Network-on-Chip Benchmarking Specification Part 2: Micro-Benchmark Specification Version 1.0

Zhonghai Lu ⋆, Axel Jantsch ⋆, Erno Salminen † and Cristian Grecu *
⋆ Royal Institute of Technology, Sweden
† Tampere University of Technology, Finland
* University of British Columbia, Canada

May 23, 2008

Abstract

The rapid development of Network-on-Chip (NoC) calls for a systematic approach to evaluate and fairly compare various NoC architectures. In this specification, we define a generic NoC architecture, a comprehensive set of synthetic workloads as micro-benchmarks, workload scenarios and evaluation criteria. These micro-benchmarks enable to measure and pinpoint particular properties of NoC architectures, complementing application benchmarks.

Keywords: Network-on-Chip, Performance Evaluation, Benchmark

Acknowledgements

The authors would like to acknowledge the contribution of other members of the Network-on-Chip Benchmarking Working Group at OCP-IP: André Ivanov (University of British Columbia), Partha Pande (Washington State University), Radu Marculescu and Umit Ogras (Carnegie Mellon University), Pascal Chauvet (Sonics), and Yasuhiko Kurosawa (Toshiba).

Valuable comments and discussions were provided by members of other OCP-IP working groups: Steven Mcmaster (Synopsys), Vesa Lahtinen (Nokia), Mark Burton (GreenSocs), Srinivasan Krishnan (Sonics), and Jeff Deutch (Texas Instruments).

Special thanks are due to OCP-IP President, Ian Mackintosh, for his continuous and active support of this initiative.

Contents

1 Introduction **3**

2 Architecture Definition **4**
2.1 The NoC model under evaluation 4
2.2 Communication entities: transaction, transfer and packet 5
2.3 Standard communication interfaces 6

3 Traffic Configuration **7**
3.1 Temporal distribution 7
3.2 Spatial distribution 7

4 Measurement **10**
4.1 Workload and metrics 10
4.1.1 The unloaded case 10
4.1.2 The loaded case 10
4.1.3 Impact of resource reservation on best-effort traffic 11
4.1.4 Network scalability 11
4.2 Measurement configuration 11

5 The Micro-Benchmark Set **13**
5.1 Naming convention for micro-benchmarks 13
5.2 Micro-Benchmark implementation 14

6 Concluding Remark **15**

Bibliography

Chapter 1

Introduction

Micro-benchmarks define synthetic workloads intending to exercise a Network-on-Chip (NoC) in a specific way or measure a single particular aspect. Hence, a measurement offers insight in a specific property and facilitates the analysis and design of a communication infrastructure. A single micro-benchmark provides only a very limited view and does not allow for far reaching conclusions about the suitability for an application domain. However, a set of well designed micro-benchmarks can give both a broad and detailed understanding of a given communication network. Since even a large set of benchmarks cannot guarantee to represent a real application well, micro-benchmarks are complementary to application benchmark. While benchmark programs evaluate the combined effect of many aspects of the platform as well as of the application, micro-benchmarks isolate individual properties and allow for a faster and deeper point analysis.

Micro-benchmarks accept a set of parameters that define traffic characteristics to evaluate NoC performance. A sweep is performed by varying the parameters within allowed range of values. This enables to covers a wide spectrum of traffic scenarios [1]. Here, the term performance refers to functional metrics such as delay and throughput, and non-functional metrics such as power/energy consumption and area. While throughput should be maximized, delay and non-functional metrics should be minimized.

This document defines a set of micro-benchmarks as part of OCP-IP NoC benchmark suite [2].

Chapter 2

Architecture Definition

In this chapter, we describe the network-on-chip under evaluation. The network is assumed to be wrapped by a hardware interface conforming to the OCP protocol [3] or a third-party protocol, for example, the AXI protocol [4]. The basic communication entity are read/write transaction at the hardware interface and the packet in the network.

2.1 The NoC model under evaluation

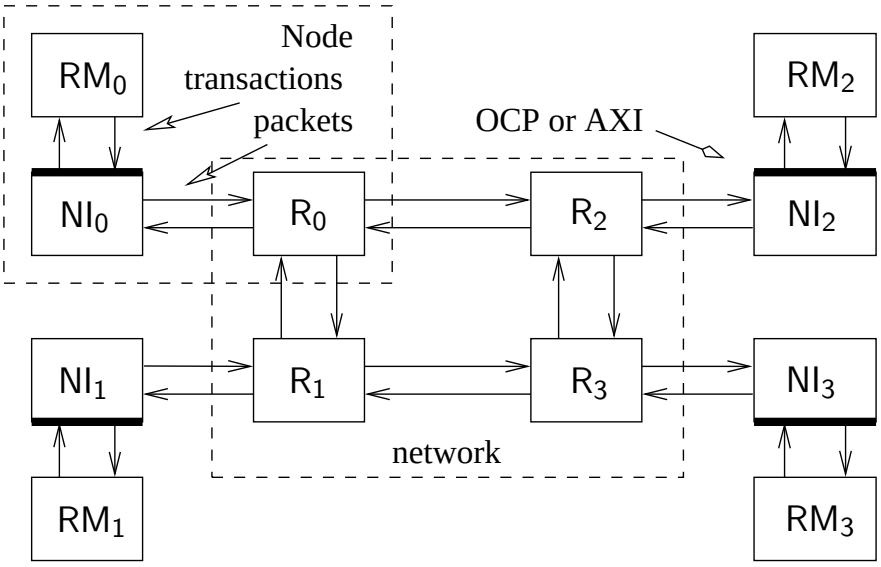

Figure 2.1: An example of network-on-chip with four nodes

In a wide sense, NoC is a System-on-Chip using a network for global communication between resources. In a narrow sense, NoC refers only to the on-chip network. In this document, we define micro-benchmarks to evaluate the on-chip network. Resources can be a processor, a memory, a configurable or dedicated logic, or a local bus-based subsystem. We illustrate the model of NoC to be evaluated with an example in Figure 2.1. It shows four nodes connected by a packet-switched network. A node consists of a Resource Model (RM), a Network Interface (NI) and a Router (R). In our context, a resource is attached to exactly one NI, which in turn connects to exactly one router. But a router may have no NI and RM connected to it in the case of indirect networks. The NI provides a hardware interconnect interface implementing an existing on-chip communication protocol, such as OCP and AXI. The network shown in Figure 2.1 has a 2D mesh topology. However, in general, it can be of any regular or irregular topology. Similarly, no assumptions are made about other network attributes such as routing algorithm, switching policy and flow control scheme.

The network may offer two classes of communication services: *best-effort* and *guaranteed services*. The best-effort service (BE) is connection-less, delivering packets in a best-effort fashion. It has no establishment phase, and sources send packets without the awareness of states in destinations. The guaranteed service (GS) is connection-oriented, providing certain bounds in latency and/or bandwidth. A connection is a unidirectional virtual circuit setting up from a source NI to a destination NI via the network. The network reserves resources such as buffers and link bandwidth for connections. The NIs manage connections in terms of establishment, configuration and tear down.

The evaluation is concerned with the network and NIs including the interconnect interface, such as OCP or a third-parity interface. In order to access a specific network service, the interface may need to be extended. The scope of evaluation is on unicast communication.

2.2 Communication entities: transaction, transfer and packet

We identify three communication entities at the three levels. Specifically, at the application level, the unit of communication between RMs is *transaction*; At the interconnect level, the unit of communication between RMs and NIs is *transfer*; At the network level, the communication unit between NIs and routers is *packet*. A transaction is composed of one or multiple transfers, which are sent to NIs. In NIs, transfers are encapsulated into packets, which are then sent from NIs to the network. After network delivery, packets are received by destination NIs where packets are de-encapsulated back into transfers.

Type	Level (between)	Direction	Length
Transaction	RM-RM	Uni-, Bi-directional	Varying
Transfer	RM-NI	Unidirectional	Fixed
Packet	NI-R	Unidirectional	Fixed, varying

Table 2.1: Communication units

Table 2.1 lists and compares the communication entities. The main difference between transaction and transfer is the length. A transaction, which may be bi-directional, is of variable length while the size of a transfer on a given architecture is fixed. The difference between transfer and packet lies in that a packet is a transfer (as payload) plus overhead (head and tail). For example, a read transaction is a roundtrip, i.e., sending the read address to the memory and loading the requested number w of words from the memory. The read address is sent across the NI as one transfer. The w-word data are sent from the memory to its associated NI as w transfers. One transfer is one word, which may be encapsulated into one packet by adding head and tail bits.

As packet is the data unit of the network, investigating the performance of individual packets and the network throughput are essential. By measuring delay, jitter, energy consumption, etc. of individual packets, the network itself, its routing, switching, buffering and flow control mechanisms are evaluated. It should also be distinguished between the latency in the network and delay for network access. Transactions are communication activities between RMs. They are used to evaluate packetization, end-to-end flow control, streaming capabilities and similar services of the network. In addition to the network properties, transactions also evaluate the performance of the interface blocks between the network and the resources. Examples of transactions are: memory read and write, opening and closing of connections, and message transmission over an open connection.

To well-define the communication entities for micro-benchmarks, we further specify communication entities for BE and GS services. For BE services, we look at three workload types: *packet*, *read transaction* and *write transaction*. Both read and write transactions are bi-directional. For GS services, we examine another three workload types: *open connection*, *close connection* and *message*. Here message is the data transmitted over connections. We shall use these workload types in Section 4.1 of Chapter 4.

2.3 Standard communication interfaces

To make a benchmark reusable for analyzing different NoCs, both the benchmark and the NoC have to adhere to a standard interface definition. Though our micro-benchmarks are independent of hardware interfaces, we allow two different types of popular interfaces: OCP [3] and AXI [4].

Open Core Protocol (OCP) is the OCP-IP Association's protocol definition and AXI is ARM's AMBA Advanced eXtensible Interface protocol. Both are popular and have sufficient high level and advanced concepts such as multiple outstanding, pipelined and out-of-order transactions to be suitable interfaces for NoCs. All benchmark programs and micro-benchmarks accepted in the NoC benchmark suite shall adhere to either one of these two protocols.

Chapter 3

Traffic Configuration

Micro-benchmarks stress the network with various different traffic scenarios in order to study how the NoC performance depends on the traffic patterns. We define synthetic workloads for evaluation, differentiating between temporal and spatial distribution of traffic generation.

3.1 Temporal distribution

The temporal distribution determines how an individual RM generates traffic over time. Based on the **b**-model [5] the burstiness of traffic generation can be controlled by a simple parameter b in the range $0 < b \leq 0.5$. In the **b**-model, a bias parameter $b = 0.4$ means that, within a given time interval, 40% of the data are generated in one half of the time interval and the remaining 60% in the other half, and this continues recursively until reaching the time resolution. More specifically, the whole construction begins with a uniform interval and recursively subdivides the number of data to be generated to each half, quarter, eighth, etc. according to the bias b. When $b = 0.5$, there is no burstiness and the emission probability is constant. When b is approaching 0, the burstiness increases.

Four different distributions shall be covered by micro-benchmarks:

1. **Bursty traffic type 1**: $b = 0.5$: The probability of traffic emission does not vary over time, giving constant probability.
2. **Bursty traffic type 2**: $b = 0.4$
3. **Bursty traffic type 3**: $b = 0.3$
4. **Bursty traffic type 4**: $b = 0.2$

3.2 Spatial distribution

The spatial distribution governs the spatial traffic pattern: who communicates with whom. The following spatial distributions shall be covered:

1. **Uniform**: In this classic case the probability to send a packet from one node to another node is 1/(N-1), assuming N nodes in the network. A node does not send data to itself.
2. **Locality**: The probability P to send a packet to a destination node depends on the source-destination distance [1] d as follows:

$$P(d) = 1/(A(D)2^d) \tag{3.1}$$

[1] Distance is defined as the number of links from a source router to a destination router.

where D is the maximum distance in the network and $A(D) = \sum_{d=1}^{D}(1/2^d)$ is a normalizing factor guaranteeing that the sum of all probabilities is 1. Within a set of nodes with the same distance, each node is selected with uniform probability.

For example, assume that, in a network, the maximum distance between a source node and all other nodes is 2. Then for this node, $D = 2$, $A(2) = 3/4$. Therefore $P(1) = 2/3$ and $P(2) = 1/3$. This means the probability of this node sending traffic to its destination nodes with distance 2 is 2/3, while the probability of sending traffic to a node with distance 1 is 1/3.

3. **Bit Rotation**: When a destination node is selected by a function of the source address, we obtain a *bit permutation* pattern [6]. This means a given source node sends only to one destination node and the destination node address is a function of the source node address. The *bit rotation* pattern means that the destination address is obtained by rotating the bit string representation of the source node address to the right by one.

   Suppose that the number of nodes N is a power of 2, and m is the number of bits used to express the addresses of source and destination nodes, bit permutations are those in which each bit d_i of the m-bit destination address is a function of one bit of the source address, s_j, where j is a function of i. For the bit rotation pattern, $d_i = s_{i+1 \bmod m}$. For example, if a source address is "0111", then its destination address is "1011".

   If N is a power of 2, given a source node, its destination node uniquely exists; Otherwise, the destination node address is determined by $destination\_address = (bit\_rotation(source\_address))\ modulo\ N$.

4. **N Complement**: Similarly to **Bit Rotation**, this scenario creates load on source-destination pairs. Suppose that nodes are numbered as naturals 0, 1, 2, ... N-1, if a source node address is n_s, its destination address is n_d such that $n_s + n_d = N$. For example, if a network has 7 nodes, the nodes are numbered from 0, 1, ..., 6. Nodes 0, 1, 2, 3, 4, 5, 6 will select nodes 6, 5, 4, 3, 2, 1 as destination nodes, respectively.

   Note that, for the two permutations traffic **Bit Rotation** and **N Complement**, how nodes are numbered determines the actual spatial distribution of traffic. It is advised that nodes should be regularly numbered according to their topology. This also implies that these two traffic patterns are most beneficial to regular topologies.

5. **Hot Spot**: This scenario selects $\lfloor N/M \rfloor$ [2] of the nodes (N is the total number of nodes) as hot spots, $M \in \{2,4,8,...,N\}$. Certain fraction ρ ($\rho \in \{0.5, 0.7\}$) of traffic is targeted to these hot-spots (one is selected at a time by uniform random selection). The other traffic is sent uniformly to all other nodes. Both number of hot spot nodes M and fraction ρ are user-defined parameters. A variant of this scheme selects different hot-spot for each source.

6. **Fork-Join Pipeline**: A fork-join pipeline [7] is a pattern where a fork node feeds c nodes that are the starting point of c parallel pipelines. Each pipeline has a depth of e nodes. At the end of the pipelines after e stages, the data is merged into a join node. An example of a fork-join pipeline in a 4×4 mesh network with $c = 3$ and $e = 3$ is shown in Figure 3.1. In general for 2D meshes, c is set to $c = h = \lfloor \sqrt{N-2} \rfloor$. Hence, for a 4×4 network, we have $c = e = 3$. It is possible the only few nodes are observed in the measurement and other nodes generate back-ground (noise) traffic with some micro-benchmark pattern.

[2] $\lfloor x \rfloor$ returns the greatest integer less than or equal to x

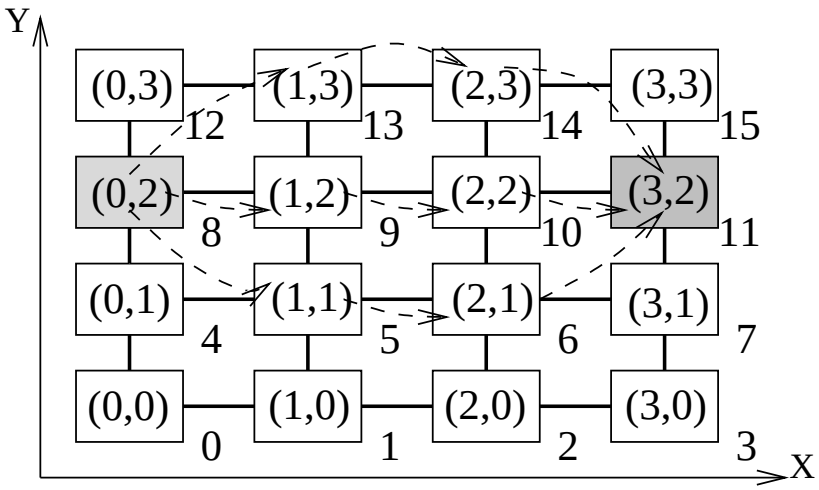

Figure 3.1: A fork-join pipeline with 3 parallel branches and a depth of 3, i.e., $c = 3$ and $e = 3$

If a network features a 2D (mesh, torus, fat tree, irregular, etc.) or 3D (cube, irregular etc.) topology, we map the nodes' coordinates to integers in order to compare different topologies. For example, as illustrated in Figure 3.1, a node (x,y) maps to an integer I by $I = x + y \cdot N_y$, where N_y is the maximum number of nodes along the Y axis for the 2D mesh. The user may define his own mapping from 2D/3D coordinates to integers.

The set of micro-benchmarks described above will systematically exercise a set of important aspects of a NoC. It will give the NoC developer insight and guidelines for improvement. It will also give the NoC user a detailed understanding of the NoC behavior, its strengths and weaknesses.

Chapter 4

Measurement

We consider different workload cases, for which we define measurement metrics. We also give measurement points in measurement configuration.

4.1 Workload and metrics

4.1.1 The unloaded case

In the unloaded case, individual unicast packets or transactions are injected/initiated and measured so that only a single traffic source is active. This gives data about minimum delay and peak performance. These values may vary depending on the location of the source. Therefore, determining the minimum, average, and maximum values require several measurements.

Table 4.1 shows the performance figures for the unloaded case, where **h** and **l** are the length of overhead (header plus tail) and payload in bits, respectively. The delay unit is nano-second (*ns*) or cycles. The latter can be used if the NoC is clocked by a single clock source, operating synchronously.

Service	Workload Type	**Delay** [*ns* or cycles]			**Throughput** [Mbits/s]			**Energy** [pJ]		
		Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
BE	Packet of **h**+**l** bits									
	Read 16/32/64 bits									
	Write 16/32/64 bits									
GS	Open connection									
	Close connection									
	Message 1/4/16/32 Bytes									

Table 4.1: Evaluation criteria for the unloaded case

4.1.2 The loaded case

The loaded case investigates the network behavior when many independent unicast packets or transactions compete for the same resources, and congestion, arbitration, buffering and flow control policies will be exercised. Sometimes part of the load in the network may be generated as background traffic with some other micro-benchmark than the one used for measurements, for example, the uniform traffic.

In the presence of congestion the network typically does not exhibit a deterministic delay behavior. To capture the delay of packets or transactions, we use the metrics $\bar{D}_1$, $\bar{D}_2$, $\bar{D}_3$ and $\bar{D}_n$. Suppose p is a packet or transaction, $\bar{D}_i$ $(i = 0, 1, 2, \cdots, n)$ is a family of delay values defined as follows:

$$1 - 10^{-i} \; of \; all \; p : \; delay(p) \leq \bar{D}_i \tag{4.1}$$

Service	Workload Type	**Delay** [*ns* or cycles]					**Jitter**				Θ_s [Mbits/s]	**Energy** [pJ]
		Avg.	$\bar{D}_1$	$\bar{D}_2$	$\bar{D}_3$	$\bar{D}_n$	Avg.	$\bar{J}_1$	$\bar{J}_2$	$\bar{J}_n$		
BE	Packet of **h+l** bits											
	Read 16/32/64 bits											
	Write 16/32/64 bits											
GS	Open connection											
	Close connection											
	Message 1/4/16/32 Bytes											

Table 4.2: Evaluation criteria for the loaded case

where $delay(p)$ is the delay for p. Thus, $\bar{D}_1$ bounds 90% of all packets or transactions, $\bar{D}_2$ bounds 99%, $\bar{D}_3$ bounds 99.9% and $\bar{D}_n$ bounds 100% of all packets or transactions.

To capture the delay variation (jitter) of packets or transactions, we use the metrics $\bar{J}_1$, $\bar{J}_2$, $\bar{J}_3$ and $\bar{J}_n$. Suppose p is a packet or transaction, $\bar{J}_i$ $(i = 0, 1, 2, \cdots, n)$ is a family of normalized jitter values defined as follows [8]:

$$1 - 10^{-i} \; of \; all \; p: \; (delay(p) - min\_delay(p))/min\_delay(p) \leq \bar{J}_i \tag{4.2}$$

where $delay(p)$ is the delay for p and $min\_delay(p)$ the minimal delay for p. Thus, $\bar{J}_1$ bounds 90% of all packets or transactions, $\bar{J}_2$ bounds 99%, $\bar{J}_3$ bounds 99.9% and $\bar{J}_n$ bounds 100% of all packets or transactions. $\bar{J}_i$ has no unit.

Table 4.2 shows the performance metrics for the loaded case, where Θ_s represents sustained throughput. These performance figures depend on the load level in the network. Thus, different micro-benchmarks have to cover different load levels of background traffic by, for example, controlling the traffic emission rate to the network. The following cases should be covered given as a fraction of the ideal throughput Θ_{ideal} of the network: 10%, 30%, 50%, 70%, 90%. The ideal throughput Θ_{ideal} is the maximum throughput that a network could carry with perfect flow control and routing [6]. It depends on the network topology and traffic pattern. In case it is difficult to derive the ideal throughput, the network bandwidth capacity should be used instead.

4.1.3 Impact of resource reservation on best-effort traffic

To study the effect of resource allocation by guaranteed services on best effort traffic performance, a set of micro-benchmarks shall therefore measure the best effort traffic performance when 10%, 30% and 50% of the bandwidth of each link in the network is allocated to a guaranteed service.

4.1.4 Network scalability

In addition to performance metrics such as delay and throughput, scalability is an important quality metric. To evaluate the scalability of the communication networks, a range of network size can be specified as an input to the micro-benchmarks. For example, we may consider the number of nodes to be increasing powers of 2, {2, 4, 8, 16, 32, 64, 128, 256, 512, ...}.

4.2 Measurement configuration

In addition to defining how to exercise a network, it is also important to unambiguously specify the measurement configuration and to define where to perform the measurements. For example, the sustained throughput must be measured at network's output terminals. Measuring at input terminals is misleading if network drops packets.

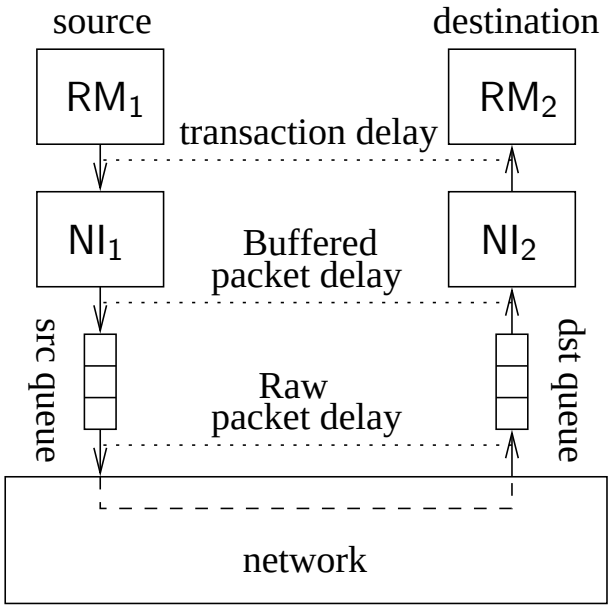

Figure 4.1: Measurement configuration

Figure 4.1 shows the measurement configuration and two different delays that have to be distinguished. The packet delay denotes the delay of a packet between the time its first bit is emitted into the network and when its last bit leaves the network. Usually, packets are buffered in the network interface (NI) if they cannot enter the network immediately. We consider open-loop traffic configuration in our measurement in which the network does not influence the packet injection, i.e., traffic is generated independent from the network behavior. By eliminating the possible interference feedback from the network, we concentrate on benchmarking networks. To this end, an infinite (”large enough”) FIFO must be placed between RMs and NIs, and between NIs and the network, or included in NIs. This ensures that traffic generation is never stalled and traffic characteristics appear as defined in the benchmark. Figure 4.1 shows the case that the FIFO is placed between the NIs and the network. Also, for transactions and longer messages the NI performs packetization, de-packetization, reordering of received packets and connection management. All packet related measurements shall be done between the NI and the network, while all transaction related measurements shall be done between NI and the RM.

We further distinguish two kinds of packet delay: *raw* and *buffered*. As shown in Figure 4.1, the raw packet delay is the delay that packets experience after being admitted into the network. The buffered delay is the raw delay plus the source queuing delay and the destination queuing delay.

To increase system performance, packets and transactions may be allowed to complete out-of-order. For example, scheduling by giving timing-constraints data higher priority in arbitration points may result in out-of-order completion. However, to enable fair comparison, networks must be compared with the same treatment of ordering, either in-order or out-of-order.

While measuring performance, network warm-up time should be excluded since the network is empty at this startup. The length of warm-up time for various traffic scenarios needs to be empirically obtained. Since our traffic sources are static, i.e., the traffic generation processes do not change over time, measurements should be taken after it has reached equilibrium.

Chapter 5

The Micro-Benchmark Set

We give the naming convention for the micro-benchmarks, and the complete benchmark set, which could be implemented in different modeling and programming languages.

5.1 Naming convention for micro-benchmarks

Each micro-benchmark has a name which reflects its function. The names have the following format:
nocmb_TEMP_SPAT_LUL_PAYLOAD_GS_SIZE_MP
Fields:

- **nocmb** is a constant string suffix standing for NoC micro-benchmark.
- **TEMP**: the temporal distribution (Section 3.1). It has the 12 possible values: B1-30, B1-50, B1-70, B2-30, B2-50, B2-70, B3-30, B3-50, B3-70, B4-30, B4-50, B4-70, which define a uniform or bursty emission probability with the average probabilities 30%, 50% and 70%.

  Note that these values cover only sampling points. The users may define a range of probability values, for example, from 0% to 70%, and provide a step value to increase the probability, in order to obtain sufficient resolution for performance measurements.
- **SPAT**: the spatial distribution (Section 3.2). It has the following 6 possible values: UNIFORM, LOC, BitRota, BitComp, HotSpot, ForkJoin.
- **LUL**: the workload case (Section 4.1.1 and 4.1.2). It has the 2 possible values: LOADED and UNLOADED.
- **PAYLOAD**: the payload type (Section 4.1.1 and 4.1.2). It has the 10 possible values: Packet, Read16, Read32, Read64, Open, Close, Message1, Message4, Message16, Message32.
- **GS**: the fraction of bandwidth reserved for the guaranteed service (Section 4.1.3). It has the following 4 possible values: GS0, GS10, GS30, GS50, which define how much of the total bandwidth is used for guaranteed service traffic, namely 0%, 10%, 30% or 50%.
- **SIZE**: the network size (Section 4.1.4). It has the following 9 possible values: 2, 4, 8, 16, 32, 64, 128, 256 and 512, which specify the number of nodes in the network.
- **MP**: the measurement point (Section 4.2). It has the following 2 possible values: RAW and BUFFERED, indicating where the performance is measured (see section 4.2, figure 4.1).

Hence, a complete set of micro benchmarks consists of 12 x 6 x 2 x 10 x 4 x 9 x 2 = 103680 programs. For instance, nocmb_U30_UNIFORM_LOADED_Packet_GS0_8_RAW would be one specific benchmark program (traffic generator).

5.2 Micro-Benchmark implementation

The micro-benchmarks may be implemented in different modeling or programming languages such as VHDL/Verilog, C/C++/SystemC. Depending on the networks-on-chip under evaluation, specific implementations are required.

NoC-WG (Work Group) at OCP-IP aims to provide an open source implementation of the micro-benchmarks in SystemC. We also encourage users to implement and share their micro-benchmark implementations in the community for the best benefits possible.

Chapter 6

Concluding Remark

In this document, we have specified a rich set of micro-benchmarks in order to evaluate and compare various emerging NoC platforms. In order to enable fair comparisons, we have defined the common features of the underlying NoC architectures on which various synthetic traffic patterns may play, and the measurement configuration.

We envision that this set of micro-benchmarks will continue to evolve together with increasing endeavors in NoC activities.

Bibliography

[1] John L. Gustafson and Rajat Todi. Conventional benchmarks as a sample of the performance spectrum. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, volume 7, pages 514–523, 1998.

[2] Cristian Grecu, Andre Ivanov, Partha Pande, Axel Jantsch, Erno Salminen, Umit Ogras, and Radu Marculescu. Towards open network-on-chip benchmarks. In *Proceedings of First International Symposium on Networks-on-Chip (NOCS'07)*, 2007.

[3] OCP International Partnership. Open Core Protocol specification, Version 2.2, Revision A. http://www.ocpip.org, 2008.

[4] ARM. AMBA Advanced eXtensible Interface (AXI) protocol specification, Version 1.0. http://www.amba.com, 2004.

[5] Mengzhi Wang, T. Madhyastha, Chan Ngai Hang, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: fast algorithms for modeling bursty traffic. In *Proceedings of the 18th International Conference on Data Engineering*, pages 507–516, 2002.

[6] William James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufman Publishers, 2004.

[7] Rikard Thid, Ingo Sander, and Axel Jantsch. Flexible bus and NoC performance analysis with configurable synthetic workloads. In *Proceedings of the 9th Euromicro Conference on Digital System Design*, August 2006.

[8] Axel Jantsch. Models of computation for networks on chip. In *Proceedings of the Sixth International Conference on Application of Concurrency to System Design*, June 2006.