

Benchmarking Mesh and Hierarchical Bus Networks in System-on-Chip Context

Erno Salminen¹, Tero Kangas¹, Jouni Riihimäki², Vesa Lahtinen³,
Kimmo Kuusilinna³, and Timo D. Hämäläinen¹

¹ Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland
erno.salminen@tut.fi

² Nokia Technology Platforms, P.O. Box 88, FIN-33721 Tampere, Finland

³ Nokia Research Center, P.O. Box 100, FIN-33721 Tampere, Finland

Abstract. A simulation-based comparison scheme for on-chip communication networks is presented. Performance of the network depends heavily on the application and therefore several test cases are required. In this paper, generic synthesizable 2-dimensional mesh and hierarchical bus, which is an extended version of a single bus, are benchmarked in a SoC context with five parameterizable test cases. The results show that the hierarchical bus offers a good performance and area trade-off. In the presented test cases, a 2-dimensional mesh offers a speedup of 1.1x - 3.3x over hierarchical bus, but the area overhead is of 2.3x - 3.4x, which is larger than performance improvement.

1 Introduction

The increasing complexity of digital systems has forced the adoption of modular design techniques and the re-use of pre-designed and pre-verified components in the design process of System-on-Chips (SoC). Due to the increasing number of connected components, the communication and interconnect wiring are becoming a serious problem [1][2][3]. Several architecture and circuit-level alternatives have been proposed to solve this but proposals frequently only deal with the theoretical limitations of the communication networks. However, the practical limitations and requirements for the networks are affected by system-level issues and the data transfer distributions of the targeted applications. This paper presents a study of hierarchical bus and mesh networks and examines the effect of transfer distributions in benchmarking SoCs. Furthermore, the area cost of network area is examined. The following Section briefly introduces the related work done in the field. The utilized networks are presented in Section 3 and the transfer patterns in Section 4. Section 5 examines the area costs of the implementations based on synthesis results and describes the execution time comparison. In Section 6, the conclusions of the work are given.

2 Related Work

The optimal SoC communication network (network-on-chip, NoC) has been subject to debate over the last few years. Traditionally, the SoC networks are based on circuit-

switched techniques, such as bus-based networks [4], crossbars [5], and 2-dimensional meshes [6][7]. Several packet-switched network topologies have also been proposed, such as fat trees [8], 2-dimensional meshes [9][10], and rings [11]. Networks are often compared by the theoretical maximum transfer capabilities. This is straightforward but the results seldom reflect the performance of real applications accurately, i.e. the order of performance values may be defined but not their ratio. However, building actual applications is laborious only for comparison purposes and their simulation is time-consuming. Furthermore, the execution time of an application is often dependent on input data.

Communication generators provide flexibility and accelerate simulation with the cost of reduced accuracy. They use transfer patterns, or profiles, that resemble the external behavior of real applications. In statistical generators [12][13], the communication profile is often independent of the profiles of other agents (processing elements) that are connected to the network. Statistical methods are suitable for analyzing and optimizing the network when the system architecture and application mapping are fixed. Transfer dependent methods [14][15][16], in contrast, enable architecture exploration including component allocation, application mapping, and communication scheduling. Transfer dependence refers to a situation in which an agent cannot proceed before it has received certain data from other agent(s).

Table 1. NoC comparisons in literature

Ref.	Topologies	Test cases	Criteria
[6]	mesh, (hier.) bus	5 applications	ex.time, throughput
[7]	mesh, bus	1 random, 1 statistical	offered load, blocking
[8]	fat-tree, bus	total exchange, 1 statistical	ex.time, latency, saturation
[10]	mesh	3 statistical	ex.time, latency, area
[11]	ring	1 application	ex.time, area
[12]	(hier.) bus, ring	tens of statistical	throughput
[13]	mesh	1 statistical	ex.time, utilization, power
[16]	mesh, bus, tree	3 transfer dependent	ex.time
[17]	mesh, bus	total exchange	ex.time, max freq.
[18]	(hier.) mesh, multibus	(at last) 3 applications	energy
this work	mesh, (hier.) bus	5 transfer dependent	ex.time, area

Table 1 lists some comparisons of different networks. All utilize simulation, except [17] that uses mathematical models. Three reported studies used real applications while seven used synthetic methods, only one of which included transfer dependence. Total exchange (also called pooling) means that all agents send and receive data with all other agents. A single bus is a viable solution for systems with a limited number of agents and bandwidth requirements because of its simplicity and the numerous legacy implementations [8][16][12]. Many comparisons examine single buses or multiple parallel

buses but omit the more versatile hierarchical bus structures. All listed comparisons neglect the cost of network in terms of area, except [10][11] which compare only different versions of single topology and analytical work in [17]. However, area of NoC routers varies greatly; between 4-700 kilogates [10][9], and therefore area should be included in comparison. A single test case is not enough to have reliable performance estimates, but a set of benchmarks is needed. This work utilizes synthetic, transfer-dependent test cases for fair benchmarking of hierarchical bus and 2-D mesh. Currently there are no commonly accepted benchmark sets for SoC networks, but the work presented in this paper could be proposed as part of such set.

3 Hierarchical Bus and Mesh

To allow fair comparison, bus, hierarchical bus, and 2-D mesh are compared for this study by using generic, synthesizable building blocks. All networks utilize the same agent interface, transfer data in fixed-size packets, and utilize store-and-forward routing. Data arrives always in-order.

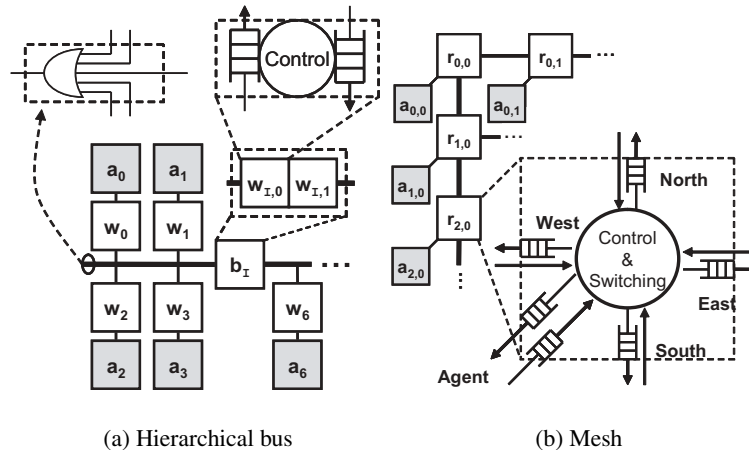


Fig. 1. Network implementations

The utilization of the single bus architecture is usually limited to systems with less than 10-20 agents due to the limited bandwidth and the problems induced by the required long signal lines. The traditional bus scheme can be extended to a hierarchical bus scheme by using bridges (marked with $b_{I,i}$) to connect several bus segments as shown in Fig 1(a). Agents (a_i) are connected to bus segments via wrappers (w_i). In this case, the number of signal lines and the operating frequency are the same in each segment. Hierarchical bus architecture used in this paper is built as a chain of bus segments, although a tree like structure could also be used. The implemented bus wrapper is a fairly simple device with two FIFO buffers and a control unit for arbitration. Arbitration is based on a distributed round-robin scheme where the ownership is passed

to the next agent after each transmitted packet. Bridge components were implemented by connecting two wrappers together. The bus signal resolution is implemented with an OR-based structure. The problems inherent in long bus signal wires are solved by grouping only a limited number of agents, in this case four, in each bus segment.

Table 2. The number of communication links in networks

Network	Number of agents				
	4	16	36	64	N
Single bus	1	1	1	1	1
Hier. bus	1	4	9	16	$N/4$
Mesh	8	48	120	224	$4(N - N^{0.5})$

A packet-switched 2-dimensional 4-way mesh used in this study is depicted in Fig 1(b). The agents are connected to an array where router elements (r_i) handle the storing and forwarding of data. Router comprises of a control unit taking care of switching and routing, and a FIFO buffer for each direction (North, East, South, West, and two for agent). Routers implement a simple dimension-order routing scheme where the transfers are first directed to the correct row and then to the requested column. The number of communication links L of networks are listed in Table 2. A single bus has only one communication link, whereas in a hierarchical bus there are as many links as there are bus segments. In mesh, there are 4 unidirectional links in each router except on those residing on edges.

4 Test Cases

The presented communication networks are compared using synthetic benchmarks that are generated to represent characteristic application properties, such as sequential/parallel behavior, communication/computation intensiveness, and spatial traffic distribution. The test cases are executed in a simulation environment called Transaction Generator (TG) [15] that is independent of the network and runs application descriptions based on the Kahn process network model [19]. Each process can be either waiting for data, reading data, processing, writing data, or finished. TG notably accelerates the simulation compared to HW/SW co-simulation with multiple instruction set simulators. At the same time, the timing error is less than 15% w.r.t. real application.

Fig 2 shows the process network graphs of the benchmarks for 8 agents. The white nodes depict computation processes that can have any arbitrary processing times of P clock cycles. The edges represent data transmissions with length of D words. Computation at a node cannot start until at least one of the arriving transfers has completed (transfer dependence). The start processes, marked with black nodes, have $P = 1$ and $D = 1$ and are executed only once to trigger computation processes. By changing P and D , the application model in TG can be made more computation or communication intensive. Both P and D can be varied randomly within a user-defined range. All these benchmarks are scaled with N so that there is one computation process and variable

number of start processes per agent. The dashed lines describe a mapping of the process graph onto eight agents. For simplicity, a 1-to-1 mapping was utilized. However, process graphs are totally independent of the hardware architecture so other mappings can be easily explored.

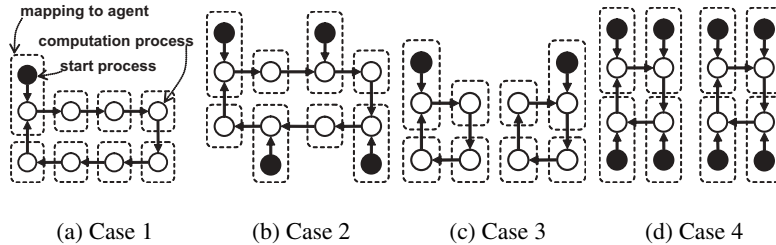


Fig. 2. Test cases used in comparison

The first benchmark, case 1, (Fig 2(a)) resembles a sequential data flow application having only one start process. Case 2 (Fig 2(b)) is partly sequential and partly parallel in nature having $N/2$ start processes. Case 3 (Fig 2(c)) presents an application where the transfers are sequential as in case 1 but in hierarchical clusters of four processes so communication is very localized. Case 4 (Fig 2(d)) has processes in a group of four transmitting data in parallel to each other. The difference to case 3 is that there are four simultaneous data transfers in each group of four agents. Cases 2 and 4 can also be thought as pipelined versions of cases 1 and 3, respectively. In addition, case 5 combines the cases 1-4 into one simulation to represent heterogeneous behavior. In case 5, cases 1-4 are run together so that each agent executes one computation process from each case. The mutual order of cases is not specified. For example, all cases have start and computation processes grouped together in the top left corner (cf. Fig 2(a)-2(d)) and they all are mapped to first agent in case 5.

The total execution time of an application is a sum of computation time and communication time. It can be estimated in a heuristic fashion for these graphs as

$$t_{\text{tot}} = \frac{\sum P_i}{\min(N, S)} + \frac{\sum (D_i * k)}{\min(N, L, S)}, \quad [\text{clock cycles}] \quad (1)$$

where

$$k = \frac{\text{payload} + \text{header} + \text{arbitration}}{\text{payload}} \quad (2)$$

is an implementation specific factor that is explained later. The divider in (1) defines the achievable parallelism. The parallelism of an application is defined here as maximum number of parallel transfers and active computation processes and it equals the number of start processes (S) in these cases. If there are less agents (N) or communication links (L) than start processes (S), the maximum parallelism of the application cannot be achieved. The maximum number of initiated transfers per clock cycle cannot exceed the number of agents in any network. For example, case 1 is a sequential application having only one start process and utilizes only one processor or communication link at

a time. Adding more communication links should not speed up the application at all. The number of start processes is shown in Table 3.

Table 3. The number of start processes in test cases

Test case	Number of agents				
	4	16	36	64	N
1	1	1	1	1	1
2	2	8	18	32	$N/2$
3	1	4	9	16	$N/4$
4	4	16	36	64	N
5	8	29	64	113	$7N/4 + 1$

The factor k , caused by arbitration and the overhead from packet headers, is calculated with (2). It is defined as the number of clock cycles needed to transfer one packet divided by the amount of transferred payload data. Ideally k would be one. Packet and header sizes are expressed as multiples of the word size, because one word can be transferred in one clock cycle. Using distributed arbitration, each agent in a bus segment has to wait a whole round-robin iteration between consecutive packets. However, there can be S agents active in each round which reduces the overall arbitration delay. The routing algorithm of a mesh checks one input each clock cycle for new transfers, thus, on average $5/2$ clock cycles are needed for routing. The term arbitration is assumed to be $(N - 1)/S$ for bus, 6 for hierachical bus, and 2.5 for mesh. Still, (1) does not take data dependencies into account and assumes uniform mapping of processes. Estimate for case 5 is a sum of individual estimates for cases 1-4.

5 Synthesis and Simulation Results

The communication networks were implemented using RTL VHDL and synthesized using a $0.18 \mu\text{m}$ CMOS technology. The packet size was set to eight 32-bit words. Since packets have a three-word header and eight-word payload data, the required buffer size is $(3 + 8) * 32$ bits in all the internal network buffers. The resulting network logic areas in kilogates are depicted in Table 4. The difference is mainly due to buffers: bus wrappers have two buffers, bus bridges four, and routers have six buffers.

Table 4. Logic areas in kilogates of network implementations

Network	Number of agents			
	4	16	36	64
Single bus	30	119	269	479
Hier. bus	30	165	390	705
Mesh	102	409	939	1635

In the following simulations, the processing time $P = 16$ and the transfer length $D = 1024$ words which sets tight requirements for the communication network. This kind of communication intensive transfer patterns can be found, for example, in packet processing inside an Internet router. The measured execution times are listed in Table 5. The poor result of the single bus in case 1 is due to inefficiency of the utilized distributed round-robin arbitration. The transfer times of the mesh and the hierarchical bus are quite close to each other due to sequential nature of the application. In test case 2, the differences between the networks are more apparent. The transfer times of the single bus grow again very fast. On the other hand, the results for hierarchical bus and mesh are quite close to each other and follow the results predicted by (1), which means that both networks are able to systematically exploit the inherent parallelism. The same happens with test cases 3 and 4. The run-time of case 5 is defined by the slowest individual application when applications 1-4 are run together. Results of case 5 with 64 agents are not available due to a limitation in the current simulation environment.

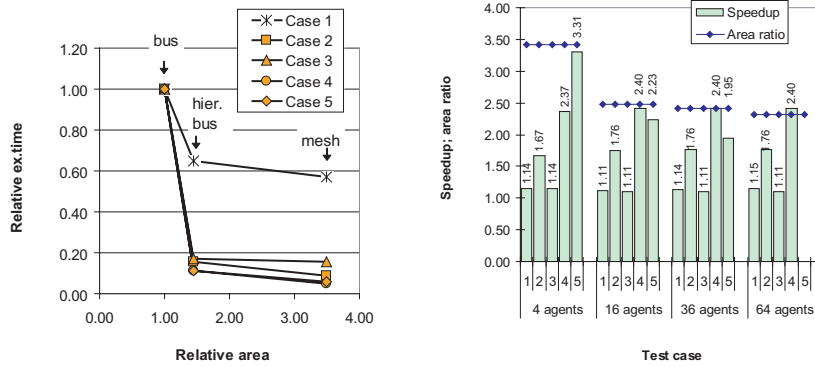
Table 5. The execution times in clock cycles for test cases

N		4 agents					16 agents				
Test case	1	2	3	4	5	1	2	3	4	5	
Single bus	17 034	9 788	17 034	9 217	76 278	68 232	38 971	43 008	36 864	933 504	
Hier. bus	17 034	9 788	17 034	9 217	76 278	69 325	13 677	16 527	9 346	222 519	
Mesh	14 913	5 858	14 913	3 886	23 040	62 240	7 793	14 913	3 886	99 860	
N		36 agents					64 agents				
Test case	1	2	3	4	5	1	2	3	4	5	
Single bus	245 556	87 535	96 768	82 944	4 171 264	663 616	233 456	172 032	147 456	x	
Hier. bus	159 865	13 727	16 527	9 346	465 375	286 620	13 782	16 527	9 346	x	
Mesh	140 040	7 808	14 913	3 886	238 878	248 950	7 823	14 913	3 886	x	

The results show that the single bus is clearly not applicable for large systems. Mesh is the fastest network in all cases but also the biggest. Fig 3(a) shows the Pareto curves for all test cases with 36 agents. Results are scaled so that both area and execution time of single bus equal one. Fig 3(b) shows the measured speedup of mesh over hierarchical bus. Speedup is defined as hierarchical bus execution time divided by mesh execution time. It also shows the area ratio, which is the area of mesh divided by the area of hierarchical bus. Mesh is faster than hierarchical bus but often the area overhead is bigger than the speedup; especially in cases 1 and 3 that do not offer much parallelism. The choice between the mesh and the hierarchical bus is, therefore, a trade-off between area cost and performance. For comparison, we define the architectural performance as the inverse of the costs:

$$Performance = cost^{-1} = (t_{tot}^{w_t} * A)^{-1} . \quad (3)$$

The cost is defined as a product of execution time t_{tot} and area A . The weighting factor w_t can be utilized to make the runtime less ($w_t < 1$) or more important ($w_t > 1$) than the area. In these cases, smaller weights favor hierarchical bus and large ones favor mesh. Fig 4 shows both estimated and measured performance of all networks so that the best case is scaled to 1 and execution and area have equal weights ($w_t = 1$).



(a) Pareto curves for 36 agents (b) Speedup and area overhead of mesh over hierarchical bus

Fig. 3. Relation of execution time and area

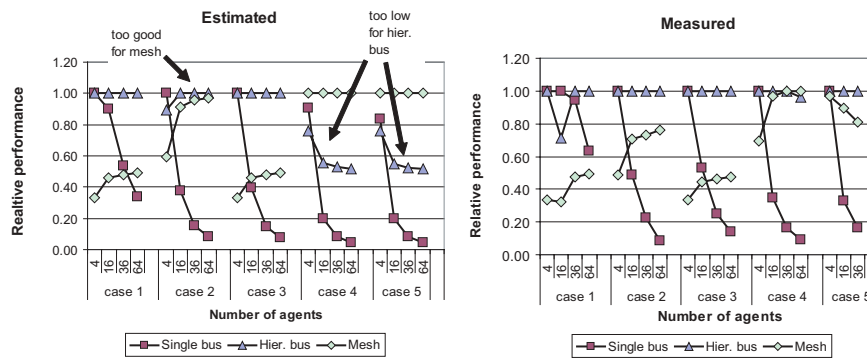


Fig. 4. Relative overall performance

Hierarchical bus offers the best trade-off for test cases 1, 2, 3, and 5. Mesh is best suited for the most parallel test case, that is test case 4 with large number of agents. A single bus is applicable only in sequential test case 1.

The shapes of the estimated and the measured performance curves match rather well. This implies that equation (1) predicts the ratio between execution times well in many cases. However, in some cases, denoted with arrows in Fig 4, the estimates are clearly inaccurate. Furthermore, the estimated times are much smaller than the measured results. This is mainly due to the implementation of TG network contention, and inefficiencies in implemented network protocols that were not included in equations. Furthermore, other than 1-to-1 mappings having more contention are likely to cause even bigger error in estimates.

6 Conclusions

This paper presented a general way for fair comparison of networks for large SoCs using synthesized HW models. Single bus, hierarchical bus, and 2-dimensional mesh networks were used in this study. Application dependence is a factor that cannot be disregarded in communication-based system design. Therefore, the analysis utilizes Transaction Generator that makes it possible to simulate the networks with different traffic patterns. Results show that theoretical performance derived from the number of links in the network does not reflect the application execution time linearly. The presented formal equation provides reasonable estimates in some case, but not in general. Moreover, more advanced estimates would require rather complex equations. Therefore, fast, cycle-accurate simulation is preferred.

Many contemporary analyzes often depict buses as poor fits to large systems because only the single bus is used as a reference. The presented hierarchical bus scales quite easily to large systems and provides a good area-performance trade-off while retaining many of the advantageous features of simpler bus arrangements. The hierarchical bus exhibits good run-time results with relatively small implementation area. The analyzed 2-dimensional mesh network provides the highest performance with the largest area. The architectural performance, defined as a product of area and execution time, favors the use of hierarchical bus. There is on-going work for developing more test cases (both synthetic and profiled real applications), exploring different process mappings, and including other network topologies. Furthermore, more performance and cost factors, such as energy and latency variation, will be analyzed. Such metrics are definitely more complex to analyze formally but can be estimated through simulation.

References

1. Benini, L., de Micheli, G.: Networks on chips: a new SoC paradigm. *Computer* **35** (2002) 70–78
2. Ho, R., Mai, K.W., Horowitz, M.A.: The future of wires. *Proc. IEEE* **89** (2001) 490–504
3. Sylvester, D., Keutzer, K.: Impact of small process geometries on microarchitectures in systems on a chip. *Proc. IEEE* **89** (2001) 467–489
4. Salminen, E., Lahtinen, V., Kuusilinna, K., Hämäläinen, T.D.: Overview of bus-based system-on-chip interconnections. In: *ISCAS*, Scottsdale, Arizona, USA (2002) 372–375
5. Lines, A.: Asynchronous interconnect for synchronous SoC design. *Micro* **24** (2004) 32–41
6. Liang, J., Laffely, A., Srinivasan, S., Tessier, R.: An architecture and compiler for scalable on-chip communication. *TVLSI* **12** (2004) 711–726
7. Wiklund, D., Sathe, S., Liu, D.: Network on chip simulations for benchmarking. In: *IWSOC*, Banff, Canada (2004) 269–274
8. Andriahantainaina, A., Charlery, H., Greiner, A., Mortiez, L., Zeferino, C.A.: SPIN: a scalable, packet switched, on-chip micro-network. In: *DATE*, Munich, Germany (2003) 70–73
9. Bartic, T.A., Mignolet, J.Y., Nollet, V., Marescaux, T., Verkest, D., Vernalde, S., Lauwereins, R.: Highly scalable network on chip for reconfigurable systems. In: *Symposium on System-on-Chip*, Tampere, Finland (2003) 79–82
10. Moraes, F., Calazans, N., Mello, A., Möller, L., Ost, L.: HERMES: an infrastructure for low area overhead packet-switched networks on chip. *Integration, the VLSI journal* **38** (2004) 69–93

11. Saastamoinen, I., Alho, M., Nurmi, J.: Buffer implementation for Proteo network-on-chip. In: ISCAS, Bangkok, Thailand (2003) 113–116
12. Lahiri, K., Raghunathan, A., Dey, S.: Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In: Conference on VLSI design, Bangalore, India (2001) 29–35
13. Thid, R., Millberg, M., Jantsch, A.: Evaluating NoC communication backbones with simulation. In: Norchip, Riga, Latvia (2003) 27–30
14. Erbas, C., Polstra, S., Pimentel, A.D.: IDF models for trace transformations: A case study in computational refinement. In: SAMOS, Samos, Greece (2003) 167–172
15. Kangas, T., Riihimäki, J., Salminen, E., Kuusilinna, K., Hämäläinen, T.D.: Using a communication generator in SoC architecture exploration. In: Symposium on System-on-Chip, Tampere, Finland (2003) 105–108
16. Kreutz, M.E., Carro, L., Zeferino, C.A., Susin, A.A.: Communication architectures for system-on-chip. In: SBCCI, Pirenopolis, Brazil (2001) 14–19
17. Zeferino, C.A., Kreutz, M.E., Carro, L., Susin, A.A.: A study on communication issues for systems-on-chip. In: SBCCI, Porto Alegre, Brazil (2002) 121–126
18. Zhang, H., Wan, M., George, V., Rabaey, J.: Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs. In: Workshop on VLSI, Orlando, Florida, USA (1999) 2–8
19. Kahn, G.: The semantics of a simple language for parallel programming. In: IFIP Conference, Stockholm, Sweden (1974) 471–475