Elaboration on the Nostrum Zero Release at F2F SAFEPOWER Meeting in Siegen

Mohamed Tage.[1]

[1]School of Information and Communication Technology, KTH Royal Institute of Technology, mtme AT kth DOT se

2016, Oct, 5

1 General View of the Network-on-Chip

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 such as bus with shared memory and point to point communication. 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. The figure depicts four resources or nodes which are either microblaze systems or dual cortex A9 processor, 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 can be 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.

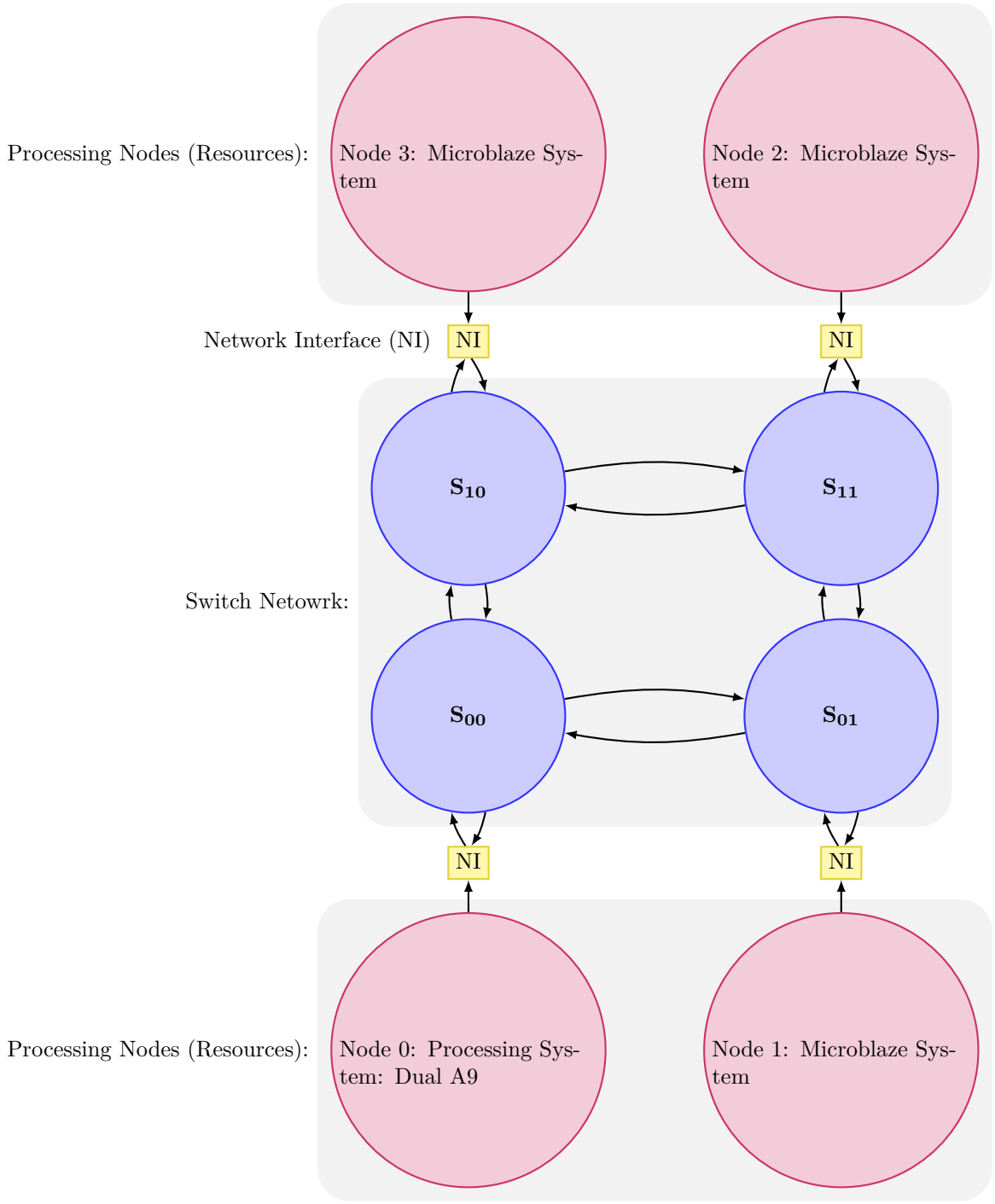

Figure 1: Architecture of Network-on-Chip Depicting Resources, Network Interfaces and Switch Network for 2x2 Mesh Type NoC

The NoC is seen from the processor side as a memory map device which can simply be reduced to 3 regions: control registers, inboxes and outboxes. For convenience, Inboxes and Outboxes are divided into 2 regions to spatially isolate messages of critical and non-critical processes. The memory map is shown in Figure 2.

Command/Control/Configuration Registers
Critical Process Region: Incoming Packets Channels (Inbox)
Critical Process Region: Outgoing Packets Channels (Outbox)
Critical Process Region: Incoming Packets Channels (Inbox)
Critical Process Region: Outgoing Packets Channels (Outbox)

Figure 2: Simple Memory Map for the Network Interface as viewed from the Processor

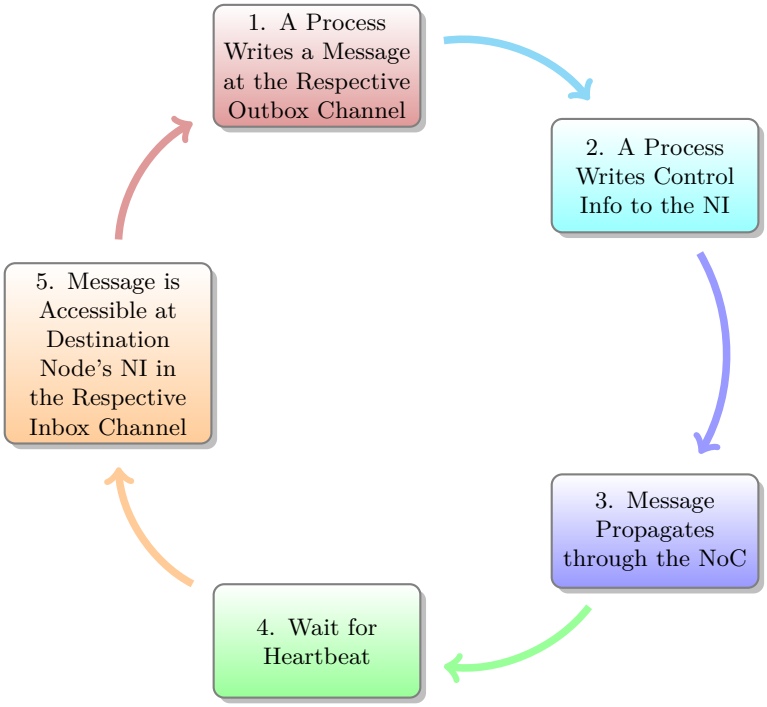

Figure 3: Flow Diagram Summarising a Resource-to-Resource Message (Packet) Delivery Cycle from a Process Point of View

A process in a processor can send a message to other processor by writing the message into the outbox and issue a send command telling the message destination at one of the control registers. The packet transmission and delivery is shown in Figure 3.

2 Implementation Example as Delivered in the Zero Release

To show case a potential energy-efficient NoC based platform, an example with emphasis on demonstrating dynamic process relocation is considered. This means, the NoC can support the delivery of messages of communicating processes even if some processes are subject to relocation. To be acquainted to the context out of which the example is draw, it is advised to refer to Figure 4.

Several use cases can be considered for showing the capacity of this work:

1. Intra-Chip Dynamic Process Relocation for critical processes acting as source process attached to a redundant hardware input wiring. This could be beneficial for a use-case in which a sensor such as engine temperature sensor or engine piston position is subject to damage but the redundancy is made locally within the chip.

2. Intra-Chip Dynamic Process Relocation for critical processes acting as a destination process attached to a redundant hardware output wiring. This could be beneficial for a use-case in which an actuator such as a valve or a brake or fuel injector is subject to damage but the redundancy is made locally within the chip.

3. Inter-Chip Dynamic Process Relocation for critical processes acting as source process attached to a redundant hardware input. This could be beneficial for a use-case in which a sensor such as airflow sensor is subject to damage but the redundancy is made at a spatially distinct chip.

4. Inter-Chip Dynamic Process Relocation for critical processes acting as a destination process attached to a redundant hardware output. This could be beneficial for a use-case in which an actuator such as airbag bump initiator is subject to damage but the redundancy is made at a spatially distinct chip.

5. Inter-chip Dynamic Process Relocation for non-critical processes for controlling complex functionality that receives and sends signals to non-critical nodes. This could be beneficial for a use-case such as a game engine receiving input from a touch screen and controlling a monitor. Such process can be suspended and can experience less optimal quality depending on the load condition.

6. Inter-chip Dynamic Process Relocation for non-critical processes for controlling complex functionality that just receives signals from non-critical nodes. This could be beneficial for a use-case such as distance logger. Such process does not have to be updated often and can be suspended and resumed depending on the load condition.

7. Inter-chip Dynamic Process Relocation to change processes of non-critical functionalities to critical ones by replacing all non-critical processes by

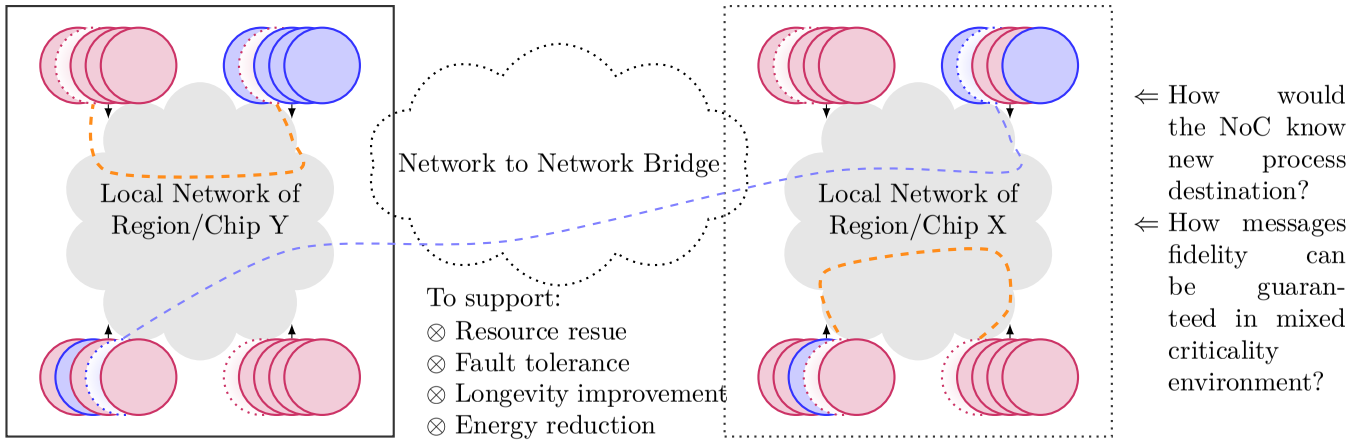

Figure 4: Problem Context: The Figure Poses a Question of How DPR Should Be Supported to Enable Efficient Execution of Expensive and Computationally-Intense Services, Distribute Processing Loads and Manage Hot-Sparring in the Context of Networked Real-Time Mixed-Critical NoC-based MPSoCs. In the Diagram, Critical and Non-Critical Processes are Indicated by Red and Blue Circles Respectively. Connected Dashed Circles Indicate Relocatable Processes.

Table 1: Power state configuration

High Performance/Balanced Load	Load is distributed, all features available
Minimal Energy Consumption	Non-critical processes are shut-off
Fault-Tolerance	Replication of P5 in another MPSoC

critical ones thus completely changing the criticality level of that node. This could be beneficial for a use-case such as disabling a non-critical node for the sake of accommodating critical demands mandated by faulty chip or load balancing.

Although all cases are demonstrable, for the sake of demonstration simplicity, switches and LEDs are used instead of sensors and actuators. Furthermore, only four scenarios are considered:

1. Intra-chip relocation of a critical process connected with input acting as source process while preserving the position of the destination process.
2. Intra-chip relocation of a critical process connected with an output acting as a destination process while preserving the position of the source process.
3. Inter-chip relocation of a non-critical process connected with two other processes one is a source connected to an input and the other is a destination connected to an output. Both source and destination processes are fixed while the middle process can relocate from the Microblaze on the FPGA to the non-critical section of the ARM to another ARM off chip to another FPGA area off-chip.
4. Inter-chip relocation of a critical process such that it replaces a node occupied by non-critical process on another FPGA area off-chip.

The implementation scenarios have been explained in Fig. 5.

3 Envisioned Implementation

The envisioned implementation on the NoC side can be thought as a system that has two power-state configurations. High power state enabling high performance and Low power state enabling minimal safety-critical performance.

The performance characteristics can be described as follows:

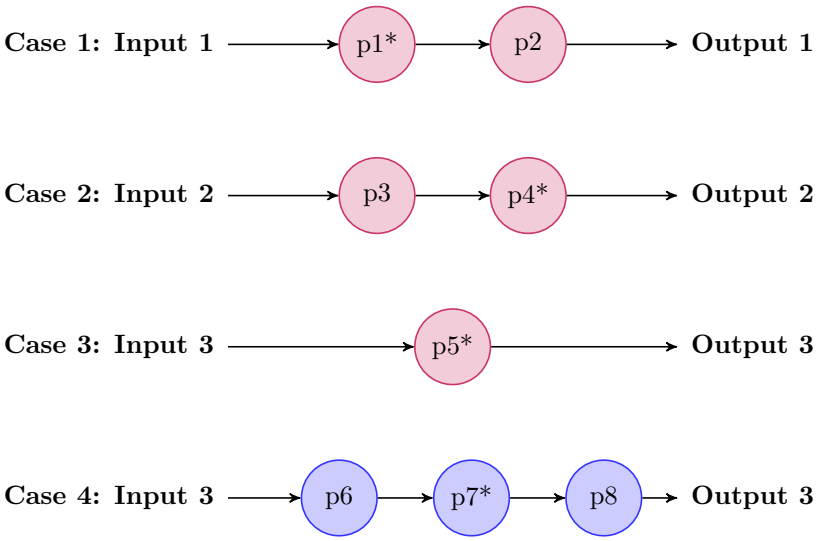

Figure 5: Implementation scenarios showing inter-process communication. Processes p1,p2,...,p5 are coloured in red to denote their high criticality level whereas p6, p7 and p8 are coloured in purple to indicate their low criticality level. The asterisk on processes denotes that they are subject to relocation with implications on how their communication channels is physically made

4 Appendix: Extended Design 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.

$$T_{packet} = T_{\text{Setup}} + T_{\text{flit}} \times flits \tag{1}$$

Or,

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

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.

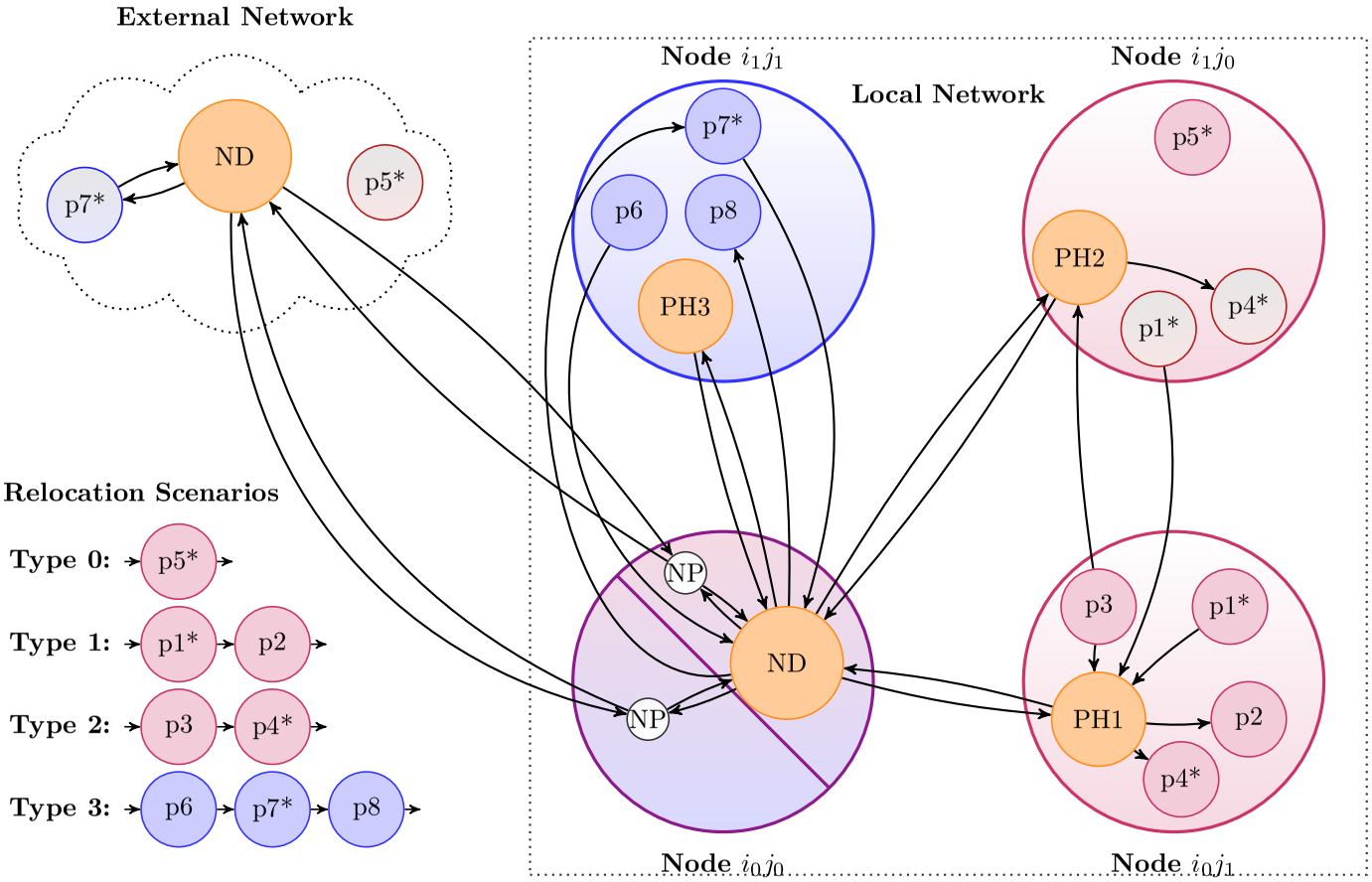

Figure 6: NoC Example Depicting Process Communication in the NoC. The orange colour signifies the supervisory process handler responsible of overseeing the communication and process activation/deactivation in each node

Table 2: Performance Metrics

Metric	NoC
Dynamic Power (Watt)	0.012
Static Power (Watt)	0.006966
Latency Per Transaction (Cycles)	24
Latency Per Transaction (micro Sec.)	2.083
Energy Per Transaction (mJ)	39.513

$flits$: Packet size in term of number of flits or words, W.

Table 3: 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 3. $T_{\text{NI to Switch}}$ 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:

$$T_{packet} = 24 \times W \tag{3}$$

$$T^{'}_{packet} = 48 + 24 \times W \tag{4}$$

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.

Due to the possibilities of dynamic relocation, packets could be subject to forwarding thus introducing an additional latency to T_{packet}. The new latency due to forwarding effects ($T_{forwarding}$) and the overall latency can be denoted as $T^{forwarding}_{packet}$. Since the forwarding is done within the process handlers/NoC Daemon, it is highly dependent on the software implementation of the forwarding. Mainly, the time required to check the status of the destination and source

processes, and the time required to point to the respective inbox/outbox channel. The additional latency is dependent on the processor frequency as well. The $T_{packet}^{forwarding}$ can be described as follows:

$$T_{packet}^{forwarding} = T_{packet} + T_{forwarding} \tag{5}$$

$T_{forwarding}$ can be characterised at software development time but usually is considerably smaller than T_{packet}.

Implementation wise, the design was synthesized at 50MHz on "xc7z020 clg484-1" chips.

The packet propagation latency in the NoC, energy consumption per flit, and resource utilization are shown in Table 4 and Table 2. The tables also give statistics for CPU (Xilinx Microblaze) and its embedded memory (64KB).

Table 4: Resource Utilisation

	Microblaze	64KB Microblaze Memory	NoC
LUTs	1115	7	2768
Registers	2272	13	6723
BRAM	0	16	8