

On network-on-chip comparison

Erno Salminen, Ari Kulmala, and Timo D. Hämäläinen
Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland
email:erno.salminen@tut.fi

Abstract—This paper presents the state-of-the-art in the field of network-on-chip (NoC) benchmarking and comparison. The study identifies the mainstream approaches, how NoCs are currently evaluated, and shows which aspects have been covered and those needing more research effort. No single article can cover all the aspects, and therefore, possibility to compare results from various sources must be ensured by proper scientific reporting. Basic guidelines for achieving that are given.

Keywords: network-on-chip, literature study, comparison, benchmarking, guidelines

I. INTRODUCTION

The future System-on-Chip (SoC) architectures are predicted to become communication-bound. Traditionally, System-on-Chips (SoCs) utilize topologies based on shared buses. Dally and Towles proposed replacing dedicated, design specific wires with general purpose, (packet-switched) network [18], hence marking the beginning of network-on-chip (NoC) era.

The vast number of NoC parameters is an inherent problem in design and comparison, for example in defining the topology, switching and flow control policy, and routing algorithm. The design space, that is the number of possible combinations for parameter values, is way too large to allow a complete, brute-force exploration. Hence, advanced design methods and heuristics are needed to obtain a functional, preferably optimized system in finite time.

No optimal NoC exists in general case. However, benchmarking allows identifying the parameters that are most significant for certain application scenarios. This reduces the design space once the major characteristics of the system and its requirements are known. Naturally, the benchmark results must be validated by other researchers by repeating the experiments. This necessitates that the benchmarking is done with care and by following strict scientific principles both in measurements and in reporting.

This paper presents the state-of-the-art in the field of network-on-chip benchmarking and comparison. To our knowledge, this paper is one of the first concentrating purely on the how NoCs are and should be compared. A vast set of studies are gathered from literature and analyzed. A deliberate choice was made not to point out mistakes in specific papers but to describe the potential pitfalls in general terms and without pointing to the sources. Several guidelines are given to alleviate the current shortcomings. The guidelines give direct practical benefit for NoC community as they enhance the reproducibility and credibility of the results.

II. GOALS OF THE NETWORK-ON-CHIP PARADIGM

The early work and basic principles of NoC paradigm were outlined in various seminal articles, for example [61][7][29][60][25][22][41][42][8][11]. The basic properties of the NoC paradigm are

- separates communication from computation
- avoids global, centralized controller for communication
- allows arbitrary number of terminals
- has a topology that allows the addition of links as the system size grows (offers scalability)
- does not utilize long, global wires spanning the whole chip¹
- customization (link width, buffer sizes, even topology)
- allow multiple voltage and frequency domains
- delivers data in-order either naturally or via layered protocol
- offers varying guarantees for transfers
- offers support for system testing

However, there is no commonly-agreed definition what is the minimum network configuration that is still a real “NoC”. Some authors consider packet-switching as a key property of a NoC. There are, however, quite a few circuit-switched approaches dubbed as NoC as well. Naturally, certain properties, such as bit error rate, energy, and area, have to be minimized as noted by several authors. However, the opposite goal is not meaningful anyway. It must be possible to measure the desired properties, either numerically or with Boolean values (true/false). For example, if scalability, flexibility, or simplicity is taken as requirement, some exact criteria or measuring unit must also be explicitly defined. Consequently, the authors of this paper adopt a simple and neutral, although loose, definition that “*network-on-chip is a communication network targeted for on chip*”.

However, the fine goals above alone help only little in selecting the most appropriate solution among all the NoCs. Especially so since no NoC is good by definition. Hence, various approaches must be compared and contrasted fairly and extensively.

III. NOC COMPARISONS IN LITERATURE

It is necessary to quantitatively measure the various NoC proposals. Table I lists network comparisons found in literature: [54][39][46][24][51][55][59][57][58][63][2][31][32][43][15]

¹This restricts the utilization of single-hop topologies, such as single bus, crossbar or point-to-point although they all are NoCs.

TABLE I
SUMMARY OF COMPARATIVE NOC STUDIES.

Author	Ref.	Topology and size										Evaluation type					Evaluation criteria								
		single bus	hier. bus	crossbar	mesh/torus	tree/fat-tree	ring	other	# networks	# system sizes	size ratio	analytical	statistical	tg	tx-dep.	application	#cases	Repeatable cases	runtime	area	power/energy	latency	throughput	frequency	other
Salminen, T.	[54]				m			1	1	1			4		4	(x)					x			x	2
Moraes	[39]				m			1	2	6			3		3	x		x	x					3	
Penolazzi	[46]				m			1	5	4			1		1	x			x					1	
Hu	[24]	b						2	1	1			30	1	31	-				x				x	2
Saastam.	[51]					tr	r	2	1	1			1		1	x		x	x					x	3
Thid	[57]	b			m			2	1	1			1		1	x		x		x				x	3
Thid	[58]	b			m			2	1	1			2		2	x		x							1
Xu	[63]	b		x				2	1	1					1	1	-	x	x	x			x	x	5
Andriahant.	[2]	b				f		2	4	8			1	1	2	x		x			x				2
Lahtinen	[31]	b		x				2	4	4		x			1	x		x			x	x	(x)		4
Lee, H.G.	[32]			x				2	4	2			1		1	x		x	x	x				x	4
Ogras	[43]				m, em			2	5	6			3		3	(x)		x	x	x					3
Charlery	[15]	b				f		2	6	6			1		1	(x)		x					x		2
Lu, R.	[37]	b	sb					2	8	8			2		2	x					x	x			2
Arteris	[1]		hb					2	$f()$	$f()$		x			1	x			x	(x)	x	x	x		5
Zeferino	[66]	b			m			2	$f()$	$f()$		x			1	x		x	x					(x)	3
Zeferino	[65]	b				f		2	$f()$	$f()$		x			1	x		x	x					(x)	3
Chang	[14]		sb		m			3	1	1			3?		3	x		x	x	x	x				4
Dumitrascu	[20]	b					er dms	3	1	1					1	1	-	x			x	x			3
Hilton	[23]	b						3	1	1					1	1	x	x	x					x	3
Kreutz	[27]				m,t	f		3	1	1		x			3	4	-			x	x				2
Lahiri	[30]	b	hb				r	3	1	1			3		3	x							x		1
Liljeberg	[35]	b	sb				r	3	1	1			3		3	(x)							x		1
Lu, Z.	[38]				m,t			3	1	1		x	1		2	x					x	x		x	3
Pimentel	[47]	b		x			omega	3	1	1			1		1	-		x						x	2
Shen	[55]	b	sb		m			3	1	1			2		2	-		x	x	x					3
Wolkotte	[62]				m			3	1	1			4		4	x		x	x				x		3
Cardoso	[13]				m			3	3	4					2	2	-		x		x				2
Liang	[34]	b	hb		m			3	5	5					5	5	-	x					x		2
Vassiliadis	[59]				m	f, tr		3	18	2 ¹⁷					1	1	-	x	x					x	3
Salminen, E.	[53]	b	(hb)				ideal	3	36	36		(x)	1		1	-		x					(x)		2
Zhang	[67]		mb		m		hier m.	3	49	49					3	3	-			x					1
Angiolini	[4]	b	mb		m			4	1	1			2		2	(x)		x	x	x	x	x	x		6
Kreutz	[28]	b			m	tr		4	1	1			3		3	-		x							1
Richardson	[49]	b			m			4	4	16			2		2	(x)						x			1
Ryu	[50]	b	hb				(c)	4	4	24					3	3	-	x	x				x		3
Bolotin	[12]	b	hb		m		p	4	$f()$	$f()$		x	3		4	x		x	x					x	3
Bartic	[6]			x	m,t	tr		5	1	1		x			1	x		x				x	x		3
Pande	[45]				m,t	f	er	5	1	1			3		3	x		x	x	x	x			x	5
Bertozi	[9]				m,t	f	c	5	3	3			4		2	6	x		x	x	x			x	4
Loghi	[36]	b	mb	x				5	4	3					4	4	-	x					x		3
Salminen, E.	[52]	b	hb		m			6	4	16			1	5	6	x		x	x					x	4
Bartic	[5]			x	m,t	tr		7	1	1					1	1	-	x	x						2
Lee, K.	[33]		mb	x	m		p	9	7	6			2		2	(x)			x	x					2
Count	44	25	15	8	26	11	5	12	44	44	44	9	18	12	13	44	29	25	24	17	17	13	11	13	44
%	-	57	34	18	59	25	11	27	-	-	-	20	41	27	30	-	66	57	55	39	39	30	25	30	-
Avg	-	-	-	-	-	-	-	-	2.9	5.1	5.9	-	2.3	4.5	2.2	3.0	-	-	-	-	-	-	-	-	2.8

Symbols for topologies:

x crossbar
b, mb (single shared) bus, multibus
hb, sb hierarchical bus, split bus
m, em mesh, extended mesh
t torus
f, tr fat-tree, tree
r, er ring, extended ring
c custom
p point-to-point
h hypercube
dms distributed shared mem.

[37][1][66][65][14][20][23][27][30][35][38][47][62][13][34][67][4][28][49][50][12][6][9][36][45][52][53][5][33]. Various comparisons of networks in general, not just NoCs, can also be found in [17]. The results are divided into three sections: *Topology and size*, *evaluation type*, and *criteria*. The studies are sorted according to the number of networks, system sizes and then alphabetically. The bottom rows show the absolute and relative occurrences. For example, single bus is included in 25 studies (57% of the cases). Average values are shown for numerical attributes.

A. Compared topologies

The evaluated topologies and number of different system sizes are shown in the column *Topology and size*. Single bus is separated from bus topologies with multiple links (hierarchical, split, and multibus). Single bus and mesh/torus are the most studied topologies (over 50% of the cases). They are followed by hierarchical buses, fat-trees, and rings. One fourth of the studies consider also other topologies, such as point-to-point or custom. A special case is [53] where a single bus is compared to “ideal” network.

Column *#networks* shows how many NoCs are compared. The value can be larger than the sum of the listed topologies because different versions of the same topology are counted separately. For example, wormhole and store-and-forward mesh are considered as two networks. On average, 3 networks are compared and the largest study includes 9 [33].

Half of the cases consider only one system size (number of terminals) and about 5 sizes are included on average. The most sizes are covered in [59](18 system sizes), [53](36 sizes), and [67](49 sizes). Analytical models are shown with $f()$ and they offer naturally very large range of systems. Large networks have been studied in simulation also; up to 100 nodes in [13][33][43] and 256 nodes in [45].

The *size ratio* is the largest system size divided by the smallest and rounded to nearest integer. It gives coarse idea of how large design space is covered. It is 5.5 on average and 11 when counting only the studies with multiple sizes. Varying the system size over $10x$ covers wide spectrum and hence gives good justification for presented claims and observations. An exceptional and clearly the largest system range, $2^3 - 2^{20}$ nodes, is covered in [59] and it is not included in the above average values.

B. Evaluation type

The basic methods for performance comparison are *analysis* and *simulation* with various test case types. Area and frequency estimates are obtained from synthesis. However, prototyping or emulation are not widely utilized. Few performance results are obtained from FPGA [23][32] [43] and from reconfigurable Maia chip [67]. Note that FPGA synthesis only is not quite the same as running full applications on the FPGA (which naturally includes synthesis).

Several cases are needed to make general conclusions. The listed test case counts for the synthetic cases are merely suggestive since it is not clear when the change in some parameter

actually defines a new test case. Most studies use only limited number traffic scenarios, 3 on average. Different scenarios are often modeled with some sort of traffic generator (*tg*), either omitting (usual case) or considering the dependencies between tasks. Real applications are used in one third of the studies. Sometimes the traffic generators are used to mimic the real applications but without performing actual computation. The largest synthetic test set, 30 random task graphs, is in [24] whereas application sets with 4 [36] and 5 cases [34] have been reported. The utilized test cases are discussed later in more detail.

The last column in this part denotes how well the test cases are documented. The test cases that can be repeated based on the publication are marked with x . Notation (x) is used when some information is missing but rough reproduction is possible. The analytical studies are reproducible by nature. The synthetic cases without dependencies can be defined with few parameters and are therefore easiest simulation cases to reproduce. Using dependencies and applications necessitates documenting also the mapping information. Applications are generally too complex to describe briefly.

C. Evaluation criteria

On average, the studies in Table I use 3 different metrics for comparing NoCs. The most popular and important metrics are application *runtime*, *silicon area*, *power* consumption, and *latency*. All these are to be minimized and usually appropriate trade-off is sought. There are few papers where the marked metrics are measured only for a fraction of studied networks or system sizes.

Short runtime is the most straightforward indication of high performance experienced by the user. The runtime is hard to measure without real applications or transfer-dependent traffic models. Latency can be measured in all cases and it affects the runtime. However, the relation is not straightforward, for example, various latency hiding techniques can remove the impact of latency as long as it is below certain threshold. Therefore, smaller latency does not necessarily translate to smaller runtime; see for example [20]. However, shorter latency will not increase the runtime either, when the possible scheduling anomalies are neglected. Therefore, achieving the same latency with fewer resources (area, power) is a viable target for optimization. Similarly, the bandwidth of the NoC is only an indirect clue of performance [4].

There are various other metrics that are important in understanding and optimizing the system, for example congestion, path diversity, error tolerance, and operating frequency. However, their impact should be reflected in the four “major” metrics.

IV. OVERVIEW OF THE REPORTED FINDINGS

Some results from the literature may sound contradicting at first but it must be noted that the performance depends on the application. The various studies have simply utilized test cases with different characteristics and requirements.

TABLE II
REPORTED FINDINGS FROM NOC STUDIES

NoC		Runtime ratio	Other ratio	Ref.
#1	#2	#1/#2	#1/#2	
bus	mesh	1x	$N^{2.5}$ area, $N^{0.5}$ power	[12]
		0.1-0.5x	-	[28]
		1.1 - 3x	-	[34]
		1.2x	1.4x energy	[55]
		2x	0.8x area	[23]
		0.3-0.9x, 3-6x 1-50x (11x avg)	- 0.3x area	[58] [52]
bus	hier.bus	1x	N_x area, 1x power	[12]
		0.9x	-	[34]
		1.4-2.4x	0.3-1.3x area	[50]
		1-30x (10x avg)	0.7x area	[52]
bus	crossbar	-	0.3-0.6x TP	[30]
		1.5x	-	[47]
		1.6x	1x area	[63]
		1.5-2.5x 1.1-4x	- 0.5x area	[36] [31]
bus	ring	-	1.5-3x TP	[30]
		-	1.3-1.9x TP	[35]
bus	fat-tree	-	0.2x sat.point	[2]
	multibus	4-7x	-	[4]
	split bus	-	0.2x BW, 5.5x lat.	[37]
	octagon	1x	3x lat.	[20]
	p2p	-	4x energy	[24]
	tree	0.3x	-	[28]
mesh	ext.mesh	1.5x	0.9x area, 1.5x energy	[43]
	hier.bus	0.2-0.9x	-	[34]
	hier.mesh	-	1.3-1.9x energy	[67]
	p2p	1x	0.6x power	[32]
mesh	crossbar	-	-	[28]
		-	0.8x area/BW 1x area, 2x energy	[6] [33]
mesh	multilayer-bus	0.9x	1.9x area (NoC)	[4]
		0.9x	1.3x area (chip)	[4]
		-	0.1-0.4x energy	[67]
		-	1.5-3x area	[33]
mesh	fat-tree	4-16x	-	[59]
		-	0.9x lat., 1.2x energy	[27]
		-	1x area/BW	[6]
hier.bus	ring	-	1x TP	[30]
		-	1x TP	[35]
crossbar	fat-tree	-	1.3x area/BW	[6]
	tree	0.5-1.8x	1x area	[59]

N =number of NoC terminals, used for asymptotic cost functions

BW =bandwidth

TP =throughput

There are large differences between various configurations of the same topology. Circuit-switched mesh outperforms packet-switched in [34](1.1 – 2x speedup), [14] (13x energy reduction, 6x speedup). On contrary, packet-switching offers 2.5 – 3.5x speedup in [28]. For buses, different versions may have 1.4 – 3x and 9.5x difference in runtime [52][23].

Comparison results from literature are summarized in Table II with emphasis on runtime comparisons. The compared topologies are given on the two left-most columns, followed by runtimes ratio when available. When ratio is < 1, NoC_1 is faster and vice versa. For example, a runtime ratio 2x means that NoC_2 offers *two-fold speedup* because NoC_1 takes twice the runtime compared to NoC_2 . The last columns show the ratio of other studied metrics and reference to the source.

Mesh and hierarchical bus (hier.bus) achieve shorter runtime than single bus in most cases. However, both single and hier.bus require smaller area than mesh. Crossbar is also superior to bus but suffers from unacceptable area in large systems. Hier.bus have quite similar performance with ring in [30][35] and mesh in [52]. On the other hand, 2-D mesh is outperformed by hierarchical mesh [67], extended mesh [43], custom [9], and fat-tree [59]. In few cases, a single bus is actually faster than hierarchical bus, 0.9x runtime [34], or mesh 0.1 – 0.5x [28] and 0.3 – 0.9x runtime [58].

The simulated speedups are clearly smaller than expected by analytical studies or by those concentrating purely on load-latency curves. However, the speedup depends on how runtime is measured; in clock cycles or in seconds. Using cycle counts favors buses and crossbars that are likely to have limited operating frequency. Fat-tree also exhibits few links near the root whose length and delay depend on the system size.

It should be noted that a network with increased parallelism does not translate directly to shorter runtime. Octagon network outperforms single bus clearly in terms of latency [26] when synthetic traffic is utilized. For video encoding [20] bus and octagon obtain identical runtime despite the fact that bus exhibits larger and more varying latency.

As another example, 5x5 crossbar enables 5 parallel transmitters and gives 50% increase in application performance compared to single bus [47]. This study emphasizes an important point: *the increased parallelism in the network allows improvements via allocating more resources* (memory banks in this case). Parallel memories are unlikely to produce any speedup with the bus. Similarly, the crossbar is unlikely to improve the performance when utilized with one centralized memory. Hence, the resource allocation and optimal network are tightly coupled to each other.

A. The reported applications

The utilized applications vary from simple kernels (FFT, IIR, sort, image binarization, vector sum) to full applications (mostly video encoding). The most common workload is uniform random traffic that is used to determine the load-latency curves for the networks. Destinations have uniform distribution which means that each source sends data to all others with equal probability. Sources generate data based on

their assigned load, for example 5% of the time. This type of traffic belongs to category *statistical tg* in Table I and it can be parameterized by changing the offered load per terminal and temporal behavior (burstiness, self-similarity). Uniform distribution is unlikely in real application and therefore localized target distributions must be evaluated also [45]. Sometimes all these parameters are not reported.

It is easy to cover wide design space by modifying such parameters. However, the researchers should also consider which parameter values are actually present in real applications currently and they are estimated to evolve. This is important research problem and definitely needs more attention. Some

work has been carried out in multiprocessor community, see for example [16], but their application domain differs from SoC. Traffic characteristics and modeling in shared-memory cache-coherent multiprocessor chips have been studied in [56].

B. Network interface

In addition to router, a network interface (NI) (also called network adapter) is usually needed to handle the packetization, packet re-ordering and for controlling the retransmissions. It offers high-level services by abstracting the underlying network to an interface. The impact of NI is too often neglected. We discuss briefly the basic considerations.

The interfacing usually needs some hardware, for example to connect processor's memory bus to the network, but some parts, for example retransmission or reordering, may be performed in software. Hardwired accelerators (especially third-party components) and memories require that interfacing is done with special purpose hardware. Software approach is slower and requires program memory [10] but may be easier to design and implement in the start of the development. Interfaces and other network components should be reused in several environments to amortize the development and verification cost, and hence, modularity and scalability are desired properties [48].

Adaptive routing may deliver packets out-of-order. The network interface is responsible for reordering if processing element (PE) requires that. Reordering typically needs large buffers and the receiver must send acknowledge at known intervals, for example after 8 packets, to avoid infinite buffers. However, buffers are still required for each active source. The buffer size per source is a product of interval between acknowledges and the maximum packet size. In the worst case, all other $N - 1$ agents may be sending to one target which means that total buffer size grows with $(N - 1)^2$. This can be reduced if sources first request a permission to send (i.e. a buffer at the receiver) but this induces a round-trip latency prior any data transfer and higher network load. Fixed size packets and locating the reordering buffers into the local data memory may simplify the procedure.

V. PRACTICAL BASIC GUIDELINES FOR SIMULATION AND BENCHMARKING

Comparison is a difficult task and requires consistent and generally accepted guidelines. It is clear that no single article can cover all aspects due to myriad of affecting parameters. Therefore, strict evaluation procedures must be followed throughout the research, not just in NoC design but especially in reporting phase. This ensures meaningful comparison between various sources and the overall view can be determined in incremental steps.

The issue of evaluation pitfalls has been addressed in two excellent articles. Anel and Yasinsac [3] analyze the credibility of mobile ad-hoc network (manet) simulations whereas Frachtenberg and Feitelson [21] concentrate on the evaluation of job scheduling policies. To alleviate shortcomings, both articles also provide few guidelines and it is interesting to note

TABLE III
SIMULATION GUIDELINES FOR NOC EVALUATION. PARTLY BASED ON [3] AND [21].

	#	Guidelines
Workload	1.1	Use several cases, ensure that they are representative
	1.2	Validate statistical workload models against real applications
	1.3	Ensure that cases are different enough and not biased towards some property
	1.4	Document properly if there are limiting assumptions about the usage scenario
	1.5	Include variable bit rate (self-throttling) and bursty (self-similar) traffic
	1.6	Scale the load with great care to avoid distortion of important properties
	1.7	Use documented, freely available benchmark workloads when available
System model	2.1	Evaluate many system sizes and configurations
	2.2	Validate models against data from external source (other publications etc.)
	2.3	Validate the analytical and simulation models against real implementations
	2.5	Include various overheads (e.g. synchronization or interrupt latency)
Measurements	3.1	Document all settings and assumptions to enable repeatability
	3.2	Determine the number of required independent runs to gain statistical validity
	3.3	Address the sources of randomness to ensure simulation run independence
	3.4	Perform sensitivity analysis to identify the significance of a certain parameter
	3.5	Carefully select and document the used metrics. Use several metrics.
	3.6	Specify units and ratios explicitly. Use (de-facto) standard units when possible.
	3.7	Discard warm-up period and saturated workloads from the results
	3.8	Use long workloads and (simulation) runs
	3.9	Use "infinite" source queue in load-latency measurements
Conclusions	4.1	Account for estimation/simulation errors in comparison
	4.2	Compare and contrast the results to the state-of-the-art
	4.3	Repeat the evaluations on a reference system
	4.4	Search for trends and correlation in the obtained results
	4.5	Do not use speedup from a small system directly to estimate the bigger ones

how well they also apply for the NoC evaluation. Table III lists those guidelines and the proposed additions.

The guidelines given here might seem obvious and self-evident at the first look. However, they are not; judging by [3][21] and the articles cited here. Some of them are usually considered but a reader cannot be sure unless the matter is explicitly stated, which is rare. First step to improve the situation is the brief checklist given in Table III. Second, the research community should evaluate the guidelines, and then thrive for their wide adoption. For example, a newly formed NoC benchmark workgroup [40] aims to provide the academic and industrial R&D communities with a set of characteristic benchmark designs and guidelines that will serve as a common repository of relevant information with

the following objectives:

- Enable the sharing and comparison of NoC-related R&D efforts and findings
- Enable and accelerate the NoC paradigm development
- Increase the reproducibility of R&D claims and results
- Bridge the gap between academic and industrial state of the art.

The proposed guidelines are divided into 4 parts: *workload*, *system modeling*, *measurement*, and *conclusions*. Each part is discussed separately in the following.

A. Workload

Workload refers to applications or their models that are used during the evaluation. Several cases are needed to obtain overall view on the behavior of a system and they must be representative for target domain, for example multiprocessor SoC targeted for mobile devices. In addition to their number, the cases must present different characteristics to justify their inclusion in the test set. If the cases are biased, for example assuming very small transfer sizes, that must be explicitly documented and motivated. In the NoC domain, the traffic cases may differ, for example, in their offered load, locality, burstiness, and latency tolerance.

Applications can be modeled by injecting synthetic traffic to the network with traffic generators. Compared to real applications, this method offers speedup for simulation and easier modification. Hence, they are suitable for covering large application space with proper scaling. However, care must be taken when modifying the basic parameters not to distort the important properties which would give misleading results [21]. The models must be validated against real applications to identify the reasonable values (e.g. offered load and spatial distribution). Common benchmarks offer partial solution to the problem as they greatly simplify documenting the workload.

B. System model

The utilized modeling technique must support various system sizes and configurations to cover reasonable design space. It is practically impossible to include all the minor details to simulation and analysis which evidently leads to estimation errors. Therefore, the complete simulation (developed protocol, traffic, environment model, and usage scenario) should be validated against a real-world implementation, analytical models, or protocol specifications. The latter is viable during early concept development but less precise, but it can be further refined later [3]. This applies to models of the NoC (transaction-level, cycle-accurate models, area and wire estimates) as well as the environment (traffic generation, third-party IP). Comparing just the analytical and in-house developed simulation models is somewhat dangerous as they both might have the same erroneous assumptions [21]. Therefore, validation must be performed against external, independent data, when possible.

If the models turn out to be inaccurate, they must be tuned and usually that means adding more details. In MP-SoC, such details include, for example, interrupt latency at the receiver,

data synchronization at the clock boundary, context switch overhead, DMA transfers, crosstalk, and chip layout. Estimates based on early models need revision during the course of research. For example, the area overhead of the NoC was first estimated to be 1 – 2% [44] but after more research the estimates were updated to range 9 – 45% [45]. The analytical studies in Table I are well motivated and seem intuitively correct. However, no results were found on the validation of the models.

C. Measurement

The results are affected by numerous parameters and assumptions related to test application, the NoC itself but also to simulation and synthesis environments. Large design space necessitates automatic exploration and optimization of mapping, scheduling, buffer, topology, and various other aspects. Complete coverage is practically always out of the question, though. Other researchers should be able to repeat the experiments and hence justify the findings. This requires very detailed description and reporting. Unfortunately, the settings and assumptions are cumbersome to define and list, especially as the available space is limited in the publication venue. Such data sheets could be published, for example, in research group's web pages so that they are freely accessible.

The environment models and synthesis tools commonly apply some pseudo-random techniques and hence several independent runs ensure statistical reliability. Independence can be achieved, for example, by explicitly varying the seed value for random number generator. Sensitivity analysis identifies the most important parameters; those that produce the greatest variation in results. Proper selection of values for these parameters is a critical step at benchmarking and requires thorough motivation and background work.

Some phenomena appear only with long enough simulation or execution run. The warmup period at the beginning must be discarded from the results to measure steady-state behavior. For example, the first few packets experience zero-load conditions because NoC congestion arises only after several packets have been injected. Therefore, inclusion of first packets distorts the average values. Large (infinite, in theory) buffers must be placed between traffic source and network to avoid self-throttling during load-latency measurements [17]. Without such buffers, generators stop the data injection every now and then and the real offered load does not match the expected. Once a NoC gets saturated, the source queues start growing infinitely. The maximum observed latency is not infinite but defined by the length of the measurement. The results must, however, assume infinite latency during saturation.

The *performance* and *cost* are hardly universal metrics with well-defined semantics and units. There is a clear distinction that the performance is to be maximized and the cost minimized. The *overall performance* or *merit* is a combination of several factors, such as runtime, latency, throughput, utilization, error rate, power consumption, and area [6][52]. They can be used to determine a Pareto-front where each point along that curve denotes one Pareto-optimal solution.

Alternatively, a performance function (inverse of cost function) can obtain a single value for comparison. It is defined case by case as a combination of available metrics, for example $cost = runtime \cdot area \cdot power$. Weighing exponents help to balance the importance of the parameters. A special case is *performance with given constraint* which seems more natural for embedded systems than raw, maximum performance. Given a strict upper (lower) bound, find the solution that meets that while minimizing other factors. For example, find the NoC with smallest area (or power) while achieving at least 500 MB/s throughput for test case X.

All the metrics and units must be defined explicitly. *Latency* is a common metric for networks but its definition is ambiguous. It can be measured per data word, header, packet, or transfer (several packets) while including/excluding the time at the source queue. Furthermore, minimum, average, or maximum value may be reported. These basics are quite often left unspecified. It does not destroy the validity of the comparisons *within* the paper as long as all cases use the same (unspecified) definition. However, comparison between publications becomes impossible.

Offered load means the traffic injected to the network. It is closely related to the latency and suffers from similar ambiguities. For example, it is not always evident whether the load is given for the terminal or the whole network. Likewise, it may be given as *packets/cycle/terminal* but the packet size is not known. Both temporal and spatial distributions and the definition of load (inclusion of headers or payload only) must be explicitly given. The unit *Bytes/s/whole_net* allows unequal load for separate terminals. Packet size has profound impact on NoC performance and must be documented. Preferably, several packet sizes must be evaluated, as in [5][64].

D. Concluding the findings

ASICs are expensive and therefore usually unsuitable for prototyping in most cases. Modern FPGA devices alleviate the prototyping since they are large enough for multiprocessor SoC implementation. They are still rarely used and hence the comparisons are mostly performed via simulation or analytically. However, the comparison must take estimation errors into account. For example, any runtime difference less than estimation error, say 10%, is negligible. The fundamental problem is how to determine estimation error beforehand. The only option seems to be using models that have been validated and verified earlier and assume that the accuracy remains static. In addition to min/avg/max values, the results can make use of confidence intervals [58]. For example, the latency of 8-word packets is less than *30 cycles* in 95% and *40 cycles* 99% of the cases.

An important, although usually difficult, issue is to compare the obtained results to the state-of-the-art in the field. Rough comparison ensures that the presented results are reasonable. For example, assume that the new proposal offers 10% reduction in latency compared to the previous version by the same authors. However, the novelty is questionable if the

optimized version is clearly slower than known approaches from literature.

Direct comparison, for example "*latency is 5 cycles lower than in [ref. X]*", is more difficult but also more beneficial. However, care must be taken to use exactly the same input values and key parameters. Therefore, common test cases and reference implementations are of great value. One option is to re-implement novel ideas from the literature such as routing or coding schemes. First, this either ensures the validity of the approach or reveals deficiencies that were not covered in the original article. Second, this allows just comparison, as many parameters, such as processing technology, can be held constant. Currently, most NoCs and their test cases are proprietary and unavailable for public evaluation. A freely available NoC, such as [19], can be used a common reference for implementation results.

It is very beneficial to identify systematic trends from the results. Especially when the source of the phenomena is analyzed. It is crucial to avoid generalizations without large coverage of parameter space. For example, deriving the speedup from a small system and extrapolating it to larger, is a common source of errors [21].

VI. CONCLUSIONS

This paper gives an overview of state-of-the-art network-on-chips. An extensive set of examples is collected from literature concentrating on the comparative studies. Currently, the most studied topologies are bus and mesh. They are compared, on average, in 5 system sizes and by using 3 traffic scenarios. The most common metrics are area, runtime, latency and power consumption. They are usually obtained via simulation and synthesis. The simulation evidently leads to some simplifications and inaccurate estimates. Therefore, more results from real prototypes are desired.

Currently, there are many details missing from the publications. The omission is very likely innocuous, but nevertheless the other investigators cannot repeat the experiments without these information. Practical guidelines were given to further enhance the NoC research.

REFERENCES

- [1] "A comparison of network-on-chip and busses," Arteris, white paper, 2005.
- [2] A. Adriahtenaina *et al.*, "SPIN: a scalable, packet switched, on-chip micro-network," in *DATE*, Mar. 2003, pp. 70–73.
- [3] T. Andel and A. Yasinsac, "On the credibility of manet simulations," *IEEE Computer*, vol. 39, no. 7, pp. 48–54, Jul. 2006.
- [4] F. Angiolini *et al.*, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *DATE*, Mar. 2006, pp. 1–6.
- [5] T. Bartic *et al.*, "Network-on-chip for reconfigurable systems: From high-level design down to implementation," in *FPL*, 2004, pp. 637–647.
- [6] —, "Topology adaptive network-on-chip design and implementation," *IEEE Proc. Comput. Digit. Tech.*, vol. 152, no. 4, pp. 467–472, Jul. 2005.
- [7] L. Benini and G. de Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.
- [8] —, *Networks on chips: technology and tools*. Morgan Kaufmann, 2006.
- [9] D. Bertozzi *et al.*, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

- [10] P. Bhojwani and R. Mahapatra, "Interfacing cores with on-chip packet-switched networks," in *VLSI design*, Jan. 2003, pp. 382–387.
- [11] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, p. article No. 1, 2006.
- [12] E. Bolotin *et al.*, "Cost considerations in network on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 19–42, Oct. 2004.
- [13] R. Cardoso *et al.*, "Design space exploration on heterogeneous network-on-chip," in *ISCAS*, May 2005, pp. 428–431.
- [14] K.-C. Chang, J.-S. Shen, and T.-F. Chen, "Evaluation and design trade-offs between circuit-switched and packet-switched NOCs for application-specific SOCs," in *DAC*, Jul. 2006, pp. 143–148.
- [15] H. Charlery and A. Greiner, "A SystemC test environment for SPIN network," in *MIXDES*, Jun. 2006, pp. 449–453.
- [16] R. Cypher, A. Ho, S. Konstantinidou, and P. Messian, "Architectural requirements of parallel scientific applications with explicit communication," in *ISCA*, 1993, pp. 2–13.
- [17] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann Publishers, 2004.
- [18] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *DAC*, 2001, pp. 684–689.
- [19] A. V. de Mello and L. H. Möller, "Hermes project web page," [Online] <http://toledo.inf.pucrs.br/gaph/Projects/Hermes/Hermes.html>, 2004.
- [20] F. Dumitrascu *et al.*, "Flexible MPSoC platform with fast interconnect exploration for optimal system performance for a specific application," in *DATE*, Mar. 2006, pp. 1–6.
- [21] E. Frachtenberg and D. G. Feitelson, "Pitfalls in parallel job scheduling evaluation," in *LNCS 3834: Job Scheduling Strategies for Parallel Processing*. Springer-Verlag, 2005, pp. 257–282.
- [22] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm," in *VLSI*, Jan. 2004, pp. 845–851.
- [23] C. Hilton and B. Nelson, "A flexible circuit switched NOC for FPGA based systems," in *FPL*, 2005, pp. 24–26.
- [24] J. Hu, Y. Deng, and R. Marculescu, "System-level point-to-point communication synthesis using floorplanning information," in *ASP-DAC/VLSI*, Jan. 2002, pp. 573–579.
- [25] A. Jantsch and H. Tenhunen, Eds., *Networks on Chip*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2003.
- [26] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *IEEE Micro*, vol. 22, no. 5, pp. 36–45, Sep.-Oct. 2002.
- [27] M. Kreutz *et al.*, "Energy and latency evaluation of NoC topologies," in *ISCAS*, vol. 6, May 2005, pp. 5866–5869.
- [28] M. E. Kreutz *et al.*, "Communication architectures for system-on-chip," in *SBCCI*, 2001, pp. 14–19.
- [29] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *VLSI*, April 2002, pp. 105–112.
- [30] K. Lahiri, A. Raghunathan, and S. Dey, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," in *Conference on VLSI design*, 2001, pp. 29–35.
- [31] V. Lahtinen *et al.*, "Comparison of synthesized bus and crossbar interconnection architectures," in *ISCAS*, vol. 5, May 2003, pp. 433–436.
- [32] H. G. Lee *et al.*, "Design space exploration and prototyping for on-chip multimedia applications," in *DAC*, Jul. 2006, pp. 137–142.
- [33] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance soc design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 2, pp. 148–160, Feb. 2006.
- [34] J. Liang *et al.*, "An architecture and compiler for scalable on-chip communication," *IEEE Trans. VLSI Syst.*, vol. 12, no. 7, pp. 711–726, 2004.
- [35] P. Liljeberg, J. Plosila, and J. Isoaho, "Self-timed ring architecture for SOC applications," in *SOCC*, Sep. 2003, pp. 359–362.
- [36] M. Loghi *et al.*, "Analyzing on-chip communication in a MPSoC environment," in *DATE*, Feb. 2004, pp. 752–757.
- [37] R. Lu and C.-K. Koh, "Samba-bus: a high performance bus architecture for system-on-chips," in *ICCAD*, Nov. 2003, pp. 8–12.
- [38] Z. Lu, M. Zhong, and A. Jantsch, "Evaluation of on-chip networks using deflection routing," in *GLSVLSI*, May 2006, pp. 296–301.
- [39] F. Moraes *et al.*, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69–93, Oct. 2004.
- [40] NoC Benchmark Workgroup, "An initiative towards open network-on-chip benchmarks," white paper, OCP-IP, [Online] <http://www.ocpip.org/socket/whitepapers/NoC-Benchmarks-WhitePaper-15.pdf>, 2007.
- [41] J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Eds., *Interconnect-Centric Design for Advanced SoC and NoC*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2004.
- [42] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in noc design: a holistic perspective," in *CODES*, 2005, pp. 69–75.
- [43] U. Ogras *et al.*, "Communication architecture optimization: making the shortest path shorter in regular networks-on-chip," in *DATE*, Mar. 2006.
- [44] P. P. Pande *et al.*, "Design of a switch for network on chip applications," in *ISCAS*, 2003, pp. 217–220.
- [45] P. Pande *et al.*, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. Computers*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.
- [46] S. Penolazzi and A. Jantsch, "A high level power model for the Nostrum NoC," in *DSD*, Aug. 2006, pp. 673–676.
- [47] A. D. Pimentel *et al.*, "Towards efficient design space exploration of heterogeneous embedded media systems," in *SAMOS*, 2002, pp. 57–63.
- [48] A. Radulescu *et al.*, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, Jan. 2005.
- [49] T. Richardson *et al.*, "A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks," in *VLSI design*, Jan. 2006.
- [50] K. K. Ryu and V. J. Mooney III, "Automated bus generation for multiprocessor SoC design," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 11, pp. 1531–1549, Nov. 2004.
- [51] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo network-on-chip," in *ISCAS*, 2003, pp. 113–116.
- [52] E. Salminen *et al.*, "Benchmarking mesh and hierarchical bus networks in system-on-chip context," *Journal of System Architectures (in press)*, vol. 53, no. 8, pp. 477–488, Aug. 2007.
- [53] E. Salminen, T. Kangas, and T. Hämäläinen, "The impact of communication on the scalability of the data-parallel video encoder on MPSoC," in *Intl. Symposium on Soc*, Nov. 2006, pp. 191–194.
- [54] T. Salminen and J.-P. Soininen, "Evaluating application mapping using network simulation," in *Intl. Symposium on Soc*, Nov. 2003, pp. 27–30.
- [55] J.-S. Shen, K.-C. Chang, and T.-F. Chen, "On a design of crossroad switches for low-power on-chip communication architectures," in *ISCAS*, May 2006.
- [56] V. Soteriou, H. Wang, and L.-S. Peh, "A statistical traffic model for on-chip interconnection networks," in *MASCOTS*, 2006, pp. 104–116.
- [57] R. Thid, M. Millberg, and A. Jantsch, "Evaluating NoC communication backbones with simulation," in *Norchip*, 2003, pp. 27–30.
- [58] R. Thid, I. Sander, and A. Jantsch, "Flexible bus and NoC performance analysis with configurable synthetic workloads," in *DSD*, 2006, pp. 681–688.
- [59] S. Vassiliadis and I. Sourdis, "Reconfigurable fabric interconnects," in *Intl. Symposium on Soc*, Nov. 2006, pp. 41–44.
- [60] P. Wielage and K. Goossens, "Networks on silicon: blessing or nightmare?" in *DSD*, Sep. 2002, pp. 196–200.
- [61] D. Wiklund and D. Liu, "Switched interconnect for system-on-a-chip designs," in *IP2000*, Oct. 2000, pp. 198–192.
- [62] P. Wolkotte *et al.*, "An energy-efficient reconfigurable circuit-switched network-on-chip," in *IPDPS*, Apr. 2005, p. 155a.
- [63] J. Xu *et al.*, "Methodology for design, modeling, and analysis of networks-on-chip," in *ISCAS*, May 2005, pp. 1778–1781.
- [64] T. Ye, L. Benini, and G. de Micheli, "Packetization and routing analysis of on-chip multiprocessor," *Journal of System Architecture*, vol. 50, pp. 81–104, Feb. 2004.
- [65] C. A. Zeferino *et al.*, "Models for communication tradeoffs on systems-on-chip," in *IP based design*, Oct. 2002, pp. 394–400.
- [66] ———, "A study on communication issues for system-on-chip," in *SBCCI*, Sep. 2002, pp. 121–126.
- [67] H. Zhang *et al.*, "Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs," in *Workshop on VLSI*, Orlando, Florida, USA, 1999, pp. 2–8.