TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering

Thesis code

Martin Dída 156447 IASM

PERFORMANCE ANALYSIS OF BONFIRE NETWORK ON CHIP

Master's thesis

Supervisor: Jaan Raik
Doctor of Philosophy
Professor
Co-Supervisor: Siavoosh Payandeh Azad
Master of Science
Early stage researcher
Co-Supervisor: Behrad Niazmand
Master of Science
Early stage researcher

Tallinn 2016

Author’s Decleration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Martin Dída

May 30, 2016

Abstract

The thesis is written in English language consisting of 44 pages of text, 4 chapters, 26 figures, 0 tables.

Nowadays a Network-on-Chip (NoC) paradigm is considered to be an efficient on-chip communication infrastructure in System on Chip (SoC) design. Over the last two decades many NoC design concepts have been proposed and analyzed. Among the important aspects of NoC's design belongs the performance of a network which should be carefully measured and analyzed.

This master's thesis deals with measurement and analysis of Bonfire NoC which has been designed in TUT university. The thesis tries to advance the general concept of NoC's design, and on the case study demonstrates the performance and behavior of Bonfire NoC. The thesis also presents the results obtained under the different conditions of the NoC's design.

Annotatsioon

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 44 leheküljel, 4 peatükki, 26 joonist, 0 tabelit.

Table of abbrevations and terms

ACK	Acknowledgment
CPU	Central Processing Unit
CTS	Clear to Send
DRTS	Detect Ready to Send
FIFO	First In First Out
GPU	Graphic Processing Unit
ID	Identification number
IP	Intellectual Property
LSB	Least Significant Bit
MSB	Most Significant Bit
NACK	No Acknowledgment
NoC	Network-on-Chip
PE	Processing Element
PI	Packet Injection Rate
RUTPG	Random Uniform Traffic Pattern Generator
SAF	Store-and-Forward
SoC	System-on-Chip
VC	Virtual Channel
VCT	Virtual Cut Through
VLSI	Very Large Scale Integration
WS	Wormhole Switching
QoS	Quality of Service

Contents

List of Figures

List of Tables

1. Introduction

Nowadays a System-on-Chip (SoC) has become a concept for designing of modern integrated circuits where the entire system is implemented to a single die. As Moore's law constantly forces to the density of integration of components on a single chip. Hence, in circuits integrating more and more components on a chip, on-chip communication has become one of the key factors for the overall cost and performance. Therefore, as the communication medium in most of the modern SoCs a global shared bus is used. A bus concept despite all of its advantages like simple structure, scalability and low area, also constitutes the bottleneck in on-chip communication. As the overall communication is efficient only up to ten components in the bus. Thus, in multi-core based SoCs the main challenge that designers face nowadays is designing of scalable, reusable and high performance communication backbone [14].

The paradigm of Network-on-Chip (NoC) which overcomes aforementioned communication bottlenecks in multi-core based SoCs has emerged more than 20 years ago. The concept constitutes the idea of interconnection networks where the components communicate with each other by using of routed, packet-switched network. Thus, also for the sake of NoC advantages like scalability, re-usability and high performance by which NoC overcomes traditional buses. NoC concept becomes researched and developed topic which involves hardware communication infrastructure design, software and operating systems services, comupter aided design (CAD) tools for NoC synthesis, NoC testing etc. [14].

The aim of the thesis: As a performance belongs to the key factors of NoC design. This thesis deals with performance analysis of Bonfire NoC designed in TUT university. In conducted study case the throughput and the average delay factors of Bonfire NoC are measured and analyzed under two conditions of the network's design. The final results of the case study present the performance and behavior of Bonfire NoC depending on applied traffic. Furthermore, they also present the effect of increased size of FIFO buffers on the performance and behavior of the network.

The second chapter of the thesis tries to bring the overall concept of NoC more in detail. It informs about essential NoC aspects as topology, switching, routing, flow control and performance.

The third chapter represents a case study which deals with the performance analysis of Bonfire NoC, and it is a subjective contribution of the author. In the first section of this chapter, the basic topology concept and communication in Bonfire NoC is described.

Finally, in the last section the author presents and justifies the results of the performance analysis of Bonfire NoC.

2. Network on Chip

Network-on-Chip (NoC) shown in Figure 1 is a communication infrastructure that interconnects components by utilizing of routers and channels on a chip. Routers in combination with components constitute nodes in NoC. Thus, the node in NoC can be composed of combination of one component like Processing Elements (PEs), Central Processing Units (CPUs), Digital Signal Processors (DSPs), custom Intellectual Properties (IPs), etc., and memory elements (embedded memory blocks) and router. Routers are interconnected by using of channels in the network. Channels in NoC are pairs of opposite, unidirectional, point-to-point buses. The data in NoC is transferred among the network's nodes . Thus, communication is performed by sending and receiving data packets among nodes within the network. Every component in NoC is connected to its router via Network Interface. (NI). The network interface packetizes and unpacketizes data either to forward them to the network or to its local component. [8] [18] .

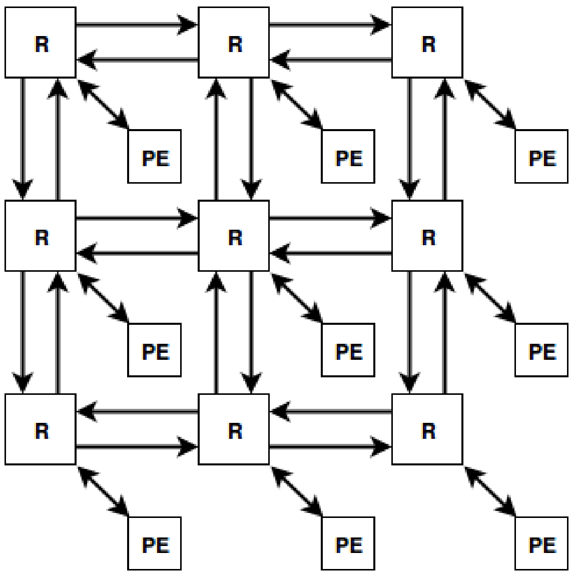


Figure 1. NoC mesh topology

Data packets in NoC are units of communication containing the destination address and sequencing information. The messages from PE are decomposed in NI into packets, and then in turn sent to the network. Such a data packet in Figure 2 consists of a header flit, tail flit and number of body flits in between. After generating a data packet, its flits are routed in a hop-by-hop way from one router to the neighboring router until they reach the destination. The information about the data packet like source and destination address, length of the packet, packet ID, etc. is stored in the header flit of an appropriate packet. Therefore, body flits and tail flit of the same packet follow the header flit during the packet's

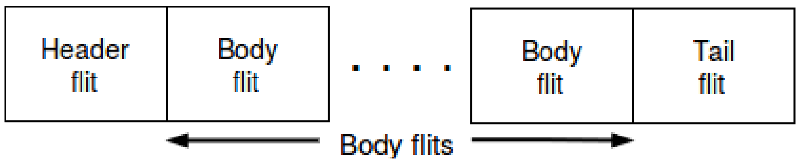


Figure 2. Data packet

transfer. [9] [20] .

Routing of packets within the network fabric is performed by routers. Each NoC router (see Figure 3) incorporates five input and output ports, where four input/output ports correspond to the north, east, south, and west direction. These ports connect the router through network channels to neighboring routers in dedicated coordinates. Whereas fifth port (local) connects the router to the PE. The function of the router is to forward input flits entering the router to their appropriate output towards the final destination. To realize this function a router uses an input buffer for each port in association with 5x5 crossbar switch and dedicated control logic. A crossbar switch within a router redirects traffic to the designated output port, while the control logic ensures the correctness of routing decisions [20].

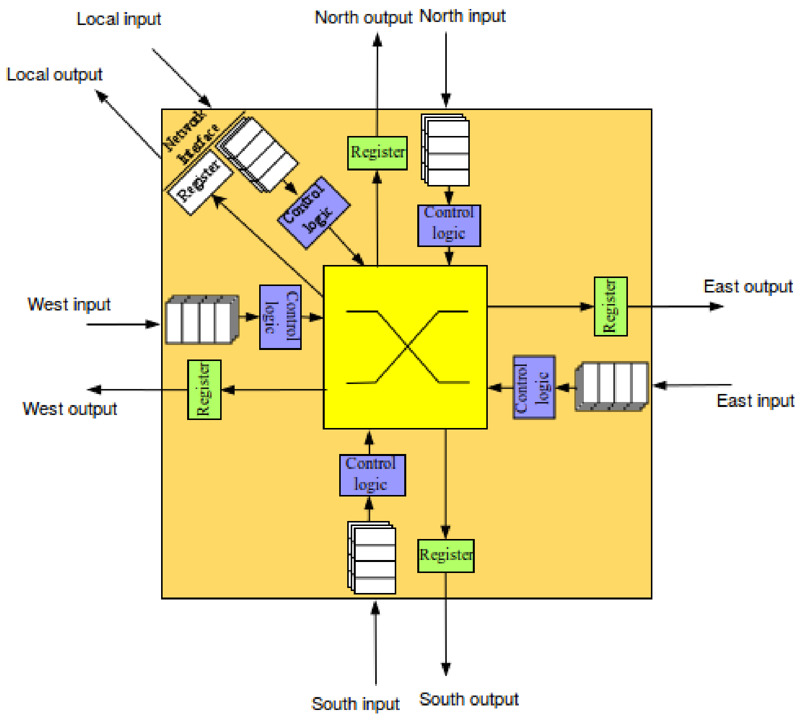


Figure 3. Typical NoC router

NoC architecture may fall into the group of direct or point-to-point networks whose routers are directly connected to each other. These types of networks offer an advantage of scalability to a large number of nodes. Direct networks are traditionally modeled by means of a graph $G(N, C)$ where N represents the set of nodes and C represents the set of communication channels. This representation of direct network fabric is shown in Figure 1 above. Direct network type is primarily characterized by three aspects: topology, routing, and switching. However, flow control which designates access of messages to particular network resources over time also plays significant role in direct networks[8] [9]. These four factors are mentioned in the following parts of this chapter.

2.1. Topology

As may be obvious from a NoC fabric the key to the efficiency of interconnection networks comes from the fact that communication resources (routers, channels) are shared. It means that instead of utilization of dedicated channel between each pair of nodes in NoC, the fabric implements set of shared network routers interconnected by shared channels. [8].

In other words a topology defines the structure of the network and organization of its nodes by determining the connections between them. So far, researchers suggest the utilization of various fundamental topologies such as bus, star, mesh, point-to-point, as well as hierarchical topologies. Unlike fundamental topologies, hierarchical topologies allow the implementation of different topologies locally and globally (for example, implementation of local bus topology and global mesh topology)[18].

2.1.1. Regular topology

One of the classifications of NoC topologies represents network's regularity. Thus, according to regularity classification, the network's structure is characterized either as regular or irregular. The regular NoC topology in Figure 4 constitutes uniform arrangement of routers, which among other things directs to lower design time and cost. Furthermore, the regular topology is largely suitable for utilization in general purpose system on chip (SoC) systems because of its matching the silicon surface. [18] The mesh regular topology is also used in analyzed Bonfire NoC.

However, in spite of the aforementioned advantages, regular topologies are not widely used in commercial products because they introduce a number of drawbacks. These drawbacks are primarily caused by non-optimal utilization of an interconnection network which results in increased latency and power consumption [18].

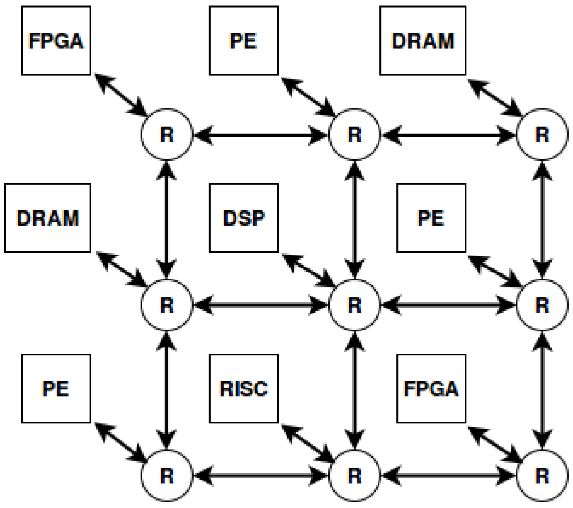


Figure 4. NoC regular topology

2.1.2. Direct topology

Another classification of NoC topologies divides topologies into two groups, direct topologies and indirect topologies. In the direct topology in Figure 5 every node has a direct point-to-point connection with other neighboring nodes in the topology. Most of direct topologies implement orthogonal layout with nodes usually arranged in an n-dimensional orthogonal space. These topologies usually direct to higher availability of communication bandwidth, however they also constitute a growth of the nodes in the network. Thus, in such networks a basic trade-off between connectivity and cost arises. There are number of products which implement direct topology among them are the Nostrum [19], SoCBus [21], Proteo [17], Octagon [12], etc. [9] [18]. In the following section are mentioned some instances of a direct orthogonal topology like the n-dimension mesh, the torus, the folded torus.

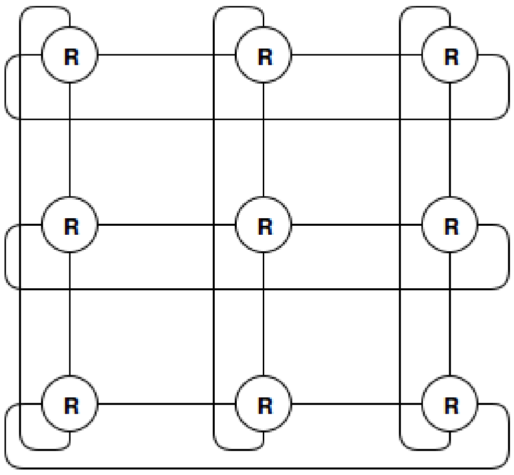


Figure 5. Direct topology

2.1.3. 2-Dimensional NoC topologies

The 2-D mesh topology shown in Figure 6 is the most popular and simplest topology for NoCs. It is composed of $M\ x\ N$ mesh of routers which interconnect nodes (PEs, memories, etc.) situated together with the routers. Each router, besides edge routers are connected to four neighboring routers in the north, the east, the south, and the west direction, and to the local node. Therefore, the number of routers in the network is equal to the number of nodes [18].

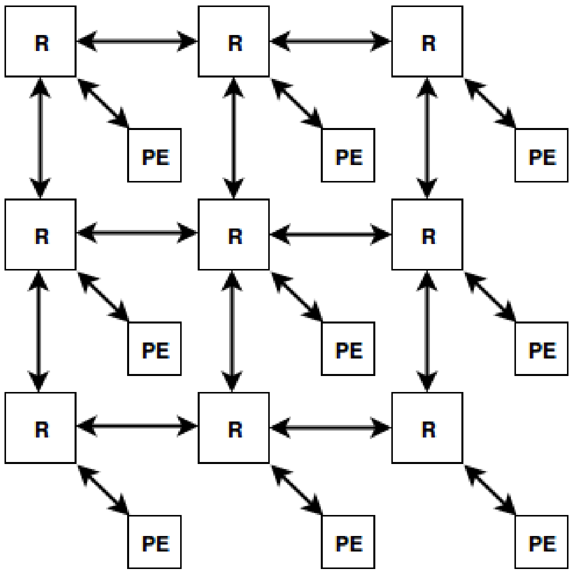


Figure 6. 2-D mesh topology

The 2-D mesh topology implies that length of the links between network's elements is the same, therefore it introduces a regularity of the topology which facilitates the design of the topology significantly. This also simplifies the calculation of the area requirements as the network grows nearly linearly with the rise of the number of nodes. On the other hand, despite of its advantages, the topology also constitutes some drawbacks. An excess of routers implemented in this topology usually directs to congested regions within the topology. Thus, from this reason more careful design and network mapping has to be considered to avoid traffic congestion primarily in the center of the mesh[18].

Good example of 2-D mesh topology can be found in the Chip-Level Integration of Communicating Heterogeneous Elements (CLICHE) NoC [13] [18].

Torus topology is the next direct topology that is implemented by n-dimensional grid with k nodes in every dimension. The torus topology depicted in Figure 7 implements the same layout of a regular mesh as 2-D mesh except for edge routers which are connected by wrap-around channels to the routers at the opposite edge of the network. Thus, timing in the torus topology can be affected by length of the wrap-around connection. This may significantly increase delay depending on the size of architecture. Hence, the length, delay and power consumption should be considered by the physical design to avoid timing constraint issues [18].

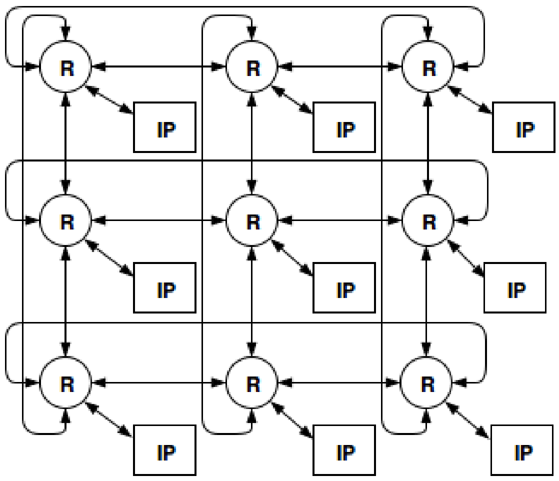


Figure 7. 2-D torus topology

Another topology among direct topologies is the folded torus topology in Figure 8 which is an extension of torus topology. The pattern of the folded torus topology in contrast to the regular torus mitigates the limitation of excessive delays caused by wrap-around channels. As the layout of the topology is regular, it implies that all the channels have the same length and thereby delay. Therefore, the routers in the topology can designate a shortest path with the same effort [18].

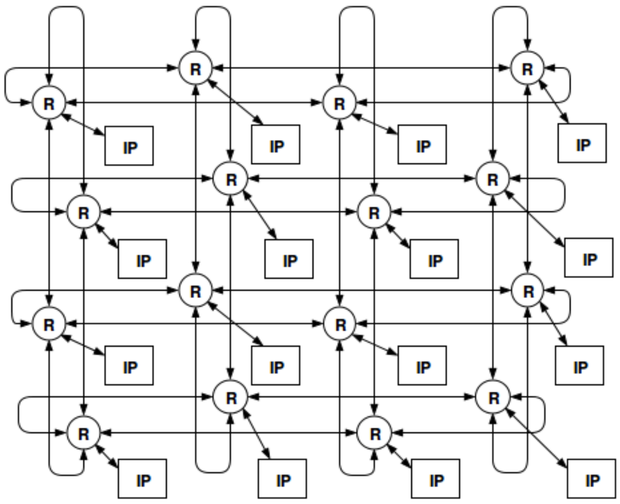


Figure 8. 2-D folded torus topology

2.2. Switching

Once we know about the topology structures in NoC we will need to consider the mechanisms necessary to send our messages from the source nodes to the destination nodes. In this case, it is useful to divide NoC communication to different layers. Physical layer which manages a transfer of messages and controls physical channels between neighboring routers. Switching layer exploiting physical layer protocols to implement forwarding mechanism for sending messages through the network. This layer is discussed more in detail within this section. Finally, the routing layer which chooses appropriate output channels and establishes routing paths between the nodes within the network[9]. The routing layer will be discussed in the following chapter.

2.2.1. Fundamentals of switching

Switching layer utilizes many different switching techniques, which vary in several aspects. The switching techniques designate the method in which the internal input and output ports are interconnected within the router. Moreover, they determine the time when the packets of a message can be transferred through this path. These techniques are bound with flow control mechanism for synchronizing packet transfers within and between the routers in the network. Flow control is bound with buffer management algorithms that determine the way in which messages are handled when they are blocked in the network. Thus, implementing of the switching layer varies according to decisions made in each of these areas [9].

Flow control is a protocol for synchronizing the transfer of data units within the network. The data unit is basically the smallest unit of information transferred between the sender and the receiver through the network. The data unit is requested by the sender and acknowledged by the receiver. Thus, these request/acknowledgment signals ensure a successful data transfer and signalize that the receiver's buffer is ready to receive new data[9].

Flow control may be executed at two levels. At the first level, as the message flow control, which runs at the level of an entire packet. However, since the transfer of the whole packet through the physical channel may require a substantial number of clock cycles. Therefore, the actual multicycle transfer exploits the physical channel flow control to transfer message flow control units through the channel between the routers[9].

Switching techniques can vary in terms of sizes between the physical and message control units. Essentially, every message can be divided into the fixed-length units called packets. Then packets can be split into the smaller message flow control units named flits. Depending on the channel width, it may be necessary to use multiple physical channel cycles to transfer one flit through the channel. Eventually, flits can be split to the smallest units of information called phits see Figure 9. Phits can be transferred in single cycle over the physical channel. If flits constitute logical units of information, thus phits represent physical quantities of information. This means, one phit presents the number of bits we can transfer in parallel within one cycle. In many NoC implementations the size of phits equals to the size of flits[9].

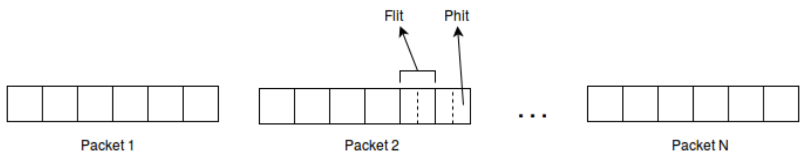


Figure 9. Message control units

2.2.2. Essential switching techniques

Nowadays, there are used several switching techniques in interconnection networks. Among these techniques the most common are circuit switching, packet switching, virtual cut-through switching, and wormhole switching [3]. We describe these techniques in this section.

For the reason of comparison of these techniques, we have to clarify some facts we will adhere to. The flit size is the same as the phit size and equal to the width of the physical channel in terms of bits. The routing header is the size of one flit [9].

The circuit switching technique belongs to connection-oriented schemes which reserve dedicated physical channels for packet transfer before actual communication will take place. The circuit switching uses routing header flit to reserve physical path between the source and the destination node beforehand. Hence, probing header flit is sent to the network before the actual data transfer will take place. This flit contains the address of the destination node and some additional control data. As this probing flit advances through intermediate routers towards the destination address. It reserves the physical channels between these adjacent routers until it reaches the destination node. After reaching of the destination node a complete path is established. Then the destination node sends back an acknowledgment. Finally, after receiving an acknowledgment an actual data transfer can be transmitted. As the overall physical path between the source and the destination node is reserved. The data is transmitted at the full bandwidth of the path. The established path can be released either by the destination node or by the last flit of the message.

Circuit switching technique has several advantages. It is technique that is very easy to implement in NoC. It also offers better performance for transferring of large packets than packet switching techniques. However, on the other hand transfers of small packets are not so efficient, thus in this case the packet switching is preferred. [9] [15] [20].

Unlike circuit switching technique where the intact payload is transferred after setting up the path between the source and the destination node. At the packet switching data is transferred by fixed-length packets which are individually transferred through the network. The transfer starts by sending header flit through the dedicated output of a router. Thus, this flit reserves the output only for the flits of the designated packet. After transmission's start, complete packets have to be received and stored in each intermediate router along the path before they can be forwarded to the next router. Hence, this technique is also called Store-and-Forward (SAF). In this way packets are forwarded until they reach the destination node. Packet switching technique allows many packets of the same message to reside in the network simultaneously. Regardless of the fact whether the first packet has arrived at the destination node. As, this technique uses flow control at the level of flits rather than the level of packets its performance turns out to be better by the transfer of short and frequent messages. On the other hand, the disadvantage of packet switching arises in a case that header flit is blocked. Then the trailing flits may remain spread over the intermediate routers along the path causing the blocking condition named deadlock. Moreover, another issue constitutes excessive latency of the packet transfer. This becomes directly proportional to the distance between the source and the destination node. Its value may depend on routing algorithms and implementation of switches in NoC. [9] [18] [22].

Since packet switching requires receiving and storing of the whole packet at the router before its forwarding to the next intermediate router. This can cause extended per-router latency and deadlocks primarily by transmission of large packets. Virtual cut-through (VCT) switching technique uses only header flit to designate the output direction, and then it can immediately forward the packet to the next router. Thus, the entire forward packet does not have to be stored at the router's buffer. Flits can be instantaneously sent to the next router when that offers free buffer to receive them. In other words, flits can

cut through the local router after the routing decision. However, in a case of congestion in NoC the router has to be able to store the whole packet. In this way VCT behaves as packet switching. From this behavior we can observe that the routing works at the flit level and routing data is contained in one flit. Thus we can also suppose there is no additional latency for cutting through a router when the output channel is free. However, the unit message flow control is still a packet as at store-and-forward switching [9] [18].

In contrast to previous two switching techniques which use packets as the units of a message flow control ,wormhole switching (WS) technique uses flits. That means that flit becomes the unit of a message flow control. Thus, WS allows to span packets through the network at the flits level. From this reason the buffer requirements in the routers are smaller than for VCT just for storing of several flits. The advantage of WS is that this technique requires implementation of relatively small buffers for storing data. As buffers are quite expensive in NoC in terms of resources, WS allows more economical utilization of resources at the target platform. Hence, WS has become most implemented packet switching method in NoC. WS obtains the same latency as virtual cut-through in a case they are not blocked packets in the network. On the other hand, in a case of blocked packets within the network. Their flits are spread over the several routers what can lead to deadlocks in the network. This can be disadvantage of this approach [9] [18].

2.3. Routing

As NoC generally implements distinct digital circuits like CPUs, PEs, IP circuits, memories etc. It arises the requirement to use different routing algorithms for an optimal communication. Thus, for such a purpose several routing algorithms have been designed which cover functionality in the routing layer [16].

Each of directing nodes in NoC is a router. The routers can receive data packets to their input ports and forward them out their output ports according to routing algorithm. Thus, routing algorithm sets up the path along which every packet is transferred. Moreover, routing algorithms play an important role in NoC. Since they have direct impact to many of its properties. Some of these properties are: [8] [9]:

- Connectivity. It is the ability of the network to direct packets from any source to any destination node within the network.
- Adaptivity. It is the ability allowing to route packets through variant paths in a case of congestion or faults in the designated paths.
- Deadlock and livelock freedom. This ability ensures free packet transfers without unwanted blocks and infinite wandering within the network.

- Fault tolerance. It is the ability allowing to direct packets in the presence of the damaged elements. Even though it seems that adaptivity covers the same functionality.

  Fault tolerance can be obtained without adaptivity. Fault tolerance requires additional circuitry to be implemented in NoC.

2.3.1. Routing algorithms

We can classify routing algorithms in NoC according to many criteria. One of the first classifications is associated with the number of destinations in NoC. According to this classification, packets can be routed to a single destination (unicast routing) or multiple destination (multicast routing) [9]. In this chapter only unicast routing algorithms are discussed. As, these algorithms are implemented in Bonfire NoC.

Another criterion for calculating routing algorithms is the place where routing decision occurs. According to this classification, routing algorithms can be considered either source routing or distributed routing algorithms. While decisions for source routing are all calculated at the source node and the destination's address is finally appended to the packet. Thus, the routers do not have to make any routing decisions. They just establish the routing path according to data stored in the packet. In distributed routing, each router designates routing path in hop-by-hop manner. This means that every intermediate router makes routing decision according to its awareness of the local network. This way is even implemented for deterministic routing [18].

Routing algorithms may be implemented by using of several mechanisms. Among the most used mechanisms belong the routing table (table lookup) and finite-state machine. Both of these mechanisms can implement either deterministic or adaptive routing algorithms. Routing table is used by routers either at the source or at every hop router across the routing path for implementation of routing algorithms. In a case of deterministic routing, a table is constrained to a single record per destination. However for adaptive routing, several records per destination can be stored in the routing table. The primary advantage of the routing table is its universality. The second routing mechanism uses algorithmic routing. This mechanism can implement finite-state machine either in hardware or software way. Finite-state machine calculates the path or next hop router for a packet at run-time. Algorithmic routing is usually used for simple algorithms and regular topologies. Even though, it does not provide such generality as the routing table. It is more efficient both in speed and area than routing table [8] [9].

Adaptivity is the capability of the network that allows the packets to be routed through different paths. Thus according to adaptivity, the routing algorithms can be divided into three categories: deterministic, oblivious and adaptive routing algorithms. In deterministic routing, packets are always transferred along the same specific path from the source to the destination. It means deterministic routing algorithms calculate the path according to destination's address. In oblivious routing which is sometimes considered to be akin to deterministic routing ,the routing decision is independent of the state of the network. The routing path is chosen either in random or cyclical way. On the other hand, in adaptive routing the decisions for making routing path are computed dynamically based on the actual state of the network. Thus, routing path is chosen at run-time to avoid congestion and faulty areas in the network [9] [14].

Nowadays, regular topology is largely used in NoC. This is not only because of its architecture which is easy to implement. Furthermore, this architecture is suitable for utilizing of simple deterministic routing algorithms. One of the most preferred regular topology became orthogonal topology like mesh (also used in Bonfire NoC implementation), torus, and hypercube. This topology allows simple calculations of distances between the nodes based on the sum of the offsets in individual dimensions. Such computations are known as dimension-order routing algorithm. Dimension-order routing algorithm reduces the offset in one dimension to zero, and after reaching targeted offset in one dimension, it proceeds to the next dimension. Thus, dimension-order routing routes the packets first in the X-coordinate until it achieves the same value of the X-coordinate as it is stored in the destination address. This means, the offset in the X-coordinate is set to zero. After that, it crosses to the next coordinate and moves the same way. Crossing of dimensions in dimension-order routing is strictly done either in increasing or decreasing order. Thus, it ensures avoiding of deadlock and livelock in the network. The advantages of this algorithm are. It provides the shortest path which reduces the energy consumption during the data transfer. As this algorithm does not use routing table, it also reduces the number of implemented gates in comparison with algorithms which use [8] [18]. In Algorithm 1 is shown algorithmic implementation of dimension-order routing algorithm for 2-D mesh topology.

The next significant category of routing algorithms according to adaptivity are adaptive routing algorithms. In adaptive routing algorithms decisions to establish routing paths are made dynamically according to actual state of the network. Thus, alternative paths are selected on the fly to avoid congested and faulty areas of the network. However, the usage of alternative paths in adaptive routing implies that in-order delivery of the packets

Dimension-order routing for 2-D meshes

```
Inputs : current node coordinates Xcurrent, Ycurrent
         destination node coordinates (Xdest, Ydest)
Output : selected output channel
Procedure:
Xoffset := Xdest - Xcurrent;
Yoffset := Ydest - Ycurrent;

if Xoffset < 0 then
    channel := X-;
end
if Xoffset > 0 then
    channel := X+;
end
if Xoffset = 0 & Yoffset < 0 then
    channel := Y-;
end
if Xoffset = 0 & Yoffset > 0 then
    channel := Y+;
end
if Xoffset = 0 & Yoffset = 0 then
    channel := destination;
end
```

Algorithm 1: Dimension-order routing for 2-D meshes

at the destination does not have to be assured. Thus, the reordering of the packets at the destination NI has to be performed. We can further classify adaptive routing algorithms according to their progressiveness as progressive and backtracking algorithms. While progressive routing algorithms only send the header forward and reserve new channel at each routing step. Backtracking algorithms also allow the header to backtrack and release thus already reserved channels. Therefore, they are primarily used for fault-tolerant

routing. Finally, we can classify adaptive routing algorithms according to the number of alternative routing paths as fully adaptive and partially adaptive routing algorithms [9] [14].

Partially adaptive algorithms constitute a trade-off between flexibility and cost. Since, they offer the flexibility of fully adaptive routing at a moderate increase of complexity with regard to deterministic routing. Partially adaptive routing algorithms are based on the ability to constrain particular turns in the network. Thus, they are also known as turn models. Turn models basically prohibit the smallest number of turns which might create cycles by moving between dimensions. Thus, they help to avoid deadlocks in the network.

However, the choice of blocked turns cannot be random because for some restricted turns deadlock is still possible as proposed in [7]. Turn models are named according to directions in which they either start or finish packet transfer. These are West-first, North-last, and Negative-first turn models. In West-first turn model the packet is directed to the west direction first if this is possible otherwise, the packet is directed adaptively in the shortest path. Thus, packet must forward through all of its hops in the west direction before heading in any other directions. As we can see in Figure 10 West-first turn model implies that both turns in which packet transfer ends in the west direction must be prohibited. Besides, West-first turn model there are other turn models North-last and Negative-first which work on the same principle. In North-last, both turns in which packet is routed first to the north direction are prohibited see in Figure 11. Thus, packet can be directed in any free direction besides north. Since a packet gets to the north direction it must continue in it until it reaches the destination. In Negative-first turn model packet is directed in negative coordinates first before heading to any positive direction see in Figure 12. Thus, once a packet is in positive direction it has to forward until it reaches the destination node [8] [9] [18].

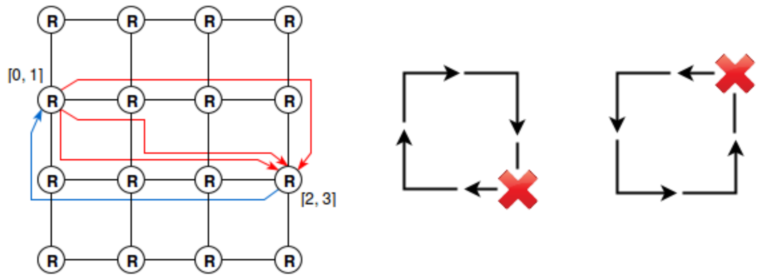


Figure 10. West-first turn model

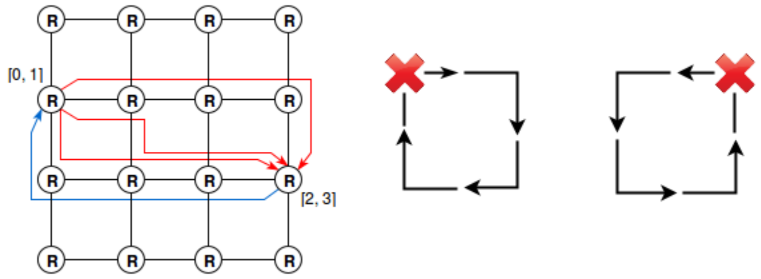


Figure 11. North-last turn model

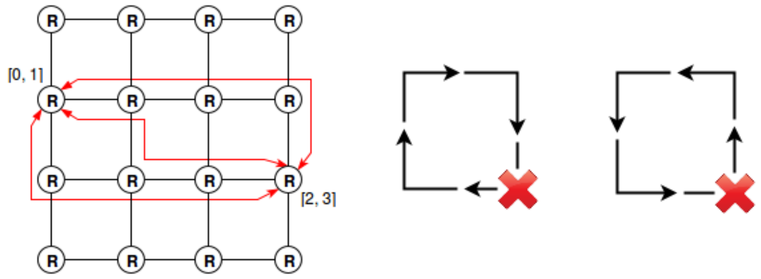


Figure 12. Negative-first turn model

Except of aforementioned turn models another partially adaptive algorithms have been proposed. A different approach of partially deadlock-free adaptive algorithm has been proposed by [6]. This approach presents adaptive wormhole routing algorithm for meshes without utilization of virtual channels (VC). Here the author suggests some restricted locations in the network in which turns can be taken out. Thus, the deadlock may be avoided. The benefit of this method is higher communication efficiency mainly in mesh networks. Another approach which integrates advantages of both deterministic and adaptive routing algorithms has been proposed by [11]. This algorithm (DyAD) utilizes routing technique that switches between deterministic and adaptive routing. This is done in routers according to information about the state of the network. Thus, if the network is congested DyAD routers work in adaptive mode to avoid congested channels by using alternative paths. On the other hand, in the network without the congestion DyAD routers work in deterministic mode. Thus, they exploit low routing delay provided by deterministic routing algorithm. Moreover, for n-dimensional meshes and hypercubes planar-adaptive routing method has been proposed by Chien and Kim [5]. In this method a routing adaptivity is provided in only two dimensions at a time. Thus, a packet is directed in a series of 2-D planes. As the packet forwards to its destination routing dimensions are also changed.

2.4. Flow control

As the NoC architecture is basically shared medium, it is necessary to decide how the resources will be allocated for an effective utilization of the bandwidth. For the purpose of such decisions contention protocols are used. Thus, flow control method referring to contention protocols allocates resources to packets in the network. In other words flow control may be considered to be a contention resolution problem. Flow control generally allocates resources as a channel bandwidth, buffer capacity, and control state. Although all flow control methods allocate control state and channel bandwidth, some of them like bufferless methods do not allocate buffers. While good flow control allocates these resources in an efficient way. This means, an effective allocation of the bandwidth with low predictable transfer latency. On the other hand, poor flow control can waste bandwidth by utilizing resources an inefficient way and leaving them idle [8] [18].

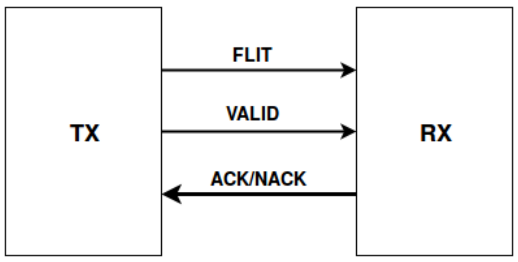
Flow control may be divided into two categories. The first is centralized flow control when the central controller makes the decisions, and the second is distributed flow control when decisions are made in each router. Nowadays, distributed flow control is the most common used method in NoC. Furthemore, flow control can be classified as intra-switch, switch to switch, and end to end. Subsequently, we can classify switch to switch flow control according to buffers utilization as bufferless and buffered flow control. The first named bufferless flow control is the simplest flow control method. As its name implies in this method routers do not have implemented any buffers. Thus, the packets are either dropped and re-transmitted or deflected if they do not have free channels. In deflection routing method ("hot potato") which has been first introduced by [2] all packets are directed to an arbitrary output port of the router. Regardless of the fact, whether the port is in the productive direction (the routing direction that reduces distance to the destination) or not. Thus it may happen that packets are deflected, since the output port in the productive direction can be already allocated to another packet. Concerning the fact there are not used any buffers at the bufferless flow control. It gives significant reduction in energy consumption by a small latency increase in contrast to buffered flow control [8] [18].

In terms of complexity and efficiency we can achieve higher level of flow control by using of circuit switching method. In this flow control method only header of packets are buffered. Hence, only the packet header is forwarded before a transfer of the payload to reserve dedicated resources along the path. If happens that the header cannot allocate resources in a given node at once, it waits at that node until the resources are released [8] [18]. Circuit switching has been described more in detail in the section 2.2.2.

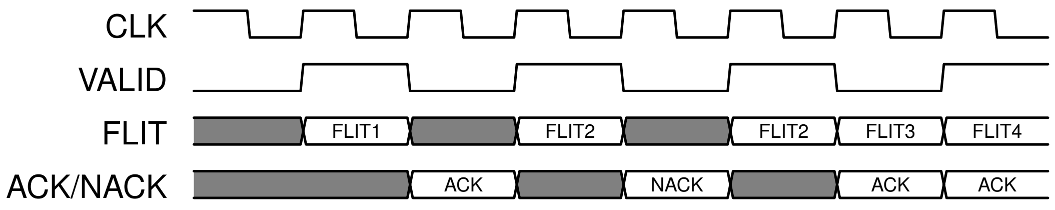
More efficient flow control arises by full utilization of buffers. Since buffers separate allocation of neighboring channels. Channels do not have to be allocated within successive cycles and packets do not have to be dropped or deflected. Thus, adding buffers provides a space for storing the packet (flit), and allows the other channel to be allocated with a delay without causing any complications. After adding buffers the flow control has to allocate both buffers as well as channel bandwidth. Moreover, by adding buffers flow control also offers the selection of buffering level. Buffers may be implemented either at the level of packet units or at the finer level of flit units. While packet units level control is implemented in SAF and VCT techniques, where flow control occurs over packets which are basic units. Flit units level control is used in WS or VC techniques where flow control runs over flits. However, the advantage is at the side of flit units flow control, where the storage at each node can be dramatically decreased by breaking large packets into smaller flits. Furthermore, the utilization of flits helps to create multiple VCs per physical channel which can mitigate blocking and rise up the throughput in the network [8].

Handshaking protocol depicted in Figure 13a is simple buffered flow control method used in NoCs. This flow control method is based on acknowledged/not acknowledged signals which are exchanged between two adjacent nodes by communication. Handshaking flow control works in such a way that sender sets data on the channel and at the same time activates the VALID signal. Thus, if the receiver is ready to receive the valid data it sends the ACK signal. However, if the data is corrupted or there is no enough space in a receiver's buffer, receiver responds with NACK signal. In such a case that the sender receives NACK signal it re-sends the flits again starting from not acknowledged one. Shown in Figure 13b The advantage of this flow control protocol is supporting of fault tolerance. On the other hand, the drawback of this flow control is the additional space for buffers which have to keep sent flits in a case when re-transmission is necessary [18].

Among other similar flow control methods based on handshake protocol belong the flow control method NoCGEN presented by [4] . This method uses three handshake signals request, grant and ready for enabling flow control on point-to-point channels.



(a)



(b)

Figure 13. ACK/NACK handshaking flow control (a) (flow control schematic), (b) (timing diagram)

Another flow control method ON/OFF with its most general implementation in NoC named STALL/GO was presented in [8]. This method uses two handshake wires depicted in Figure 14a. One wire for signalizing that the data is available, and the second one for giving feedback about the state of receiving buffers. Buffers can be either in full state (STALL) or in free state (GO). As it can be seen in timing diagram in Figure 14b, the flit is only sent when the feedback signal is set to GO state. Otherwise, when the feedback signal is set to STALL state the receiving buffers of a consecutive router are full. The feedback signal STALL/GO is set according to threshold which is calculated by availability of the receiving buffers in a consecutive router. The advantage of this method is, that the method significantly reduces the number of upstream signaling in particular cases.[8] [18]

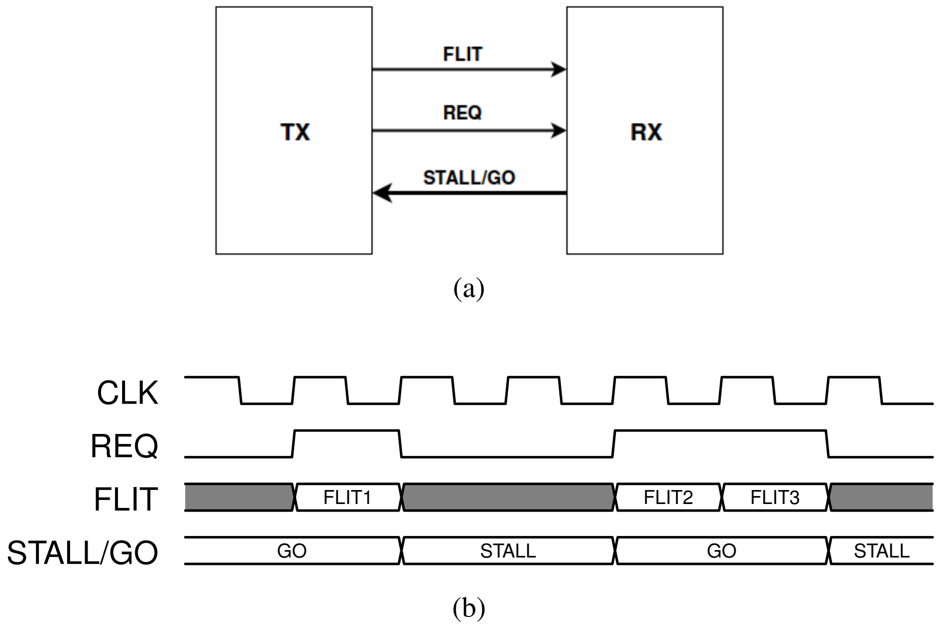


Figure 14. STALL/GO handshaking flow control (a) (flow control schematic), (b) (timing diagram)

2.5. Performance

Before physical implementation of NoC in the target technology we are generally concerned with evaluating of some of its characteristics. These characteristics are primarily cost and performance of NoC.

Performance of NoC can be evaluated by using of two basic quantities: throughput and latency. Throughput of NoC is expressed as the maximum number of information transferred through the network within unit of time. Latency expresses the time which elapses since the message transfer is initiated until the message is completely received at the destination node [9]. The following part of the section describes performance characteristics, which are good to understand for the sake of clarity of the following research covered in the next chapter.

2.5.1. Throughput

As was already mentioned throughput of NoC is the maximum amount of the data transmitted through the network per unit of time. In other words throughput may be specified as the maximum traffic accepted by the network. Where the maximum traffic constitutes the number of information delivered per unit of time. Throughput is a feature of the whole network which depends not only on the topology, however on routing and flow control as well. We can measure throughput either in messages per second or messages per clock cycle. This depends on used timing which can be relative or absolute [8] [9].

The maximum throughput in NoC results from the saturation of any channel in the network. This implies, if there are not any saturated channels in the network. The network may deliver more traffic and does not work at the maximum throughput. For the purpose of throughput calculation at the network a channel load has to be considered. Thus, the channel load γ_c on the channel c, can be defined as the ratio of the channel's c bandwidth to the input ports bandwidth. In other words this ratio represents the number of traffic which has to cross the channel c in a case that every input injects one unit of the traffic according to the desired traffic pattern. As long as it is not specified, channel loads are considered under balanced traffic [8].

In a case of certain traffic pattern, the channel which transfers the largest portion of the traffic designates the $maximum\ channel\ load\ \gamma_{max}$ of the topology, where $\gamma_{max} = max_{c \in C} \gamma_c$. Therefore when the traffic achieves the throughput of the network, the maximum load on the dedicated channel will equal to its bandwidth b. Hence, any additional traffic will overload this channel. Thus, we can define the ideal throughput of the topology Θ_{ideal} as the input bandwidth which saturates the dedicated channel see Equation (1) [8].

$$\Theta_{ideal} = \frac{b}{\gamma_{max}} \tag{1}$$

where b is in bits per second, γ_{max} is unitless.

2.5.2. Latency

The second quantity designating NoC performance is latency. Latency T may be defined as the time necessary to transfer packet from the source to the destination node. This time is measured since the header flit is injected in the input port of a source router to the point when the tail flit leaves the output port of a destination router. We can divide latency T

into two sections as shown in Equation (2).

$$T = T_h + \frac{L}{b} \tag{2}$$

The first section T_h constitutes head latency which is the time necessary for the head of the message to be transferred over the network. The second section expresses the serialization latency T_s shown in Equation (3). Serialization latency represents the time for a packet of the length L to pass through the channel of bandwidth b.

$$T_s = \frac{L}{b} \tag{3}$$

Likewise throughput, latency also depends on routing and flow control, as either on a topology and the design of the router [8].

In a case of any contention in the network, we can express head latency as the sum of two factors which are designated by the topology. The first is router delay T_r, and the second is time of flight T_w. While T_r represents the time which head spends in the routers, T_w constitutes the time head spends on the wires. Thus, we can express the average routing delay $T_r = H_{min} t_r$ as product of average hop count H_{min} in the network and delay t_r in a single router. In the same way we express the average time of flight $T_w = \frac{D_{min}}{v}$ as the ratio of distance D_{min} to propagation rate v in an average network. By combination of these sections we can infer expression for average latency see Equation (4) providing of absence of contention in the network.

$$T_0 = H_{min} t_r + \frac{D_{min}}{v} + \frac{L}{b} \tag{4}$$

These three terms in our equation (4) present the three parts of total latency: switch delay, time of flight and serialization latency. We consider average latency T_0 as the latency at zero load whit no contention in the network. We can add the fourth term T_c to the equation by increasing the load. T_c represents the time at which the message waits for resources [8].

The three critical parameters in the equation (4) are primarily affected by mapping of the network's topology into the physical packaging. Average hop count H_{min} is completely a characteristic of the topology. Average distance D_{min} is influenced by both packaging and topology. Eventually, bandwidth b is designated by the node degree and the packaging restriction [8].

3. Case Study: Performance analysis of Bonfire NoC

This part of master's thesis deals with case study of the interconnected network design developed in Tallinn University of Technology. This NoC design is a part of the project Bonfire which started in 2015. The main idea of this project is to research and design fault-tolerant interconnected network infrastructure. The task of the author of the thesis in the project is measuring, analyzing and evaluating performance of Bonfire NoC. For this purpose the author has designed the random uniform traffic pattern generator for analyzing traffic in Bonfire NoC. In the following chapter the performance analysis of Bonfire NoC is covered, at which the author tries to justify the network's performance behavior.

3.1. Bonfire NoC

The analyzed Bonfire NoC topology is 2x2 mesh topology shown in Figure 15 which is easy to implement because of its regular layout. Furthermore, the other advantage of the mesh structure is its possibility to easily scale the network [13]. In this case study only 2x2 Bonfire NoC topology is analyzed because such a size facilitates representation and understanding of the analyzed data. However, Bonfire NoC design can be scaled up to the higher number of nodes. Scalability of Bonfire NoC is simplified not only by its regular topology, but also by utilizing of the routers with simple unified fabric. Such a unified structure of the router allows to place the router at any coordinates in the network.

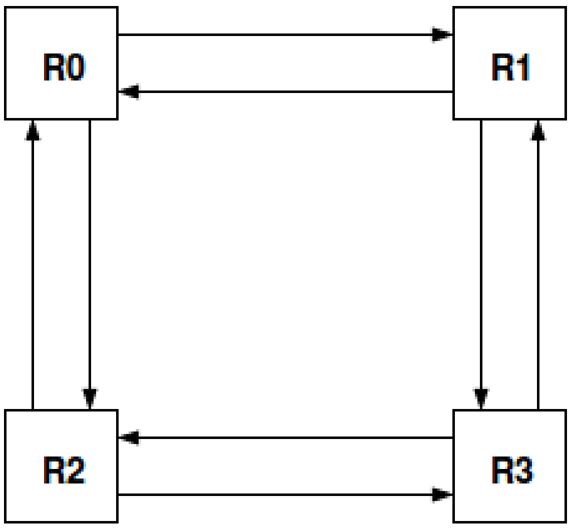


Figure 15. 2x2 Bonfire NoC mesh topology

3.1.1. Communication in Bonfire NoC

In terms of routing, Bonfire NoC uses XY routing algorithm. Individual nodes communicate by transmitting of messages. Messages are decomposed into packets, where every packet consists of three or more 32 bit wide flits. Every packet should consist of at least 3 flits: the header flit, one body flit and the tail flit. The transfer of the packet starts by sending of the header flit. Such a header flit, shown in Figure 16 is decomposed into several bit fields. Starting by the Most Significant Bit (MSB) the first 3 bits denote the flit type. Flit type can be either *header - 001* or *body - 010* or *tail - 100* flit. Following 12 bits determine the length of the packet which can be up to 4096 flits. The next 4 bits representing destination router address are followed by 4 bits representing source router address. In turn, the next 8 bits constitute packet ID which is the number for keeping the order of sent packets within the network. Finally, the Least Significant Bit (LSB) represents the parity bit. Each flit in the packet contains the parity control bit.

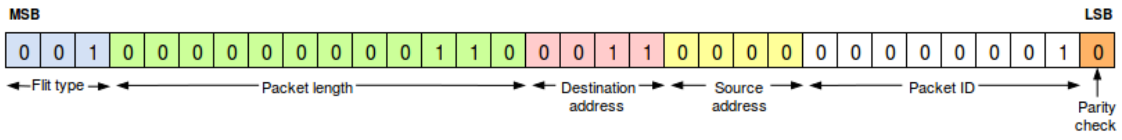


Figure 16. Header flit composition

The packet transfer continues by transmitting one or more body flits. Such a body flit shown in Figure 17 is comprised of 3 bit fields. The first bit field starting from MSB constitutes the flit type. The second bit field represents a payload, which is the actual transferred data. The last bit constitutes parity control bit.

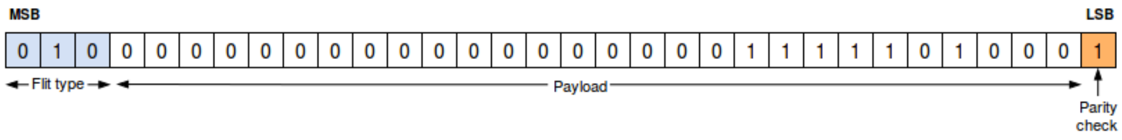


Figure 17. Body flit composition

Finally, every packet transfer ends with a tail flit. Such a tail flit shown in Figure 18 is akin to the body flit. It is also comprised of 3 bits of flit type followed by payload bits and finished with a parity control bit.

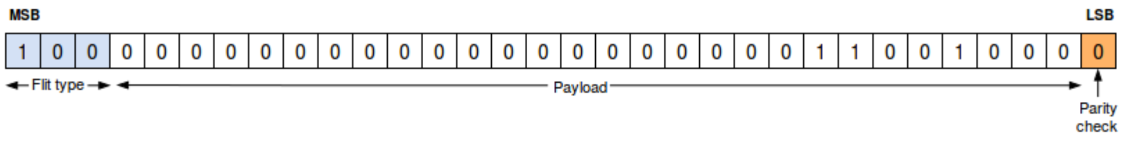


Figure 18. Tail flit composition

For the sake of readability all values in simulation wave-forms are depicted in decimal format. Thus, frames beginning with digit 5 constitute header flits. The frames beginning with with digit 1 represent body flits. Finally, the frames beginning with minus sign represent tail flits.

As depicted in Figure 19, Figure 20 and Figure 21 respectively, the packet is transferred from node 0 to node 3. The header flit 537681922 is injected into the local input port RX_L_0 of router 0 from which it is routed to the west input port RX_W_1 of router 1 as shown in Figure 19. From router 1 the header flit 537681922 is forwarded to its destination router and received on the north input port RX_N_3 of router 3, as depicted in Figure 20. Finally, the header flit is sent to the local output port TX_L_3 of router 3 as shown in Figure 21. In the same manner, the body flits follow the header flit until the tail flit is received at the destination.

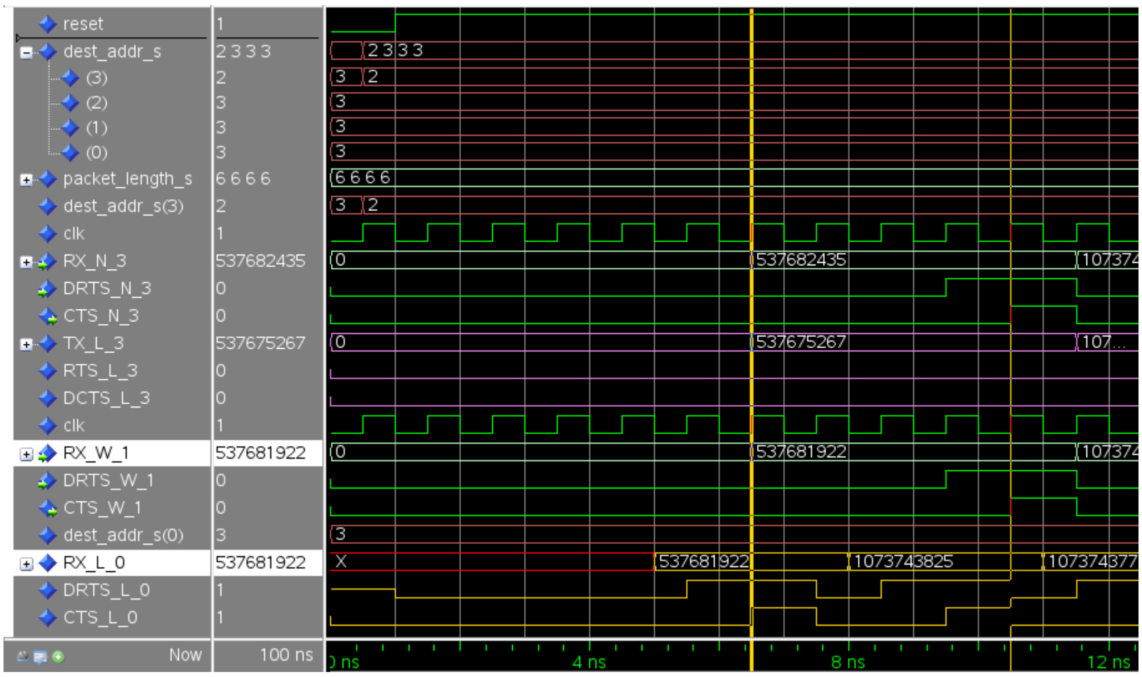


Figure 19. Routing packet from router 0 to router 1

As it can be seen in figures Figure 19 - 21, every accepted flit is acknowledged by handshaking signals Detect Ready to Send (DRTS) and Clear to Send (CTS). Handshaking starts with detection of Ready to Send (RTS) signal, which is sent from the previous router or NI for notifying that the data is valid and ready to be sent. In return, the addressed router or NI acknowledges the request by asserting Clear to Send signal and receiving the data. All valid data transfers in Bonfire NoC are acknowledged by dedicated handshaking.

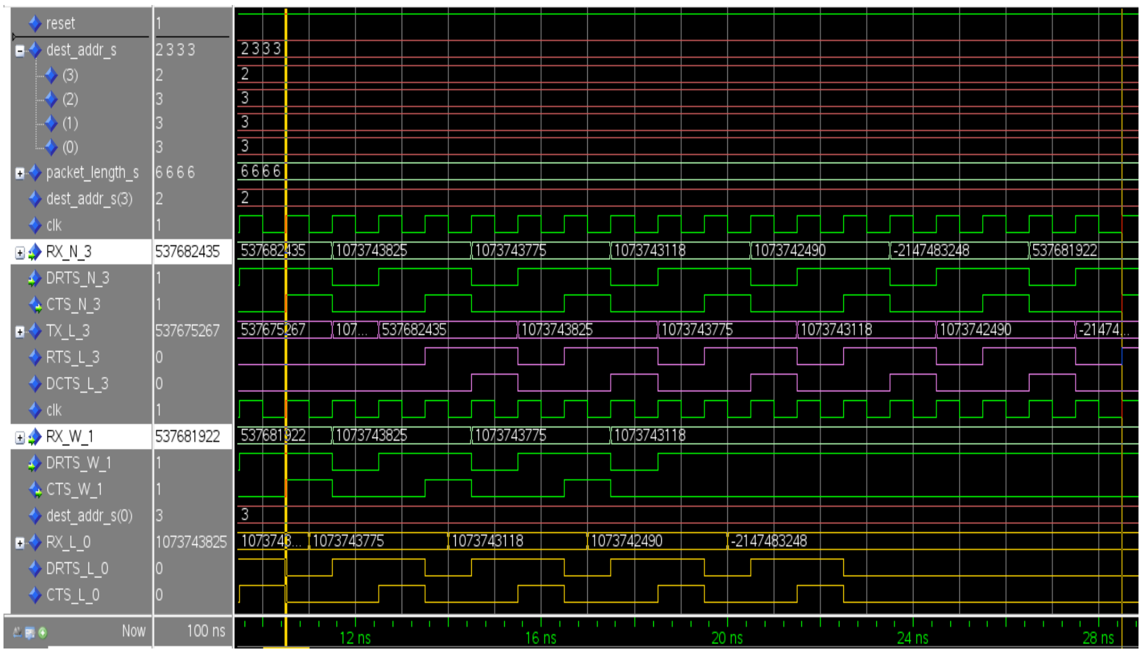


Figure 20. Routing packet from router 1 to router 3

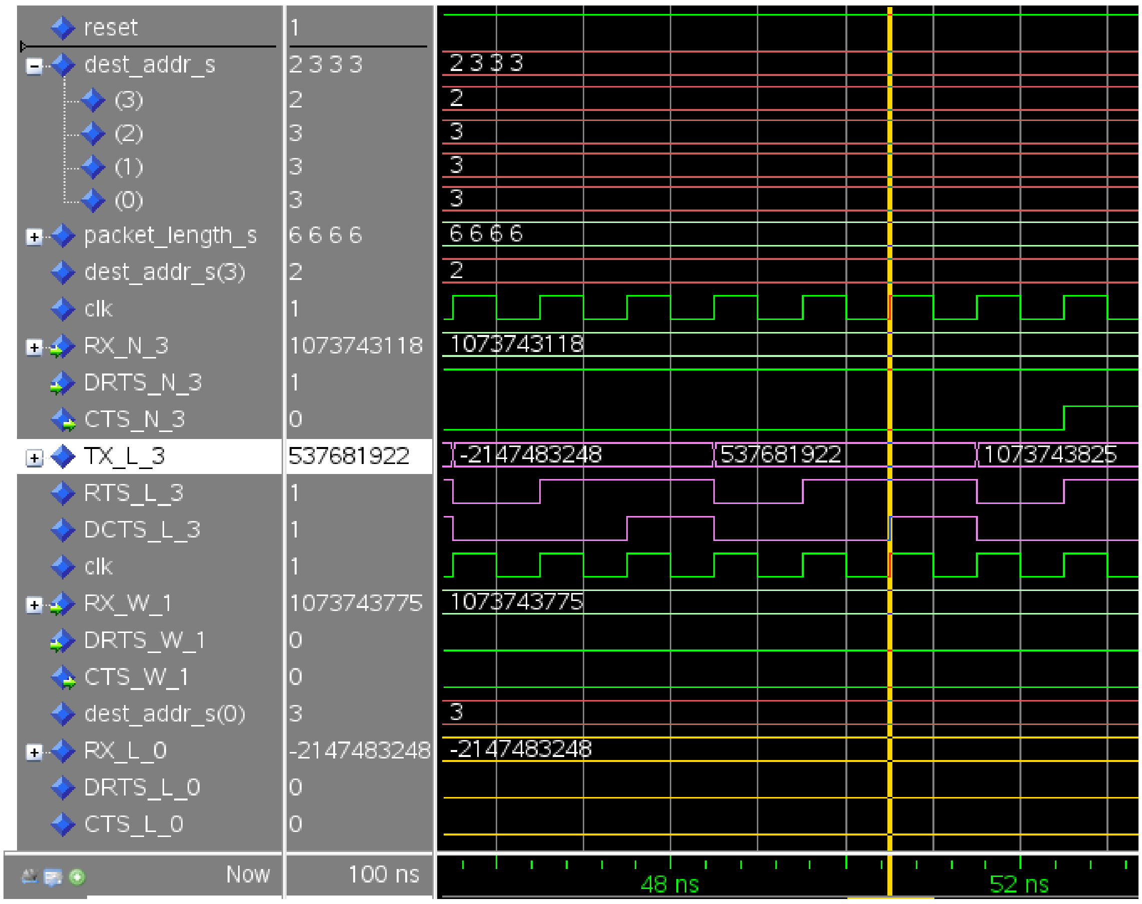


Figure 21. Routing packet to NI at router 3

3.1.2. Bonfire NoC performance analysis

Performance analysis of Bonfire network is tested by using of the Random Uniform Traffic Pattern Generator (RUTPG). The task of the RUTPG is to generate 4 sets of packets of the fixed length $packet\_length\_s$ that are injected into the local input ports RX_L_x (where x constitutes the ID of the router) of the routers in Bonfire NoC and routed to the destination routers $dest\_addr\_s(x)\ y$ (where x constitutes the ID of the source router, and y constitutes the destination address) according to randomly generated addresses. Every new packet is generated within the frame time $FRAME\_TIME$. The frame time is calculated by using of Packet Injection Rate (PIR) [1]. During each frame time the packet is generated and injected into Bonfire NoC after running out of the initial delay $init\_delay\_s$. For example: The RUTPG generates a set of 4 packets each of length 6 flits after elapsing of initial delay 4 cycles within the frame time 100 cycles. Subsequently, the packets would be routed from source nodes 0 to 3 to destination nodes 3, 3, 3, 2 respectively as depicted in Figure 22.

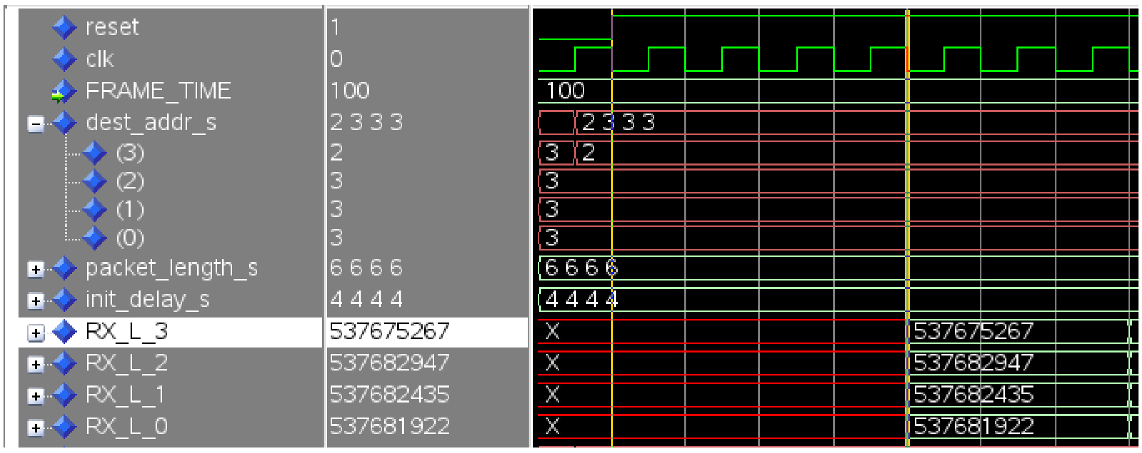

```
reset          1
clk            0
FRAME_TIME     100
dest_addr_s    2 3 3 3
  (3)          2
  (2)          3
  (1)          3
  (0)          3
packet_length_s 6 6 6 6
init_delay_s   4 4 4 4
RX_L_3         537675267
RX_L_2         537682947
RX_L_1         537682435
RX_L_0         537681922
```

Figure 22. Initial setting of RUTPG

The performance analysis of Bonfire NoC is conducted at two conditions of router's design. In the first condition, the size of a FIFO buffer is set to 4 and maintains this uniform value during all tests. While in the second condition, the size of a FIFO buffer is set to the same initial value as the depth of NI and changes with the change of this value. In each scenario, 4 tests with different depth of NI are conducted.

During the tests it is assumed that the transfers of packets between nodes with the same distance would maintain the uniform value of delay. Such a delay of the packet transfer is calculated from Equation (5) which was derived from general latency equation in section

2.5.2, and it is used for both scenarios.

$$T_{total} = hops * T_{rh} + \frac{L * T_{HDSHK}}{b} \tag{5}$$

In equation (5) individual terms represents:

- T_{total} - the overall delay of the packet transfer from the source node to the destination in the network [cycles],
- $hops$ - the number of the routers the packet has to cross along its path from the source to the destination [unitless] (including both the source and the destination router),
- T_{rh} - it is the time required for the header flit to cross the router [cycles] (this time is set to 4),
- L - the length of the packet [bits],
- T_{HDSHK} - is the handshaking time set to 3 [cycles],
- b - the bandwidth of the channel [bit/s] (bandwidth is set to 32).

Thus for example, the delay of the transfer of 3 flits long packet from node 0 to node 3 can be calculated as $T_{total} = 3 * 4 + \frac{3*3}{32} = 12 cycles$.

In the first scenario, the small packets with the fixed length of 3 flits are generated. PIR range is set in the borders of 0.01 - 0.11 with uniform step 0.01. PIR also designates frame time during which every packet can be generated and injected into the network. Hence, frame time is calculated by means of Equation (6). The depth of NI is set to 4 flits and changes up to 32 flits with the uniform step of 4 flits for each test. The size of FIFO buffer is set to store 4 flits and maintains this value in each of four tests. The overall simulation time is set to 10000 cycles.

$$frame\ time = \frac{1}{PIR} \tag{6}$$

As depicted in Figure 23 the average delay for small fixed packets of the length 3 flits is low within the PIR range 0.01 - 0.06. Moreover, it can also be seen in Figure 23 that the average delay for all depths of NI have linear and uniform progress in this range of PIR values. The traffic by this range of PIR constitutes normal traffic under the saturation point of the network [9]. It means, the packets at this traffic are transferred with calculated latency from Equation (5) and do not overload the network.

As it can be seen in Figure 24, the throughput of the network within the range of PIR 0.01-0.06 has also linear progress. This means, the number of the transferred packets should equal to the number of generated and injected packets into the network. Thus, from PIR point 0.05 can be readily observed that the number of transferred packets in this point is ≈ 2000 packets. This corresponds to the number of the packets generated and injected into the network $4 * 500 = 2000$ packets. The evidence of generated packets is inferred from the fact. RUTPG generates 4 sets of packets at a time ,and the number of generated and injected packets during simulation time for PIR 0.05 is 500 packets per set.

As shown in Figure 23 since the point PIR 0.06 the network goes into the saturation. This can be observed by the rapid rise of the average delay primarily for NIs with bigger depth $NI\ 16$ and $NI\ 32$. Such a behavior is caused by the fact that the packets are generated and injected at a rate which exceeds the acceptable load of the traffic. The network becomes congested and the delay of the packet transfers rises up. Moreover, it is noticeable that in the tests with deeper NIs the rise of the average delay is even faster. This could be caused by waiting in injection queues of NIs [10] which is bigger for NIs with bigger depth. This in turn adds some additional delay to the packet transfer.

Reverse progress to the average delay can be seen in Figure 24 where the throughput of the network starts decreasing after the saturation point PIR 0.06. This means, the network is not capable to transfer the applied load which is increasing with PIR. Hence, from the observation in Figure 24 can be seen that the number of transferred packets in point 0.08 is ≈ 2700 packets. However, the actual number of generated and injected packets into the network is $4 * 769 = 3076$ packets during simulation time. Thus, the difference between generated and transferred packets constitutes over 300 packets. This points out the decreasing throughput because of the network congestion.

Finally within the highest range of PIR 0.09 - 0.11 depicted in Figure 23 the growth of the average delay mitigates and maintains almost at constant value. This is probably caused by the design of the hardware components which constraints the network to store and process only certain number of packets in FIFO and NI buffers. Thus, neither increasing of PIR causes exponential rise of the average delay. In this point the network is significantly congested and the packets cannot be any more injected into the network in such a rate.

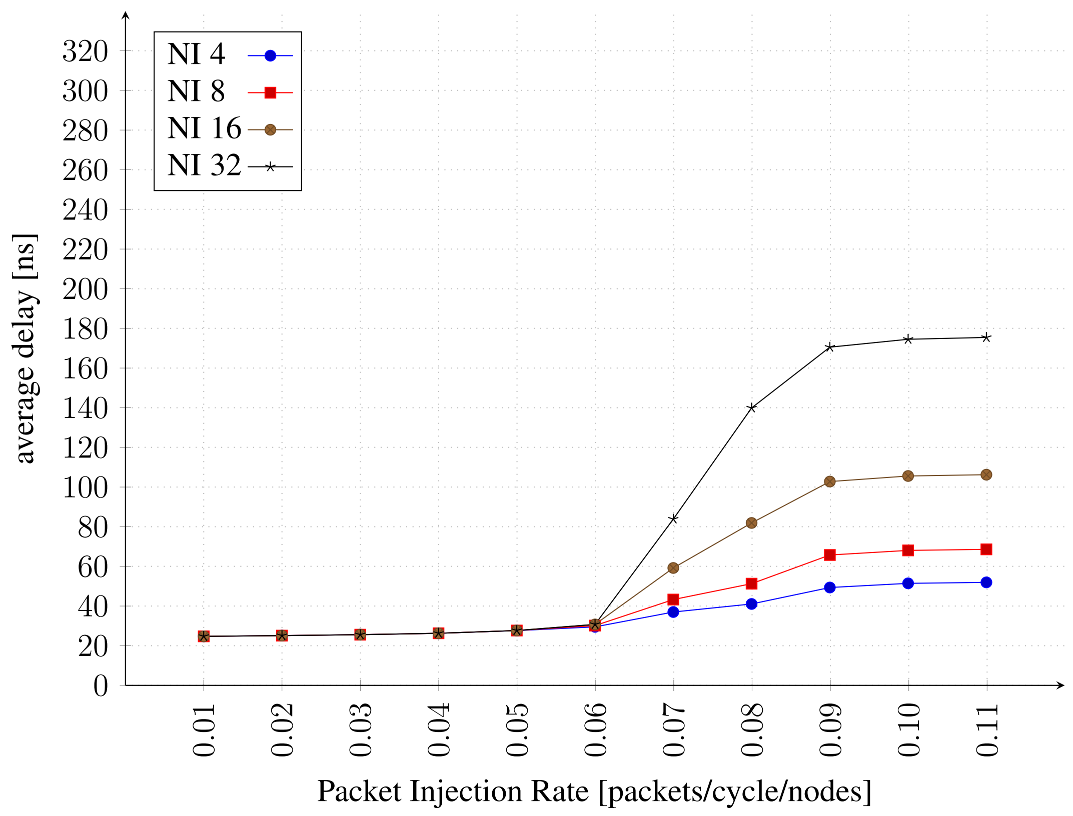


Figure 23. Average delay with FIFO depth 4 flits

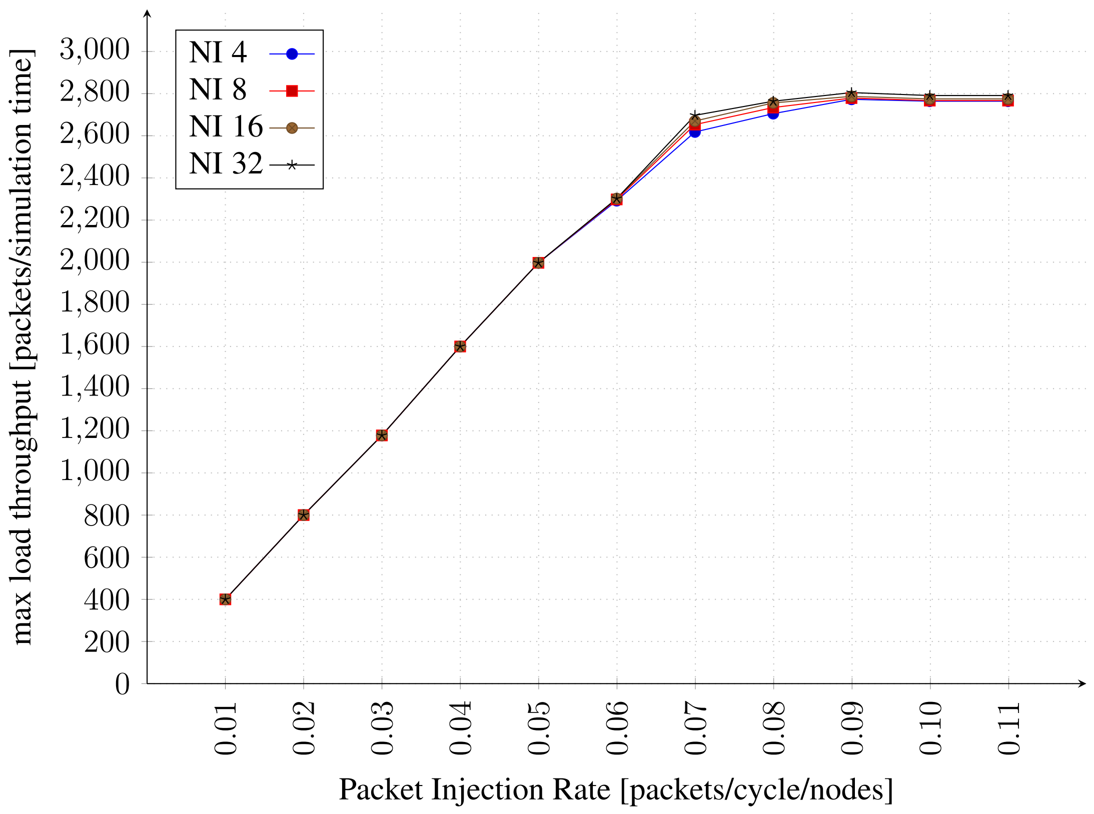


Figure 24. Throughput with FIFO depth 4 flits

In the second scenario, the initial conditions for the traffic are the same as in the previous scenario besides the size of router's FIFO buffer. In this scenario, the size of FIFO buffer is initially set to the depth of NI, and in turn it changes uniformly with NI's depth.

As it can be seen in Figure 25, the overall progress of the average delay is akin to the progress from the previous scenario except for some differences. The average delay sustains low values within the range of PIR 0.01 - 0.06. This is caused by the fact that the network works under the normal traffic without congestion. This fact is also confirmed with the linear rise of the throughput shown in Figure 26. As it can also be seen in Figure 26, the number of the transferred packets ≈ 2000 packets in PIR point 0.05 corresponds to the number of packets generated during simulation time $4 * 500 = 2000$ packets.

As depicted in Figure 25 the network becomes saturated since the PIR point 0.06. However, for the FIFOs and NIs with bigger buffers $FIFO\ NI\ 8$, $FIFO\ NI\ 16$ and $FIFO\ NI\ 32$ the rise of the average delay is even more rapid. This can caused by waiting time in the injection queue as mentioned in the previous scenario. Such a waiting time is even higher because of variable size of FIFO buffers in this scenario. Thus, waiting time in FIFO queues is added to the waiting time of NI's queues. In Figure 26 can be seen, that since the saturation point PIR 0.06 the throughput of the network also starts to decrease. However, as it can be observed the bigger size of FIFO buffers could have caused lower drop in the throughput of the network. Thus, the bigger FIFO buffers sustain the throughput of the network at higher value than the FIFO buffers with a smaller uniform value.

Finally, the average delays for 3 smaller FIFO buffers $FIFO\ NI\ 4$, $FIFO\ NI\ 8$ and $FIFO\ NI\ 16$ depicted in Figure 25 have similar progress within the PIR range 0.09 - 0.11 as the tests from the previous scenario. However, the FIFO buffer $FIFO\ NI\ 32$ in the PIR range 0.09 - 0.10 still turns out the rapid rise of the average delay. This can be caused by combination of FIFOs and NIs with big size of buffers. Such a combination could add waiting time from injection queues to the average delay at a time when the network is not completely congested. As it can be seen in Figure 26 the throughput of the network in the PIR range 0.09 -0.11 decreased and keeps at the constant level until the point 0.11 of PIR. However, from progresses of the tests with bigger FIFO and NI buffers $FIFO\ NI\ 8$, $FIFO\ NI\ 16$ and $FIFO\ NI\ 32$ can be observed that these designs offer higher throughput than designs with small FIFO buffers from the previous scenario.

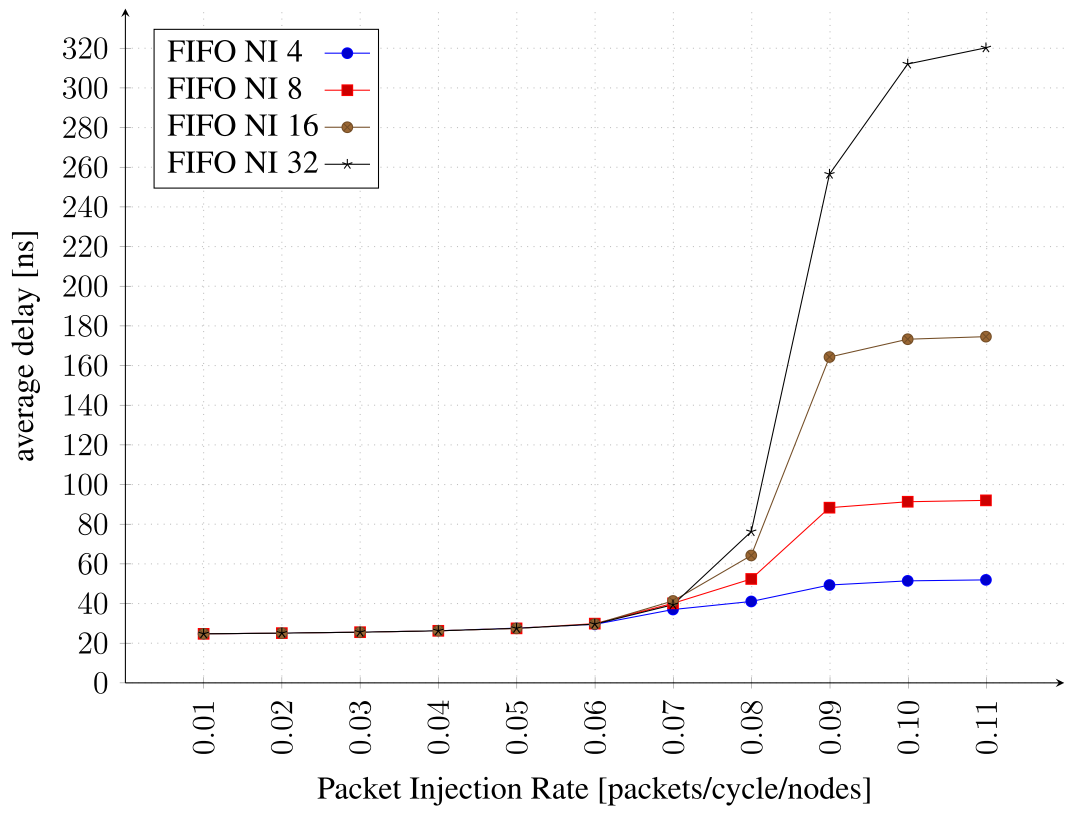


Figure 25. Average delay with uniform FIFO/NI depth

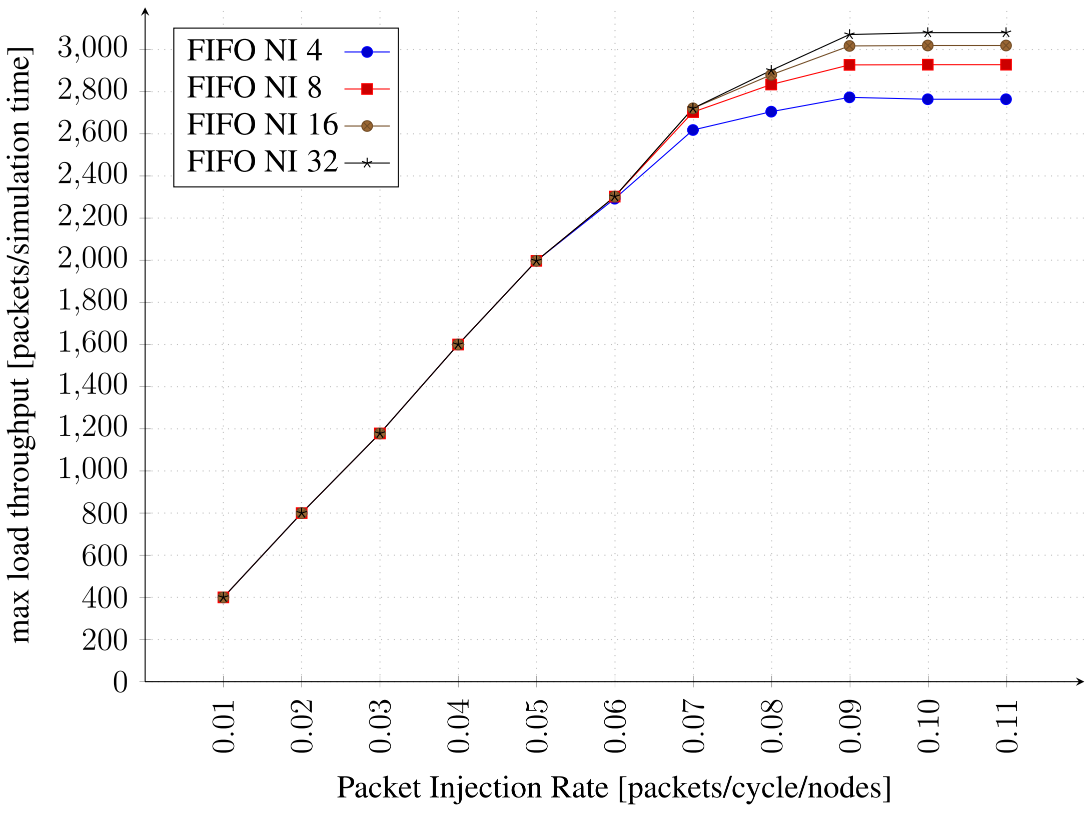


Figure 26. Throughput with uniform FIFO/NI depth

4. Summary

The aim of the thesis has been to measure and analyze performance aspects of Bonfire NoC designed in TUT university. Thus, during study case of Bonfire NoC the observation, the measurement and the analysis of the average delay and the throughput of Bonfire NoC traffic has been performed. The results of the tests not only give us the notion about the behavior of the network traffic before and after saturation point. Moreover, they demonstrate the influence of different concepts of the design on these factors. Therefore, the results can be helpful in the further research to assess optimal applied load for NoC's traffic.

References

[1] Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, and Davide Patti. Neighbors-on-path: A new selection strategy for on-chip networks. In *Proceedings of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia*, pages 79–84. IEEE Computer Society, 2006.

[2] Paul Baran. On distributed communications networks. *IEEE Transactions on Communications Systems*, 12, 1964.

[3] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38, 2006.

[4] Jeremy Chan and Sri Parameswaran. Nocgen: A template based reuse methodology for networks on chip architecture. In *Proceedings of the 17th International Conference on VLSI Design*, 2004.

[5] A.A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proceedings of 19th Annual International Symposium On Computer Architecture*, 1992.

[6] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, 11, 2000.

[7] C.J.Glass and L.M.Ni. The turn model for adaptive routing. In *Proceedings of the 19th International Symposium on Computer Architecture*, 1992.

[8] William James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[9] José Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection Networks An Engineering Approach*. Elsevier Science, 2003.

[10] P Ezhumalai, M Sriram, and A Saranya. Dynamic packet injection mechanism to control congestion in conventional noc. *International Journal of ElectroComputational World & Knowledge Interface*, 1(1), 2011.

[11] Jingcao Hu and Radu Marculescu. Dyad – smart routing for networks-on-chip. *Design Automation Conference (DAC)*, 2004.

[12] Faraydon Karim, Anh Nguyen, and Sujit Dey. An interconnect architecture for networking systems on chips. *IEEE micro*, (5):36–45, 2002.

[13] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *IEEE Computer Society Annual Symposium on VLSI*, 2002.

[14] Santanu Kundu and Santanu Chattopadhyay. *Network-on-Chip The Next Generation of System-on-Chip Integration*. CRC Press, 2015.

[15] Shaoteng Liu, Axel Jantsch, and Zhonghai Lu. Analysis and evaluation of circuit switched noc and packet switched noc. In *2013 Euromicro Conference on Digital System Design*, 2013.

[16] Ville Rantala, Teijo Lehtonen, and Juha Plosila. Network on chip routing algorithms. Technical report, University of Turku, Department of Information Technology, 2006.

[17] Ilkka Saastamoinen, Mikko Alho, and Jari Nurmi. Buffer implementation for proteo network-on-chip. In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–113. IEEE, 2003.

[18] Konstantinos Tatas, Kostas Siozios, Dimitrios Soudris, and Axel Jantsch. *Designing 2D and 3D Network-on-Chip Architectures*. Springer, 2014.

[19] Rikard Thid, Mikael Millberg, and Axel Jantsch. Evaluating noc communication backbones with simulation. In *The IEEE NorChip Conference, Riga, Latvia, Nov 10-11, 2003*, pages 27–30. IEEE conference proceedings, 2003.

[20] Wen-Chung Tsai, Ying-Cherng Lan, Yu-Hen Hu, and Sao-Jie Chen. Networks on chips: Structure and design methodologies. *Journal of Electrical and Computer Engineering*, 2012:15, 2012.

[21] Daniel Wiklund and Dake Liu. Socbus: switched network on chip for hard real time embedded systems. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 8–pp. IEEE, 2003.

[22] Daniel Wiklund and Dake Liu. Socbus: Switched network on chip for hard real time embedded systems. In *Parallel and Distributed Processing Symposium*. IEEE, 2003.