Signal Process. and Computing Technol.*, Nagpur, 2014, pp. 53-57.

[Sid18] Siddhartha and N. Kapre, "Hoplite-Q: Priority-Aware Routing in FPGA Overlay NoCs, " in *Proc. 2018 IEEE 26th Annual Int. Symp. on Field-Programmable Custom Computing Machines* (*FCCM*), Boulder, 2018, pp. 17-24.

[Sny17] W. Snyder, "Introduction to Verilator," [Online], Available: https://www.veripool.org/wiki/verilator, Accessed on: Dec. 19, 2017.

[Stanki07] O. Stankiewicz and A. Łuczak, "Flexible processor architecture optimized for advanced coding algorithms," in *Proc. Picture Coding Symposium 2007*, Lisbon, Portugal, November 7-9 2007.

[Stę06a] M. Stępniewska and A. Łuczak, "A Bit-serial Architecture for H.264/AVC Encoder," in *Proc. IEEE Int. Symposium on Circuits and Syst. ISCAS 2006*, 2006, pp. 818-821.

[Stę06b] M. Stępniewska and A. Łuczak, "Reconfigurable architecture of AVC/H.264 Integer Transform," in *Proc. European Signal Processing Conf. EUSIPCO2006*, Florence, Italy, 4-8 September 2006, pp. 1-5.

[Stę13] M. Stępniewska, "Advanced video codecs implementation using Network-on-Chip in FPGA devices," PhD Dissertation at Poznan University of Technology, Faculty of Electronics and Telecommunications, Poznań, 2013.

[Son03] Y. H. Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," in *IEEE Trans. Parallel and Distrib. Syst.*, vol. 14, no. 3, pp. 259-275, Mar. 2003.

[Syn17] *Synopsys FPGA synthesis user guide*, Synopsys Inc., Apr. 2017.

[Syn19] Synopsys Inc., "Synplify Premier: Accelerate Implementation of FPGA Designs and FPGA-based Prototypes," 2019, [Online]. Available: https://www.synopsys.com/implementation-and-signoff/fpga-based-design/synplify-premier.html, Accessed on: 13 Mar. 2019.

[Tat14] K. Tatas, K. Siozios, D. Soudris, A. Jantsch, *Designing 2D and 3D Network-on-Chip Architectures*, New York, USA: Springer-Verlag, 2014.

[The11] D. Theodoropoulos, G. Kuzmanov and G. Gaydadjiev, "Multi-Core Platforms for Beamforming and Wave Field Synthesis," in *IEEE Trans. on Multimedia*, vol. 13, no. 2, pp. 235-245, April 2011.

[Tod14] E. Todorovich, M. Leonetti and R. Brinks, "An advanced NoC with debug services on FPGA," in *Proc. Southern Conf. Programmable Logic (SPL)*, Buenos Aires, 2014, pp. 1-6.

[Tto16] C. Ttofis, C. Kyrkou and T. Theocharides, “A Low-Cost Real-Time Embedded Stereo Vision System for Accurate Disparity Estimation Based on Guided Image Filtering,” in *IEEE Trans. on Computers*, vol. 65, no. 9, pp. 2678-2693, Sept. 1 2016.

[Was17] S. Wasly, R. Pellizzoni and N. Kapre, “HopliteRT: An efficient FPGA NoC for real-time applications,” in *Proc. Int. Conf. Field Programmable Technol. (ICFPT)*, Melbourne, 2017, pp. 64-71.

[Weh16] T. Wehbe and X. Wang, “Secure and dependable NoC-connected systems on an FPGA chip,” in *IEEE Trans. on Reliability*, vol. 65, no. 4, pp. 1852-1863, Dec. 2016.

[Wyr13] B. Wyrwoł, E. Hrynkiewicz, “Decomposition of the fuzzy inference system for implementation in the FPGA structure,” in *Int. Journal of Applied Math. and Computer Science*, vol. 23, no. 2, pp. 473–483, June 2013.

[Vip17] K. Vipin, J. Gray and N. Kapre, “Enabling partial reconfiguration and low latency routing using segmented FPGA NoCs,” in *Proc. Int. Conf. Field Programmable Logic and Applications (FPL)*, Ghent, 2017, pp. 1-8.

[Xil15] *Vivado AXI reference guide*, UG1037, v3.0, Xilinx Inc., June 2015.

[Xil16a] *Zynq UltraScale+ MPSoC*, XMP104, v2.1, Xilinx Inc., 2016.

[Xil16b] *UltraScale FPGA*, XMP102, v1.7, Xilinx Inc., 2016.

[Xil16c] *All programmable 7 Series*, XMP101, v1.2, Xilinx Inc., 2016.

[Xil16d] *Cost-optimized portfolio*, XMP100, v1.6, Xilinx Inc., 2016.

[Xil16e] *7 Series FPGAs: Configurable Logic Block*, UG474, v1.8, Xilinx Inc., 2016.

[Xil17] *UltraScale+ FPGAs*, XMP103, v1.10, Xilinx Inc., 2017.

[Yoo13] Y. J. Yoon, N. Concer, M. Petracca and L. P. Carloni, “Virtual channels and multiple physical networks: Two alternatives to improve NoC performance,” in *IEEE Trans. Computer-Aided Design of Integr. Circuits and Syst.*, vol. 32, no. 12, pp. 1906-1919, Dec. 2013.

[Zhu17] M. Zhu, Y. Jiang, M. Yang and L. De Luna, “A scalable parameterized NoC emulator built upon Xilinx Virtex-7 FPGA,” in *Proc. Int. Conf. Syst. Engineering (ICSEng)*, Las Vegas, 2017, pp. 287-290.

[Zyd11] D. Zydek, H. Selvaraj, G. Borowik and T. Łuba, “Energy characteristic of a processor allocator and a Network-on-Chip,” in *Int. Journal of Applied Math. and Computer Science*, vol. 21, no. 2. pp. 385-399, 2011.