LNoC, Lagom Network On Chip

A thesis about network on chip and the design of a specification of a network on chip optimized for FPGAs called LNoC.

Robert Åkerblom-Andersson
Abstract

This thesis presents a new network on chip (NoC) designed specifically to suit the growing FPGA market. Network on chip is the interconnect of tomorrow, as of writing this, traditional computer buses are still the most used type of interconnect technology. At the same time research on NoC has been extensive over the last couple of years and is today used in some of the newest and most capable CPUs from leading companies like Intel, Texas Instruments and IBM. As more cores need to communicate on the same die traditional shared buses simple don’t cut it anymore, a network for interconnect communication is the solution. FPGAs are today produced at 22 and 28 nm and are capable of holding very complex logic designs with many cores. Low end FPGAs are also falling in price and it opens up the usage of FPGAs in markets where they earlier could not be used because of their high price.

LNoC is a network on chip designed to suit the need of both high performance systems and low cost FPGA designs. The key to LNoCs success is it’s reconfigurability and ability to adapt to the demands of each unique FPGA design, while keeping a standard interface for IP blocks enabling effective hardware reusability.

The presentation of this thesis work contains two main parts, the content of the thesis and the LNoC specification. The thesis chapters focus on the background and development of LNoC. Some example use cases are also discussed. It explains some concepts and technologies that the reader might not be familiar with. The specification contains an in depth description of LNoC and how it should be implemented in hardware.
# Table of Contents

1. **Introduction** ........................................................................................................... 7

2. **Programmable logic** .............................................................................................. 9
   2.1. Differences between CPLDs and FPGAs .......................................................... 9
      2.1.1. CPLD architecture quick facts ................................................................. 10
      2.1.2. FPGA architecture quick facts ............................................................... 10
   2.2. More information on CPLDs and FPGAs ........................................................ 10

3. **Network on chip** .................................................................................................... 11
   3.1. What is a shared bus system? ............................................................................ 11
   3.2. What is a crossbar switch? ............................................................................... 11
   3.3. What is a network on chip? What makes it better? ........................................ 12
   3.4. Comparing NoCs ............................................................................................ 12

4. **Lagom Network on Chip, LNoC** ......................................................................... 13
   4.1. Why LNoC? .................................................................................................... 13
   4.2. Choosing a topology ...................................................................................... 13
      4.2.1. Mesh networks ...................................................................................... 14
      4.2.2. Ring networks ....................................................................................... 14
      4.2.3. Augmented Ring Networks .................................................................. 15
   4.3. A special case of Hyper-Ring ......................................................................... 16
   4.4. Switching ....................................................................................................... 16
   4.5. Routing ........................................................................................................... 17
      4.5.1. Static routing (also called oblivious or deterministic) ............................ 17
      4.5.2. Dynamic routing (also called adaptive) .................................................. 18
      4.5.3. Distributed and source routing ............................................................... 18
      4.5.4. LNoC routing ......................................................................................... 18
   4.6. LNoC Quality of Service .................................................................................. 18

5. **Network layer model** ........................................................................................... 19
   5.1. The LNoC layer model ..................................................................................... 20
      5.1.1. Fabric layer (OSI layer 1) ....................................................................... 21
      5.1.2. Network layer (OSI layer 2, 3) ................................................................. 21
      5.1.3. Transport layer (OSI layer 4) .................................................................. 21
      5.1.4. Application layer (OSI layer 5, 6, 7) ......................................................... 21

6. **Use cases for LNoC** .............................................................................................. 22
   6.1. High throughput custom processing ............................................................... 23
   6.2. USB PWM controller ...................................................................................... 24
   6.3. Code lock ........................................................................................................ 24

7. **Summary and Conclusions** .................................................................................. 25
   7.1. Topology ........................................................................................................ 25
   7.2. Switching technique ...................................................................................... 25
   7.3. Routing algorithm ......................................................................................... 26
   7.4. Network layer model ..................................................................................... 26

8. **References** ........................................................................................................... 27

9. **Appendices** .......................................................................................................... 29
   A. LNoC Specification ............................................................................................ 29
1. Introduction

During this thesis project a network on chip specification specifically aimed at FPGA (Field-programmable gate array) fabric has been developed, it’s name is LNoC, “Lagom Network On Chip”. There are not a lot of existing options if you want to use a network on chip for FPGAs, therefor developing one was an interesting project. Because FPGAs and their fabric is expensive and complex, an optimized network could save money and open up for new types of applications.

The ASIC (Application-specific Integrated Circuit) market is a lot bigger than the FPGA market (including the fact that an FPGA is an ASIC from the FPGA manufacturers point of view). Therefor most research and information about networks on chips is based around the requirements and capabilities available to an ASIC designer. Compared to a network on chip designed with ASIC technology in mind, a network for FPGAs have to make use of a more limited set of features. An ASIC design could be thought of as a blank canvas where the designer have the freedom to add the exact structures that are needed, in an FPGA you instead are limited by a set of predefined structures that can’t be changed. On-chip RAM used for buffers is an example of such a resource, in an ASIC design of a NoC buffer sizes can be determined beforehand and the exact sized cheap on-chip RAM can then be added where it’s needed. In an FPGA you have to use predetermined on-chip RAM blocks, a limited and popular resource, or you have to implement buffers using logic. Because of this and the fact that FPGAs have very limited amount of on-chip RAM compared to what an ASIC can have, buffers sizes for LNoC needs to be kept down. At the same time FPGAs can be reconfigured between projects and designs at a low cost, so you can make use of those limited structures differently depending on the needs of the project. An ASIC design can’t be reconfigured after it’s be manufactured, therefor ASIC designs generally have to be more generic to fit many different project needs and have high NRE (Non-recurring engineering) costs. The NRE costs for respinning or customizing an FPGA design for a specific project is almost nonexistent in comparison. Because of these fundamental differences between the two technologies I believe FPGA developers might benefit from not using the same type of network that would have been used for an ASIC.

To optimize the design for FPGA fabric focus has been on making it a flexible network that can be implemented in different ways in order to make best use of the available fabric. In ASIC design, after tapeout you can’t change the design, FPGAs can be reprogrammed multiple times the same day. The reprogrammability of FPGAs makes it possible to iterate and test different configurations of a network and to optimize the network for different applications with different needs. One application might need high speed and low latency, a different one just the connectivity of a large number of devices where high speeds is not important. This is a use case where LNoC could be used in both projects but be optimized differently to suit the needs of the application. The ability to have a flexible network where you can trade between performance and resource usage could make it possible to choose a cheaper FPGA for a specific design project, earning it’s value compared to a more generic network design.

The key goal in the design of LNoC was to make the right compromises between performance and reconfigurability. The strongest selling point of an FPGA is it’s reconfigurability, the weakest, their price. In the design of a network on chip specification there are many choices to be made, in this rapport four of them have been highlighted extra because of their importance. The first question is about topology, all networks have a topology and different topologies have different benefits. To fulfill LNoCs key goals the topology needed to supports reconfigurability and make it possible to compromise between performance and logic usage and vise versa when needed. A wanted feature of the topology was also flexibility in the layout of the network, it should be possible to represent the same network in different configurations such that different network layouts could be chosen to minimize critical paths and to maximize logic utilization of
the FPGA. Switching and routing algorithms needed to be chosen to suit the needs of LNoC but also to work good with LNoCs topology, especially the routing algorithm chosen depends a lot of the topology. The forth question was how could the network be explain with the help of a network layer model, more specifically how does it map against the OSI model. A network layer model makes it easier understand the network and to develop hardware and software for it.

A summery of the four main questions for this thesis:

1. LNoC needs a topology that supports reconfigurability and that makes it possible to compromise between performance and logic usage and vise versa, what topology should be chosen for LNoC?
2. What switching algorithm should be used to archive the goals of LNoC and to fit the LNoC topology?
3. What routing algorithm should be used to archive the goals of LNoC and to fit the LNoC topology?
4. How could LNoC be explain with the help of a network layer model?

In this thesis the reader can get some insight in the design choices that have been made during the development of the LNoC specification. Chapters two and three introduces the reader to programmable logic and the concept of network on chip. Readers already familiar with those concepts can start reading from chapter four, where LNoC is introduced. Chapter four handles three of the four main questions, topology, routing and switching, each in it’s own sub chapter. A proposed network layer model is presented in chapter five followed by a chapter listing some possible use case of LNoC. Chapter seven contains a summery and conclusions of the thesis.
2. Programmable logic

Programmable logic is a family of integrated circuits (IC) that has been developed to be reconfigurable by the user. Programmable logic in its fundamental form is a set of logic gates inside a chip that you can program to be connected in different ways. It’s important to understand the difference between programmable logic and microprocessors. A microprocessor is a built up with logic and then executes program code, a programmable logic device is just gates setup to be connected in a certain pattern, and if enough gates is available that pattern could for example make up the base of a microprocessor or any other logic construction.

In the early days the programmable logic the chips were very small in capacity and their main purpose was to replace standard logic chips containing a few gates each or to be used as what is called “glue logic”, small pieces of logic that “glue” together two systems that might have some smaller mismatches. With configurable logic it became easier to make smaller adjustments to circuit boards in late stages or when some chip had to be replaced in a newer version without having to respin the whole design. Over the years programmable logic has evolved a lot, the early simple architectures called PAL and GAL were later superseded by the two types of logic architectures used for most designs today called CPLDs and FPGAs. CPLD stands for complex programmable logic device and FPGA for field-programmable gate array. The CPLDs came first are were as the name suggest a more complex and enhanced version of the original programmable devices. The FPGA on the other hand were more unique and different since it was based on a lookup table architecture, an architecture proved over time to be the most scalable and successful way to implement programmable logic.

Since a network on chip only is relevant on bigger chips the two types of programmable logic technologies that LNoC has been developed for are CPLDs and primarily FPGAs. Both CPLD and FPGA devices are programmable and can contains a big amount of logic resources, although the biggest CPLD is small compared to the average sized FPGA there are at the same time FPGAs that are smaller than the biggest CPLD. Because of FPGAs more scalable architecture most research and development goes into FPGAs but CPLDs also have their place on the market.

2.1. Differences between CPLDs and FPGAs

In general CPLDs can only be used for smaller designs compared to FPGAs since the lookup table architecture scales much better and FPGA models exists with very high amount of logic resources. CPLDs on the other hand, despite having the word “complex” in its name, are generally simpler to use and cheaper than FPGAs. For smaller to medium sized projects it all comes down to price and functionality and using the best chip for the job. For medium to big sized projects where a lot of logic is needed FPGAs becomes the only option possible.

At an architectural level, what makes CPLDs and FPGAs different is how they are built inside and how the gates are connected to construct the programmers design. It is worth noting that both CPLDs and FPGAs are programmed in the same hardware description languages, VHDL or Verilog, the same code/design can be synthesized to either a CPLD or an FPGA without any changes. The logic synthesis step is in a way equivalent to the compilation step in the software world, however the output is more like a description of how the logic should “look like” whereas the output of a compilers is more of like a set of instruction of “how to” do something. The CPLD architecture is more coarse-grained and have a more connected interconnect. FPGAs on the other hand are more distributed and consists of a bigger amount of smaller logic blocks that are connected together to form the FPGA.
2.1.1. CPLD architecture quick facts
1. Simple and predictable timing model as a result of the coarse granularity (signals can go one way more or less, it’s like a highway where all signals are routed).
2. CPLD in some cases run at higher frequencies than equivalent FPGAs.
3. Pinout can be changed without change timing (Cypress claim).
5. Cheap.
6. Some CPLDs have simpler hard IP block included.
7. Less flexibility and more predictable timing delays.
8. Higher logic-to-interconnect ratio.
9. Often used for on PCB “glue logic”.

2.1.2. FPGA architecture quick facts
1. Higher routing flexibility but less predictable timing delays. Signals have to travel between multiple small “islands” of logic inside the FPGA and the same signals can be routed different ways around the logic blocks to get to it’s destination.
2. Fine grained, many smaller logic units.
3. Scales very good in logic size.
4. Mostly volatile but more and more non volatile options exists.
5. Modern FPGAs gets more and more hard IP block included.
6. Many FPGA series have one "logic" version and one "transceiver" version that included different types of hard IP transceivers.
7. Some FPGA have on chip hard CPU cores.

2.2. More information on CPLDs and FPGAs
This introduction to CPLDs and FPGAs is not a complete reference although it might function as a good refresher for readers that have some previous programmable logic experience. For more information on programmable logic architectures the interested reader is suggested to read Stephen Brown and Jonathan Rose paper “Architecture of FPGAs and CPLDs: A Tutorial” [1].
3. Network on chip

On a FPGA chip or any IC with different processing nodes that should communicate there has to be sort of communication interface between each node. Traditionally there have been two different ways to implement this communication, in processor based systems the shared bus has been the most common and in FPGA designs crossbar switches. Both the shared bus and the crossbar switch have their benefits but neither of them scales that very well and that’s where the network on chip, the third alternative, comes in. A network on chip is not as effective at low node counts, a reason why it has not been used that much in the past, but as the node count rises it beats the competing technologies. In the following sub chapters 3.1 through 3.3 each technology is presented a little more in detail, in sub chapters 3.4 some characteristics of different network on chip architectures are discussed.

3.1. What is a shared bus system?
The shared bus system is common in microcontrollers and for almost all CPUs in general. A shared bus system is based on the concept that all nodes are connected to an address bus and a data bus and some control signals like write/read. All nodes in the shared bus system is assigned its own memory range, eg 0x20-0x30 (the complete set of all nodes memory ranges is called a memory map). To write or read to a specific node a processor can use its memory map to determine where a certain peripherals or node exist. The nodes on the bus are setup to only "react" when the address on the address bus is inside its specified memory range. This technology works very well for the typical microcontroller and or any architecture where you have only one or a few processing nodes that should act as masters on the bus. The biggest downsides of a shared bus system is that only one master can use the bus at any time, this make the bus very ineffective as the number of master nodes increases and the nodes trying to use the bus have to wait on each other since they all share the bus.

3.2. What is a crossbar switch?
A crossbar switched interconnect is basically a big matrix of X inputs and Y outputs. For each input there is a “switch” where you can choose to connect it to any of the Y outputs. Using this technique you can create simple and effective connections between different nodes without that much complexity. It comes at a price though, a crossbar switched interconnect are both more complex and simpler than a shared bus system at the same time. The biggest improvement is that switched system support multiple simultaneous transactions. The crossbar switch setup makes it possible to connect any node directly to any other node in the network. The downside of the crossbar switch is that it does not scale very well, because of it’s generous amount of connections and support for any to any connections it gets exponentially more expensive and power hungry as more nodes are added [2].
3.3. What is a network on chip? What makes it better?
The network on chip is a technology inspired by the local computer networks and the Internet that we use on an everyday basis. The most prominent feature of computer networks as we know them are that they scale, the one criteria where both the shared bus and the crossbar switch fails for performance or economical reasons. The network on chip, commonly written as NoC, is just what it sound like, a network that exist inside a silicon chip. Compared to the shared bus or crossbar technologies, network on chip would be more rightfully classified as a family of different types of networks since them can be built in so many different ways. Network on chips biggest strength over the shared bus is performance scalability, a shared bus does simply not scale since they only permits one master at any time. Crossbars switches can do performance scaling quite good, but economically, size and power wise they are inherently expensive since their costs scale exponentially with added nodes. The NoC approach makes it possible to make up for the downsides of the shared bus or the crossbar switch based systems. Even though different NoC may have their own downsides as well at the end of the day the best solution does always have to have the right performance/price ratio.

3.4. Comparing NoCs
Network on chips can be categorized by how they implement different parts of the network. The most visual difference between different NoC designs is the network topology, how the nodes are actually connected to each other geographically. The mesh network is the most common topology for NoC designs. Each topology also has a node degree, that is how many connections each node have. The network diameter is a different parameter measuring the highest hop count for a packet in the network. Bisection bandwidth is used to determine how much traffic can travel through the center of the network. It can be calculated visually by cutting a topology in the middle and count the number of broken links. When two nodes should communicate the switching technique and routing algorithm determines how the message will be delivered. If the network is a packet based, network packets are used to send data between two nodes, similar to a computer network. Circuit based networks on the other hand is more similar to the original phone systems where each call or transfer between two nodes are first setup and then all traffic is send over a “dedicated line” that then is busy until the transfer is finished. The routing algorithms comes in many flavors and depends on the topology whereas switching technique are less bound to topologies. Depending on the topology a routing algorithm is chosen to fit the needs of the network.

Some NoC features:
1. Topology
2. Node degree
3. Diameter
4. Bisection bandwidth
5. Switching technique
6. Routing algorithm
4. Lagom Network on Chip, LNoC

LNoC is a network on chip designed to be able to suit a wide range of applications and FPGAs. LNoC has been developed with flexibility and scalability in mind from day one. For details on the specifications on the network see appendix A, LNoC Specification.

There were not many specific goals setup before the development started. During the development of a specification it's hard to tell how the complete system should look like before the work has begun. One goal was set up though, that was to try and have a good balance between readability and effectivity of the specification. Almost all successful bus standards are often quite simple. The secret to a good specification is to make a complicated enough and effective system that at the same time is easy to understand and delivers good performance.

4.1. Why LNoC?
LNoC stands for Lagom Network on Chip, and the name describes many of the design goals of the network. In order to be functional the right design decisions needs to be taken in order to create a network specification that is “lagom” (a Swedish word with no direct English translation, mean something like not too much and not too little). FPGAs are getting more capable and cheaper for each year, therefore does LNoC lay very good in time to function as a backbone for FPGA system design development with IP blocks.

4.2. Choosing a topology
The first thing that was decided in the development of LNoC was the topology. The network topology for any network, and especially for NoCs is very important since it affects the amount of logic resources needed and therefore the total cost of the network. Many different network topologies exists today, and many have been thoroughly researched over the years in PC networks. Two common topologies that was looked at from the start was mesh and ring networks.
4.2.1. Mesh networks
Mesh network are the most common network topology for network on chip designs [3]. A mesh network is a network where all nodes are connected into a big mesh of nodes (see Illustration 1). One of the reasons why mesh networks are so common is that they have high performance and are simple to create effective routing algorithms for. XY routing is a simple and popular routing algorithm. A notable NoC design based around a mesh topology is the Swedish Nostrum [4].

Illustration 1, a 16 node mesh based NoC.

4.2.2. Ring networks
The ring topology has been used in many high performance systems. One example is the Cell microprocessor used in the PlayStation 3 (see Illustration 2), developed as a joint venture by Sony, Sony Computer Entertainment, Toshiba, and IBM, an alliance known as "STI". The cell processor uses multiple rings that the processing nodes are connected to, but the topology is still a circular ring [10].

Illustration 2, the ring network used in the Cell microprocessor [11].

Because of rings simplicity and low node degree, rings early became an interesting alternative. The problem with the ring topology is that it does not scale that very well. As the ring gets bigger
the distance between all nodes increases and multiple nodes share the same shortest paths. A ring network was therefore chosen as the base structure because of its simplicity and high performance (see Illustration 3 for a LNoC ring). Smaller ring networks are a well tested and proven topology, however it does not scale that well over 8-10 nodes. The main reason that ring network are effective and low cost compared to many other topologies is their simplicity. Most other topologies has a higher node degree, ie, they require more than two interfaces, a mesh element/router has a node degree of 4 for example. Since a ring router only has two interfaces, it can be made very effective by using a dual ring.

Illustration 3, LNoC topology with 6 nodes.

4.2.3. Augmented Ring Networks
In order to make use of the benefits of the ring topology, and yet make the network scale, it is possible to create augmented ring networks. In the literature four base types of augmented ring networks was found [5]. These different types of augmented rings are;

1. Chordal rings
2. Express rings
3. Multi-rings
4. Hierarchical Ring Networks

The hierarchical ring topology was considered for a while, however a specialized type of augmented ring topology was then found. The Hyper-Ring topology proposed by Sibai [6] is a scalable and ring-based topology and it did fit the requirements of being scalable while still keeping down the average node degree and therefore also being low on resources. Since the Hyper-Ring meet the requirement of LNoC it was chosen as the base topology.
4.3. A special case of Hyper-Ring

The Hyper-Ring topology that Sibai presented in his paper did not specify any limits on the size of each ring and dimension, the paper rather presents the general model of the Hyper-Ring network. LNoC is a more strict version and has a maximum ring size of 6 nodes, this number came primarily from the fact that three bits can represent 8 numbers. Using four bits for addressing would yield 16 possible addresses, 16 or thereabout was considered a too high number of nodes to have on one ring, three bits and 6 nodes inside each ring was simple and a very good fit. Therefore LNoC has a maximum ring size of 6 nodes. The remaining two bit patterns not used for node addresses are used to create broadcast, multicast and anycast addressing (for details please see the LNoC specification). As seen in Illustration 4 LNoC uses rings of 6 nodes connected together at different levels to create the network topology.

![Illustration 4, LNoC topology with 1296 nodes.](image)

4.4. Switching

Switching is one of the things a network designer has to take into account. The three main alternatives in switching on NoC are store and forward, virtual cut-through and wormhole switching [7].

In store and forward switching, a message is buffered in its whole entirety at each node before its sent forward in the network. This types of switching requires big buffers in order to hold complete messages at each node and has an inherent high latency since there is a wait stage at each node when it waits for the complete message to arrive. One benefit of this technique is that it only "occupies" one part of the network at a time. For on-chip networks low latency and small buffers are preferred, therefore is store and forward not a good option.

Virtual cut-through switching is different from store and forward in the sense that it does not buffer the whole message at each node, that means smaller buffer sizes. Virtual cut-through on the other hand buffers one complete packet at a time (a message is made out of a set of packets), when one whole packet has arrived it is sent forward while the next packet is received. This
approach makes use of smaller buffers and has also lower latency, therefore virtual cut-through switching could be an option for NoC networks.

Wormhole switching on the other hand, has even lower latency than virtual cut-through switching. The concept of wormhole switching is the same as for virtual cut-through, the big difference is that in wormhole switching it is not packets that are buffered, rather flits, the smallest unit of flow in a network (packets are made from 1-N flits). Buffering at the smallest unit of transfer on the network makes wormhole switching the lowest latency version of these three proposed switching techniques. Wormhole switching was therefore chosen for LNoC, it is also a common choice for other NoC because of the reasons just stated.

### 4.5. Routing
Routing inside any network is a fundamental function that has to work well. This applies especially well for NoCs, you could argue that latency is more important on a NoC network compared to a general purpose network. What kind of routing algorithm that is used for a specific network depends a lot on the network topology but can also vary with the same network topology.

Routing algorithms in general can be classified in some different ways. Three of them are "static vs dynamic routing", "distributed vs source routing" and "minimal vs non-minimal routing". Except making sure that a packet arrives to its destination it is always the routing algorithms responsibility to make sure that no packet get stuck in deadlocks or livelocks. A deadlock is when two nodes wait on each other to finish, and neither will ever finish since they wait on each other. In the case of a so called livelock, then a packet can move in the network but it never reaches its destination.

#### 4.5.1. Static routing (also called oblivious or deterministic)
Static/oblivious/deterministic routing does as the name suggest not adjust to current network activities. Here are some properties of static routing:

1. Fixed paths.
2. Does not take current state of network into account.
3. Unaware of router loads.
4. Little logic required.
5. Multiple paths for different package in a predeterministic way.

Some examples of static routing algorithms are:

1. Dimension order routing (DOR).
2. XY routing.
3. Turn model (west-first, north-last, negative-first).
4. Source routing.
4.5.2. Dynamic routing (also called adaptive)
Dynamic/adaptive routing is the opposite of static routing in terms of adaptivity. An adaptive network is meant to route packets differently depending on the current network load. Some packet may be redirected a longer way but still get to their destination faster since the shorter way was congested.

1. Routing decisions are made according to current state of the network.
2. The route between two nodes can change over time.
3. Additional resources.
4. Better to distribute traffic in a network.
5. Can utilize alternate paths if congesting happens.

Some examples of dynamic algorithms are:
1. Minimal adaptive.
2. Fully adaptive.
3. Odd-even.
4. Hot potato routing (also called deflective).

4.5.3. Distributed and source routing
In distributed routing each packet carries the destination address, routing decisions are made in each router with the help of a table lookup or hardware execution. Source routing on the other hand, carries a list of the whole set of routers that is should pass in the network. This may add to the total amount of data sent on the network, but at the same time the amount of work that each router has to do gets less. Source routing might get really big address headers depending on how big the network is.

4.5.4. LNoC routing
For the LNoC network, a specific algorithm have been developed to support the topology and to consume as little logic and power as possible. For a detailed description of the routing algorithm see appendix A, LNoC Specification.

LNoC routing algorithm characteristics:
1. Static routing
2. Distributed routing
3. Wormhole switching
4. Support for virtual channels

4.6. LNoC Quality of Service
What types of QoS that a system should support is always hard to determine. So was also the case with LNoC. On one hand, sometime you don’t want any QoS services, and other times they could be priceless. Therefore quite high level of QoS is provided, but none is required per default if the user don’t want to. Three main types of QoS are supported, priority routing, time to live and cyclic redundancy checks.

Priority based routing makes it possible to give packets different priority and thereby ensure that some packets always should be routed first. The TTL functionality help to make sure that no packets get stuck in the network, it is not a required field but might be used in many cases. Switch to switch CRC support is the most expensive types of QoS supported by LNoC. CRCs between 1 and 8 can be used depending on the requirements. It provides extra security that no bits have been flipped as a flit is transferred between two network element.
5. Network layer model

It is common practice in the definition of network protocols and stacks to define a network layer model. The OSI model is an ISO standard model for network layer models. It defines 7 layers for different network functions and specifies what each layers task is. LNoC does not implement the OSI layers as intended in the OSI standard, it rather implements it’s own layer model. Few networks do actually implement the OSI model exactly as it is specified, however it is still a good reference point and networks usually include all or most functionality, just not in the exact seven layers specified.

The reader is expected to know the basics of the OSI model, so this list should be considered as just a review or reference list of the different layers function in the OSI model [8].

Layer 1, the physical layer
Voltage levels, cable types, interface pins.

Layer 2, the data link layer
Packet handling, transmits/receives packages in the network between two nodes.

Layer 3, the network layer
Routes data through various physical networks while the data is traveling to its known destination node.

Layer 4, the transport layer
The transport layer takes care of reliability and error checking. In this layer it is possible to make the decision to optimize for speed or reliability, ex, TCP vs UDP, error checking or dropping package.

Layer 5, the session layer
Sometimes called "port layer", manages setup up and take down of the association between two communicating nodes, basically it takes care of the session handling of a connection between two nodes.

Layer 6, the presentation layer
Makes sure that the information received is in the right format.

Layer 7, the application layer
Defines standards for interaction at the user or application program level.
5.1. The LNoC layer model
The LNoC layer model consists of four layers (see Illustration 5 and 6), where three of them are implemented directly in hardware. The fourth layer is a software layer that might implement different protocols and/or APIs depending on the application. For implementation details on the different layers see appendix A, LNoC Specification, this section tries to be a more higher level introduction to LNoCs layer model.

Illustration 5, LNoCs layer model.

Illustration 6, an illustration of how different layers in LNoC interface with each other.
5.1.1. Fabric layer (OSI layer 1)
The bottom layer is the fabric layer, it has the equivalent functions of the first layer in the OSI model. The fabric layer basically takes care of all things at the lowest level, how many bits should be used for a specific field etc.

5.1.2. Network layer (OSI layer 2, 3)
The network layer in LNoC takes care of all the parts that has to do with the actual networking. In the OSI model its functions is equivalent to those of the layer 2 and 3. It is in the network layer that the behavior of each router is defined, how switches should behave and how data should be buffered.

5.1.3. Transport layer (OSI layer 4)
The transport layer in LNoC is quite similar to the 4th OSI layer, also called the transport layer. It is responsible for node to node connections and offer resources similar but not equal to the UDP and TCP protocols that the reader might be familiar with. The transport layer is the first layer that an end designer should have access to. Depending on the application the transport layer could be the highest layer in use. Applications including some sort of processor will most likely also use the application layer.

5.1.4. Application layer (OSI layer 5, 6, 7)
The application layer of LNoC is the only layer that is not meant to be completely implemented in hardware, rather in software. LNoC does not provide a specific application layer protocol but rather suggests some functionality that can be implemented. The MCAPI API [9] is one example of an API that could be implemented as an application layer protocol/API for LNoC. Functionalities needed by MCAPI is provided by the transport layer.
6. Use cases for LNoC

To support bigger FPGA designs a network on chip is a good solution that offers flexibility and good support for scaling designs. The low end FPGAs starts to become affordable for more and more projects, although the problem that many embedded software engineers don’t know how to design in a hardware description languages remains. With the help of LNoC easier to use development tools can be built and FPGAs could be accessible for a larger audience. In order to harvest the capabilities of programmable logic, a hardware description languages such as VHDL or Verilog have to be used today. For bigger FPGA designs it gets more and more common that developers abstract away much of the actual logic design by using reusable predefined logic blocks called IP blocks. Designing more parts of the design at a higher level of abstraction reduces development cycles and time to market. In order to use a block or system level design methodology a good underlying interconnect technology is crucial. LNoC is meant to provide just that, a reliable, high performance and scalable backbone on chip network to support effective block level designs. LNoC has been developed as an alternative for both high end applications and as a backbone for simpler designs.

During the development of LNoC three different end users have been considered:

1. The experienced hardware engineer designing high performance hardware designs.
2. The embedded engineer sometimes using programmable logic.
3. The embedded programmer or general purpose programmer building embedded systems.

Experienced hardware engineers need a NoC to support state of the art systems with multiple processing cores that need to communicate fast and on chip. It may be custom developed IP blocks performing a specific purpose, simpler processing nodes working together to create a pipeline or just a set of different systems that need to share the same silicon to save space on a PCB. An example of an application could be a system where 16 processing cores work together to create a pipeline where each node does some processing on a set of incoming data before it sends it of to the next processing core for a different task and together the cores collaborate to create a high performance system.

The average embedded engineer is normally familiar with FPGAs but might not work with them on a regular basis since they simply are too expensive or MCUs better suits most of the tasks they are asked to solve. When FPGAs can meet the price demand and would benefit the end product for engineers in this category he or she simply might not be up to the challenge of dusting of his/her FPGA skills. Solutions like ASSPs (application specific standard products) or other ASICs (application-specific integrated circuits) might fit the job sometimes, others no suited IC even exist. As an example it could be project that would benefit from a chip capable of individually control 12 high performance PWM output, and this functionality should be accessed from a USB interface.

Since hardware design is a complicated process, and the amount of engineers mastering hardware descriptions languages is limited compared to more mainstream embedded systems programming or general purpose programmers. There is also a market for specially designed chips that are easy to program or that simple don’t need any programming at all, just a graphical representation of the functionality. In this case LNoC comes in handy as a good back end for auto generated logic, based on a set of predefined functionalities that the user can combine in a graphical interface. A trivial example could be a simple code lock that could be designed in a drag and drop fashion.

The following sub chapters describes how LNoC could have been used in the discussed use cases.
6.1. High throughput custom processing
In this use case LNoC is used to implement a custom processing pipeline. Reasons to use an FPGA for an application like this is its raw speed and parallelism. The pipeline takes in a signal or a stream of data to the left on the picture below (Illustration 7), the data is then processes and outputted at the right side of the network in the picture.

Illustration 7, 16 processing units connected together using LNoC.

Inside the FPGA the data arrives to the first processing node (001.001) that does its part of the processing, next it send the data forward and receives new data. When the pipeline is full one completed commutation will comes out at every X clocks ticks (if it takes X clock ticks for the slowest node in the pipeline).

This types of pipelined processing is commonly found in different networking and telecom equipment where packet processing often can be accelerated by splitting it up into smaller parts and let a set of processors work together to get a higher throughput. Video and audio processing does also use this type of pipelines heavily where each node could represent a filter or some other processing.
6.2. USB PWM controller

The second use case is of a custom circuit design of a set of PWM controllers that can be individually controlled from a PC through a USB interface (see Illustration 8). PWM stands for pulse width modulation is used for many things, one use case is to fade the intensity of LED based lighting. A different application where PWM is used is in motor control where a PWM signal is sent to the electric motor to determine its speed (depending on the motor some additional circuitry might be needed in between).

Illustration 8, a USB interface connected to three four-output PWM controllers using LNoC.

6.3. Code lock

The last example is a code lock with an alarm function (see Illustration 9). A keyboard decoder has been implemented and connected to a codelock block, a sound output is also connected to give off the alarm (PWM can also be used to create sounds).

Illustration 9, a keyboard decoder connected to a single PWM controller and codelock logic using LNoC.
7. Summary and Conclusions

During this thesis work a specification of a network on chip called LNoC have been developed successfully. In the introduction chapter the four main questions of network topology, switching technique, routing algorithm and network layer model were introduced. In this chapter the original questions are reflected upon, discussing the alternatives and the final choice.

7.1. Topology
For LNoC, raw performance was not necessarily the number one priority. Finding a topology that offered flexibility and the ability to connect the same devices in different network layouts while still having the same functionality were. Resource usage was also a big priority since lower resource usage means lower cost and more resources for the actual IP blocks (developed in-house or by a third party) that makes up the main part of an FPGA design. The mesh topology was considered because of it’s simple design and common usage. The reason why it was not chosen for LNoC was mainly because of it’s high node degree. The high node degree made it too expensive on resources. Ring networks have the benefit of a lower node degree, making them an interesting topology from the start. However, they are not perfect, a single ring simply does not scale very well, and therefor was not chosen. Beyond the simple single ring different types of augmented ring networks have been created over the years. These were looked at and later a specialized type of ring topology called Hyper-Ring was found. LNoC ended up using a custom version of the Hyper-Ring topology. The Hyper-Ring topology was chosen as a base topology and LNoCs topology was later developed as a more strict version of the Hyper-Ring. Beyond the topology also a new addressing schema was developed that did not come from the original specification of Hyper-Ring topology. A static ring size of 6 nodes was chosen, it made for a good compromise for speed and connectivity while it at the same time allowed for an effective address encoding system used by LNoC. The node degree of LNoCs topology is in between that of a mesh or a ring network and the performance and flexibility of the Hyper-Ring topology made it the final choice.

7.2. Switching technique
Three different switching techniques were considered for LNoC, store and forward, virtual cut-thought and wormhole switching. Here the main focus was on resource usage, how can we do good enough switching and still keep down the resource usage. Store and forward was not chosen because of its dependence on big buffers, since complete message are buffered at each router. Buffers in FPGAs are either implemented with help of on-board RAM blocks (limited resource that other parts of the design wants to use) or by implementing RAM using logic (an expensive way to implement buffers), because of this small buffers was a top priority. Virtual cut-thought switching looked a lot better since it buffers packets instead of messages, making it possible to use smaller buffers, however it was yet not the best alternative. Wormhole switching goes one step further and came out as the winning switching technique. Wormhole switching is very similar to virtual cut-thought but instead of buffering packets it buffers flits instead, the smallest unit of flow in the network. Essentially a more fine grain version of virtual cut-thought switching that makes it possible to have smaller buffers. Wormhole switching provides low latency switching and can be implemented with relatively small buffers compared to the other techniques.
7.3. Routing algorithm
To communicate over a network messages have to have a way to reach their end destination. A routing algorithm makes this possible and it depends a lot on the topology that it should route the message though. Since LNoCs topology is a special version of a rare topology a new routing algorithm was developed for it. Again resource usage was the one of the top priorities. The sizes of the rings that makes up the topology, and the addressing used, were both chosen such that it would be possible to develop an effective routing algorithm. To keep down overhead and because of the nature of the topology the developed algorithm is a static and distributed algorithm. The algorithm makes use of the addresses of each node that is based on three bits in hardware to do effective routing, based on the destination address each router can determine if the package is destined to it’s node address or to which one of it’s neighbors it should send it forward to.

7.4. Network layer model
The network layer model was not really a matter of choosing between A or B, rather a mapping problem of how to map the commonly known OSI layers to LNoC. LNoCs network model consists of four layers; the fabric layer, the network layer, the transport layer and the application layer. Each one mapping to different parts of the seven part OSI layer model. In an end application there might be more layers of protocols on top of LNoC if the application requires it, whereas LNoCs specification stop at a relatively low level. Making up a network layer model does not really create a direct performance enhancement per say, since it’s more or less just an abstract way of looking at how the network works. However it’s still a critical part, it looks obvious when it’s done, but without it it’s not as easy to understand the network. Implementing the network specification and IP block to go with it also becomes a lot easier when a network layer model exists.

The goal of the project was meet, a first version of the LNoC specification is now done. The amount of work required to finish the job might have been underestimated a bit, at the same time it might have been done on purpose since it was a fun project and I was willing to put in some extra time. The next step would be to take the specification and implement it in VHDL or Verilog and synthesise it to a bitstream that can be programmed into an FPGA, I hope to be able to that one day. Implementing the specification should be a good and industry relevant task for at least a couple of years to come as NoC for FPGA still is a very young area and the industry currently lacks something like LNoC.
8. References


9. Appendices

A. LNoC Specification