Nostrum Network-on-Chip RTL Manual

Johnny Öberg (johnnyob@kth.se)

Technical Manual/User Guide

Electronic Systems Department
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

30 August 2016

Contributed By: Mohamed T. Mohammadat (mtme@kth.se)

Chapter 1

Introduction

Network-on-Chip or on-chip network is considered a scalable architecture for many-core applications providing high communication bandwidth, reduced latency and improved energy-efficiency compared to other alternatives. The major elements of such architecture are switches connected in a particular topology and interfaced to processing nodes through network interfaces (NI). A typical example of a NoC can be seen in Fig. 1.1. The figure depicts four resources or nodes marked as Rxx, connected to a structure of switches array each is denoted by Sxx via network interfaces (NI). There are several terminologies concerning the NoC and to reduce possible ambiguity in the remainder of this work, some terminologies have been defined briefly as follows:

- Network-on-Chip/On Chip Network: The collection of switches comprising the network.
- Switch/Router: an intermediate module connecting to a processing resource and other switches to relay messages between resources or nodes.
- Resource/Node: one or more module connected to the same bus and has access to the network, through a network interface, to exchange messages. Usually it is a processing system with its own memory space and peripherals but could be any type or resource such as memory or special purpose Intellectual Property (IP) block.
- Network Interface (NI): an intermediate module connecting a resource to the network.
- Packet: the message in one or more words being sent or received in the network. Usually it consists of number of flits.

- Flit: originally short for flow control digit and is used interchangeably in this work with Phit, physical control digit. A flit is the actual bits propagating from or to the NI through the switches. Flits can be carrying setup or data words.

- Routing scheme: algorithm used to decide the path of a flit propagation through the network, for example, XY routing scheme in the Nostrum NoC routes flits horizontally until it reaches its X position and then vertically until it reaches its Y position.

- NoC Topology: The connection between the switches, usually in shape of mesh, torus, star, tree, etc.

- NoC Flow Control: the procedure employed to for transmission and reception of packets to allow correct message delivery between NI and switch and between switches. Examples for flow control includes: buffer-less flow control, store and forward flow control, wormhole, etc.

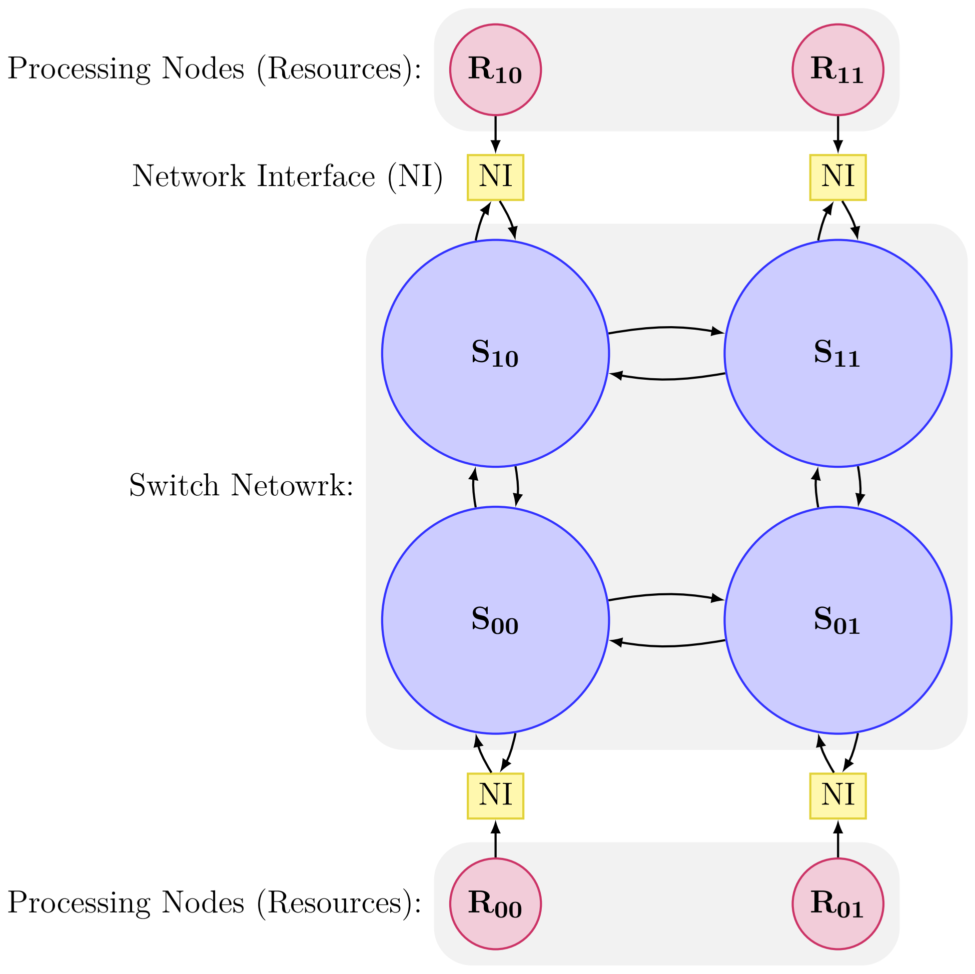

Figure 1.1: Architecture of Network-on-Chip Depicting Resources, Network Interfaces and Switch Network for 2x2 Mesh Type NoC

In the following parts of this document, an implementation of a Nostrum NoC will be described in more details.

Chapter 2

Principle of Operation

The Nostrum NoC base design is a two dimensional mesh array whose message passing follows a synchronous time-triggered manner. The architecture and operation mechanism are described in [1, 2]. A NoC has been initially developed with the aid of a Network-on-Chip System Generator (NSG) [3] for Xilinx Virtex 6 FPGA series [4] using Embedded Design Kit [5] of Xilinx ISE design flow methodology [6]. Since the MPSoC Zynq chip has reprogrammable FPGA different from Virtex 6, further adaptation and adjustment works on Xilinx Vivado and Xilinx Software Developement Kit tools have been made [7, 8].

Since the Nostrum NoC can have several implementations that could vary in their operation details, it is deemed appropriate to explain the principle of operation of the chosen base design. The principle can be explained as follows:

- NoC configuration such as mesh topology and size, number of nodes, node type, packet size, flit size, bus interface, processes per node, and the virtual communication direction between processes are made at design time.
- The message transaction through the network is built upon the philosophy of process-to-process communication. This means, the packet or message source is a process and the destination is a process.
- Each process has an ID which is an integer number defined at design time by the NSG.
- The processes can configure and send messages through the NoC by accessing the memory mapped NI peripheral. The address range of the NI includes a range of addresses dedicated for control registers

responsible for initiating the inter-process communication and allowing NoC status monitoring, reconfiguration and process synchronisation. The rest of the NI memory range is dedicated for buffering incoming packets (inbox) and outgoing packets (outbox). The inbox and outbox are divided into smaller subsets called channels, each of which is dedicated for a process. Each channel has an ID decided at design time. The binding of proceses to inbox and outbox channels is also decided at design time by the NSG.

- All NI addresses and parameters are symbolically accessible and defined in a C header file. NI control functions are provided in another C. The collection of these files is the NoC driver.
- The process within a node starts a transaction to another process by putting data into the respective memory mapped address range referred to as outbox channel.
- Processes are held in suspended state until a 'HeartBeat' signal is received. The HeartBeat is nothing but a global low frequency pulse generated at the NoC switch network tuned to provide a synchronisation support for periodic processes.
- A process can initiate a 'send transaction' by writing control info on the control register at the NI. The info is a 32-bit word encoding the ID of process destination, process source ID, send channel ID, message length, message priority.
- The NI divides the packet, i.e. the entire message into words, each can be called a flit. The flit is a word, 32-bit in size, with an extra header describing the routing information explained previously.
- The send transaction starts by sending two flits incorporating information about the time of message insertion and message length.
- The packet send transaction from the NI side can be described as follows:
  - The destination process ID is used to fetch physical address of the switch connected to the node holding that process. An image of the process to node mapping table is stored locally at all the NI. This information is used repeatedly for any outgoing flit.
  - At this stage, the status of the transmitter control logic changes from 'idle' to 'read input'.

  - The NI prepares the first flit and marks it as a setup flit with the time instance and injects it to the input of the switch when its time slot is granted. The time slot of each NI is determined and calculated in advance such that network can not experience congestion. Subsequently the NI transmitter changes its state to 'wait for read signal' from the switch.
  - Once the NI receives the read signal acknowledging that the switch successfully manages to get the flit, then the NI goes into a 'delay state' to wait until the NoC is free to receive another flit.
  - Once the delay period elapsed, the transmitter logic goes into 'read memory' state in order to: first send further timing information and subsequently send flits by from the outbox memory. A counter is incremented with each transition to 'read memory' whose value is used as an offset to point to the exact memory location in the outbox channel.
  - If all flits are received then the transmitter goes to 'idle' state, otherwise, it goes into 'wait for read signal' to receive any remaining flit.
  - The three steps are repeated periodically until eventually the packet is entirely sent. A state diagram showing the entire send process at the NI is shown in Fig. 2.1.

- The flits propagation through the switches can be described as follows:

  - Whenever a flit is made available at the switch, the switch starts looking at the routing information in the flit header in addition to all other flits coming from the other switches.
  - First, the destination physical location or address, i.e. column and row indices, are compared with the current switch column and row to decide on which direction should the flit be routed.
  - Second, based on XY routing algorithm and predetermined priority in case of contention in the destination, the destination of all flits are decided.
  - Third, if the flit is coming from the NI, then a Read signal is issued to acknowledge reading from the NI connected to the switch.
  - Fourth if one of the incoming flits is reaching the NI, then a write signal is issued.

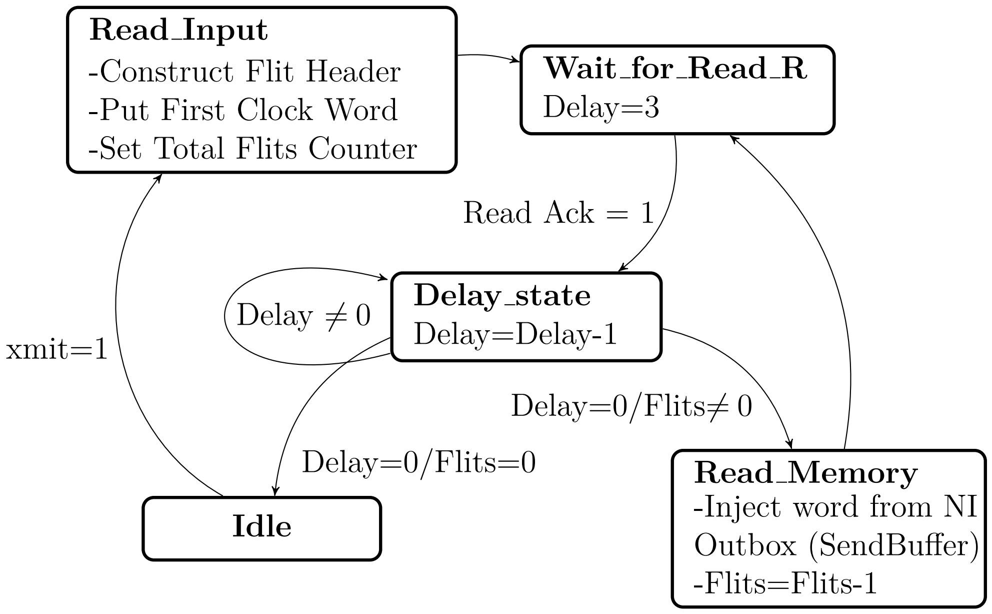

Figure 2.1: State Diagram for a Packet Transmission from NI

  - The last 4 steps are repeated at each switch till the flits finally arrive to its NI destination. This is depicted in the state diagram in Fig. 2.2.

- At the receiving NI, a receiving transaction is initiated once a 'write' signal is received from the switch. The following steps explain the details of the reception:
  - First, the flit is read to fetch the source process, destination process and packet length.
  - The source and destination processes are used to decide the right memory address or inbox channel corresponding to the particular source and destination process IDs.
  - If the incoming flit is of a 'Setup' type then the receiver logic state changes to 'Setup state'. In the setup state, the timing information from the setup flit is stored in a control register and the respective channel is opened. Subsequently, the receiver goes to wait for 'write signal'.
  - While being in 'wait for write signal state', the flit counter is updated to allow pointing to the right inbox channel address for the successive arriving flit(s). The receiver state changes to Idle.

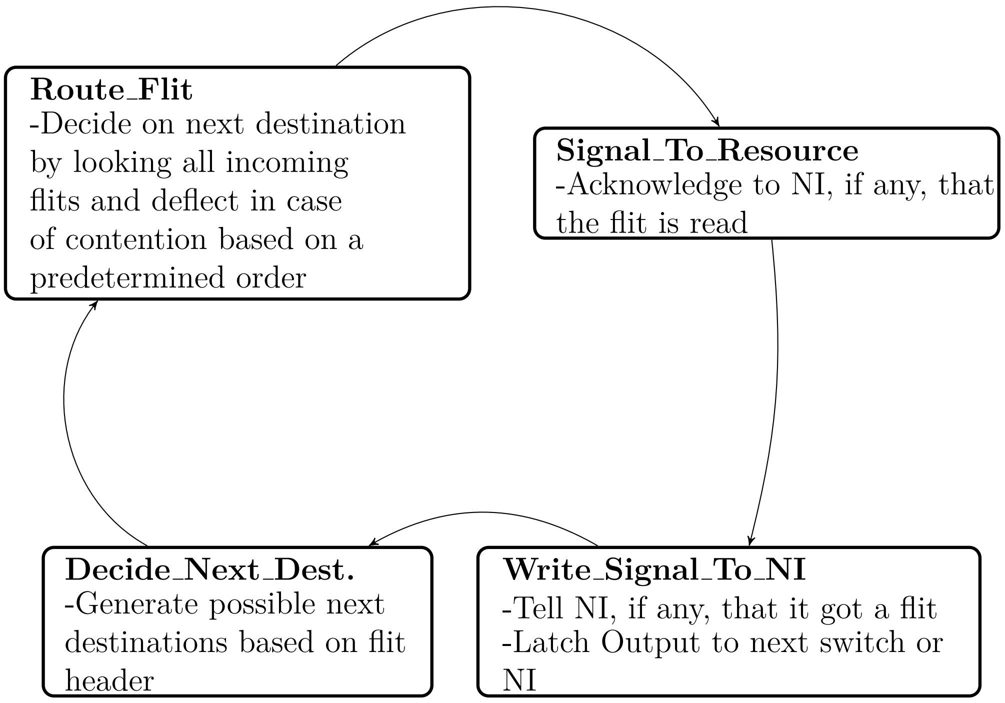
Route_Flit
-Decide on next destination by looking all incoming flits and deflect in case of contention based on a predetermined order
Signal_To_Resource
-Acknowledge to NI, if any, that the flit is read
Decide_Next_Dest.
-Generate possible next destinations based on flit header
Write_Signal_To_NI
-Tell NI, if any, that it got a flit
-Latch Output to next switch or NI

Figure 2.2: State Diagram for an Flit Propagation Through Switch Network

- A change in the status of the receiver from 'idle state' to receiver to 'write data' is triggered when 'write data' signal is received.
- Being in 'write data' state enables writing to the open inbox channel using the flit counter. Then the receiver goes back to 'wait for write signal state'.
- The last three steps are repeated till all flits are received therefore causing the inbox channel to be closed. The reception diagram is described in Fig. 2.3.

The packet flow is summarized in Fig. 2.4.

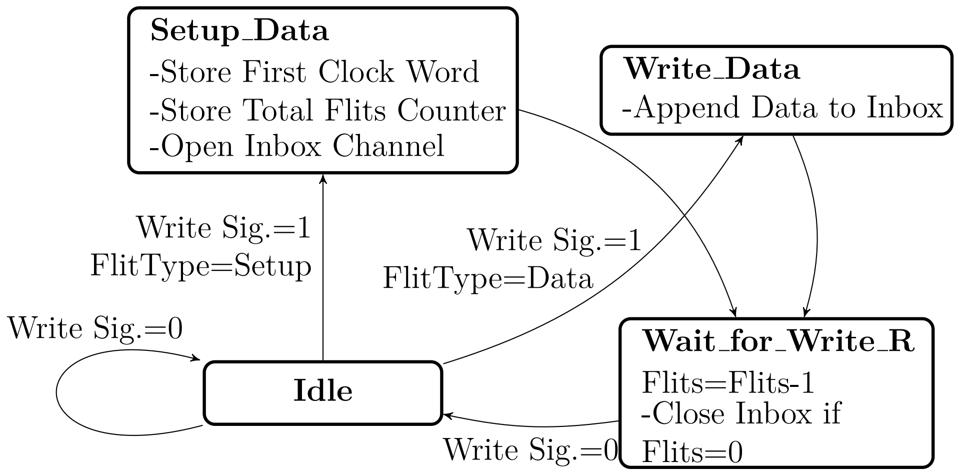

Figure 2.3: State Diagram for a Packet Reception to NI

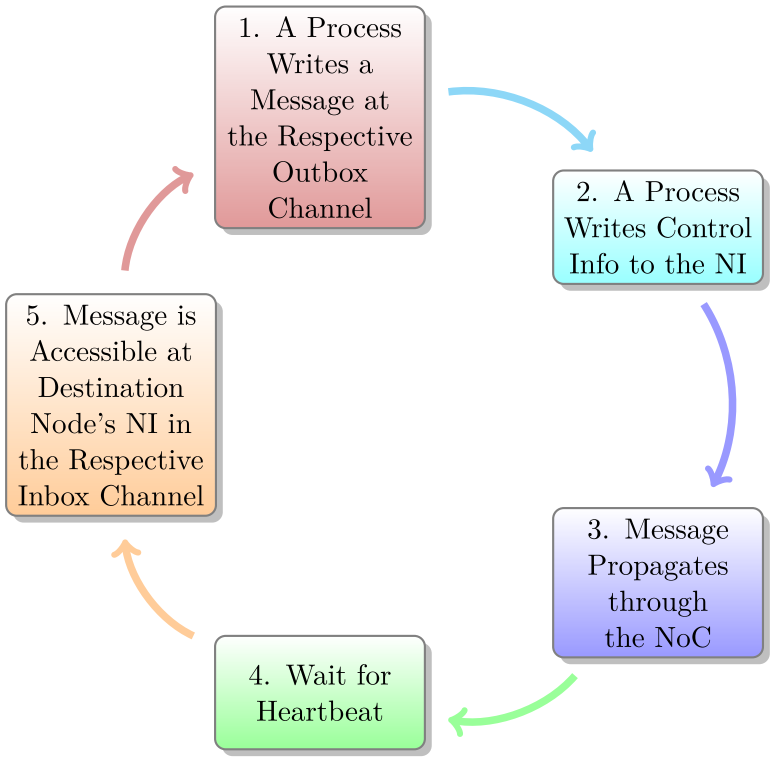
1. A Process Writes a Message at the Respective Outbox Channel
2. A Process Writes Control Info to the NI
3. Message Propagates through the NoC
4. Wait for Heartbeat
5. Message is Accessible at Destination Node's NI in the Respective Inbox Channel

Figure 2.4: Flow Diagram Summarising a Resource-to-Resource Message (Packet) Delivery Cycle from a Process Point of View

Chapter 3

Intellectual Property Blocks

3.1 Embedded Processor Nodes

The embedded processing nodes were the Dual Cortex A9 processor [9, 10] and Microblaze softcore processor [11]. A design flow adopted from [12] has been followed to instantiate and configure the processors accordingly. In favour of predictability and low resource utilization of Microblaze systems, the cache and error correction features have not been used.

3.2 Network-on-Chip IP Blocks

The NoC comprises two IP blocks: a network of switches called the NoC2x2 and the interface to the NoC switches which is called Network Interface, NI, which are described as follows.

The NI has two interfaces, direct access port (DAP) whose data bus width is 32 or 64 bit and AXI bus compatible with ARM AMBA bus specifications [13]. The DAP interface is intended to connect to resources such as a custom Very Large Instruction Word (VLIW) processor, whereas the AXI interface is compatible with ARM Application Processors and Xilinx Microblaze Microprocessor.

The NoC2x2 IP is a network of 4 switches each is connected to a resource through (54 bits bi-directional links and read/write/syc single-bit signals). The sync signal, otherwise called heartbeat signal, allows synchronous scheduling of processes and message passing. The configuration parameters for the NI and the NoC2x2 IPs are shown in Table 3.1 and Table 3.2 respectively.

The entire system however makes use of several other IPs which are listed in Table 3.3.

Table 3.1: Network Interface (NI) Configuration

Configuration Name	**Description**
NI Number	An identifier for the network interface. Each resource, usually a processor, should have a unique identifier for the interface to the network.
Base Address	The first accessible memory location within the NI, defaulted to 0x7002000
High Address	The last accessible memory location within the NI, defaulted to 0x7003FFFF
Address Range	High Address - Base Address + 1: 128KB
USE 64bit	Boolean number to indicate whether 64bits or 32bits Direct Access Port (DAP) will be used or not. DAP is intended to be used with custom Very Large Instruction Word (VLIW) microprocessor-kind of resource.

Table 3.2: Switches Network (NoC2x2) Configuration

Configuration Name	**Description**
HeartBeat Constant	A constant to set a periodic one clock-cycle signal, HeartBeat or Sync signal, every number of clocks (e.g. 5000000 corresponding to 1 Hz and 500 corresponding 10KHz for a system clock frequency of 50MHz).

Table 3.3: List of IPs Used in the System Implementation

IP	Version	Description
AXI GPIO	2.0.9	General Purpose Input/Output core provides a general purpose input/output interface to the AXI interface.
AXI Uartlite	2.0.11	Generic UART (Universal Asynchronous Receiver/Transmitter) with an AXI Interface
Clocking Wizard	5.2.1	The Clocking Wizard creates an HDL file that contains a clocking circuit customized to the user's clocking requirements.
MicroBlaze	9.5.3	The MicroBlaze 32 bit soft processor core, providing an instruction set optimized for embedded applications with many user-configurable options such as Instruction and Data-side cache with AXI interfaces, Floating-Point unit, Memory Management Unit, and fault tolerance support.
LMB BRAM Controller	4.0.7	Local Memory Bus (LMB) Block RAM (BRAM) Interface Controller connects to an lmb bus
Local Memory Bus (LMB) 1.0	3.0.1	The LMB is a fast, local bus for connecting MicroBlaze I and D ports to peripherals and BRAM
Constant	1.1.2	Gives a constant signed value
Block Memory Generator	8.3.1	The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs. It supports RAM and ROM functions over a wide range of widths and depths.
MicroBlaze Debug Module	3.2.4	Debug module for MicroBlaze Soft Processor.
Processor System Reset	5.0.8	Processor Reset System
AXI Crossbar	2.1.8	The AXI Crossbar IP provides the infrastructure to connect multiple AXI4/AXI3/AXI4-Lite masters and slaves.

3.3 Computer-Aided Design Automation

Hardware and Software development have been generated using Network-on-Chip System Generator (NGS) [3]. The generated system produces necessary files and folders for the IPs in a structure format compatible with Xilixn ISE design suite. The IPs have been restructured according to [14] to be compatible with Vivado Desing Suite [7, 8] and guide in [15].

The specific versions of tools that have been used are shown in Table 3.4.

Table 3.4: Versions of Tools Used in the Design and Development

Tool name	By	Version
Vivado Design Suite	Xilinx	2016.2
Software Design Kit	Xilinx	2016.2
NoC System Generator	KTH	1.00

3.4 Off Chip Communication

Chip-to-Chip communication can be done by another separate process. Such process can exploit the CAN controller or for instance the Gigabit Ethernet controller in conjunction with a Light weight Internet Protocol (lwIP) [16] as a driver for Ethernet based Chip to Chip communication as a physical communication media.

Chapter 4

Characterisation

Generally, packet propagation latency has an inter-dependent relation with the activities of the network and packet length in units of words (a flit carries one word of info). But due to the time-triggered operation of the NoC, the dependence on network activities is negligible and therefore, the packet latency, T_{packet}, follows this equation.

$$T_{packet} = T_{\text{Setup}} + T_{\text{flit}} \times flits \tag{4.1}$$

Or,

$$T_{\text{flit}} = 2T_{\text{NI to NoC}} + T_{\text{Switch to Switch}} \times N_{\text{switches}} \tag{4.2}$$

Where:
T_{flit}: Total propagation latency for one flit from sending NI to receiving NI.
T_{Setup}: Setup time required before the actual flits can be sent.
$T_{\text{NI to NoC}}$: Flit propagation latency within the NI until the flit is injected at the Switch.
$T_{\text{Switch to Switch}}$: Flit propagation latency within the switch receiver and transmitter until the flit is injected at the subsequent Switch or NI.
N_{switches}: The number of switches the flit has to traverse to reach the destination NI. Also known as the number of hops or hops counter.
$flits$: Packet size in term of number of flits or words, W.

Table 4.1: NoC Timing Parameters

Parameter	Value (Cycles)
$T_{\text{NI to Switch}}$	16
$T_{\text{Switch to Switch}}$	4
$T_{\text{Switch to NI}}$	4

The NoC timing parameters for NI to Switch, Switch to Switch and Switch to NI are reported in Table 4.1. $T_{\text{NI to Switch}}$ encapsulates the worst case delay imposed by the time-trigger mechanism of the NoC in addition to the access time to fetch data from the memory. $T_{\text{Switch to Switch}}$ includes the timing in the switch's receiver and transmitter. $T_{\text{Switch to NI}}$ includes time to fetch data from the switch and store it at the right memory address. Those numbers can be used to formulate timing analysis for various NoC topologies. For 2x2 NoC, the maximum delay can be taken from diagonal nodes can thus be described as follows:

$$T_{packet} = 24 \times W \tag{4.3}$$

$$T^{'}_{packet} = 48 + 24 \times W \tag{4.4}$$

Where T_{packet} is in the units of cycles and W is the packet length in words (1 word is 32 bit). The 48 cycles are actually for the first two flits which are used to relay the time at which the packet was inserted and the length of the packet.

Implementation wise, the design was synthesized at 50MHz on ”xc7z020 clg484-1” chips. Reports on resource utilisation, timing, energy consumption are extracted from Vivado. The longest path delay is found on the routing table that is used for extracting the destination process ID based on the channel source. This signifies a bottleneck for larger designs.

The packet propagation latency in the NoC, energy consumption per flit, and resource utilization are shown in Table 4.2 and Table 4.3. In these Tables, a hypothetical case of inter-process communication, via shared memory, assuming the use of AXI Cross bar peripheral and round-robin arbiter considered. The round robin arbitration gives a slot of 4 cycles per processor to access a shared memory and the arbiter logic itself is not given and considered to be negligible. This is used to enable comparative analysis. The tables also give statistics for CPU (Xilinx Microblaze) and its embedded memory (64KB).

Table 4.2: Resource Utilisation

	CPU	Memory	NoC	Shared Memory
LUTs	1115	7	2768	484
Registers	2272	13	6723	1016
BRAM	0	16	8	1

Table 4.3: Performance Metrics

Metric	NoC	Shared Memory
Dynamic Power (Watt)	0.012	0.004
Static Power (Watt)	0.006966	0.004128
Latency Per Transaction (Cycles)	24	16
Latency Per Transaction (micro Sec.)	2.083	3.125
Energy Per Transaction (mJ)	39.513	25.4

The static power consumption is derived by multiplying the resource utilisation ratio of the NoC times the whole device resources. The dynamic power consumption per module is provided directly from Vivado simulation. At this frequency the estimated dynamic power consumption is 0.012 Watt/Flit. The dynamic power consumption of the NoC depends on the number of switches which is 0.003 Watt/switch. Generally the statistics show better performance for the shared memory and that is expected for smaller designs. However, the difference is small enough in term of latency per transaction that one can reasonably expect an optimised NoC to have better performance for a design with more than 8 nodes. In term of energy efficiency, a design at the order of 10-20 nodes following method [17].

Bibliography

[1] W. H. Minhass, J. Öberg, and I. Sander, "Design and Implementation of a Plesiochronous Multi-core 4x4 Network-on-chip FPGA Platform with MPI HAL Support," in *Proceedings of the 6th FPGAworld Conference*. New York, NY, USA: ACM, 2009, pp. 52–57. [Online]. Available: http://doi.acm.org/10.1145/1667520.1667527

[2] F. Robino and J. Öberg, "The HeartBeat Model: a Platform Abstraction Enabling Fast Prototyping of Real-Time Applications on NoC-based MPSoC on FPGA," in *8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, July 2013, pp. 1–8.

[3] Johnny Öberg, "NoC System Generator Manual, Version 1.0 2014," User Guide, Information and Communication Technology School, Department of Electronic Systems, KTH Royal Institute of Technology.

[4] "Virtex-6 Family Overview, DS150 (v2.5) August 20, 2015," Product Specification, Xilinx, August 2015.

[5] "Embedded System Tools Reference Manual, EDK, UG111 (v14.6) June 19, 2013," User Guide, Xilinx, June 2013.

[6] "Synthesis and Simulation Design Guide, UG626 (v 14.4) December 18, 2012," User Guide, Xilinx, December 2012.

[7] "UltraFast Design Methodology Guide for the Vivado Design Suite, UG949 (v2015.3), november 23, 2015," User Guide, Xilinx.

[8] "OS and Libraries Document Collection, UG643 (v2015.4) november 18, 2015," User Guide, Xilinx.

[9] "Cortex™-A9 Revision: r4p1, Technical Reference Manual," ARM, 2012.

[10] “Cortex -A9 MPCore™®Revision: r4p1, Technical Reference Manual,” ARM, 2012.

[11] “MicroBlaze Processor Reference Guide,UG984 (v2015.4) November 18, 2015,” Xilinx, November 2015.

[12] “Zynq-7000 All Programmable SoC: Embedded Design Tutorial, UG1165 (v2015.4),” User Guide, Xilinx, November 2015.

[13] “AMBA AXI™and ACE™Protocol Specification: AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite,” ARM, November 2011.

[14] “Vivado Design Suite User Guide, Designing with IP: UG896 (v2016.1) April 12, 2016,” User Guide, Xilinx.

[15] “Zynq-7000 All Programmable SoC Software Developers Guide, UG821 (v12.0) September 30,” User Guide, Xilinx, September 2015.

[16] A. Dunkels, “Design and Implementation of the lwIP TCP/IP Stack,” *Swedish Institute of Computer Science*, vol. 2, p. 77, 2001.

[17] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, “On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip Approaches,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 3, pp. 23:1–23:20, May 2008. [Online]. Available: http://doi.acm.org/10.1145/1255456.1255460