Communication Performance of Mesh- and Ring-Based NoCs

Vaclav Dvorak
Brno University of Technology
dvorak@fit.vutbr.cz

Abstract

As multi-core systems begin to appear, their possible applications, parallel performance and on-chip interconnection networks have to be clarified, analyzed and optimized. The paper investigates an impact of collective communication (CC) overhead that may be critical for performance of parallel applications. Two potential topologies of networks on chip (NoC) are investigated, a ring-based network and 2D-mesh, due to their easy manufacturability on a chip. The wormhole switching, full duplex links and 1-port non-combining as well as combining nodes are considered. The lower bounds on the number of communication steps and upper bounds of CC times based on real CC algorithms are given. They can be evaluated for any given start-up time and link bandwidth. This enables performance prediction of applications with CCs among computing nodes.

1. Introduction

With an increasing number of processor cores, memory modules and other hardware units in the latest chips, the importance of communication among them and of related interconnection networks is steadily growing. The memory of many-core systems is physically distributed among computing nodes that communicate by sending data through a Network on Chip (NoC), [1].

Communication operations can be either point-to-point, with one source and one destination, or collective, with more than two participating processes. Collective operations are invoked by nodes to distribute, gather, and exchange data; to perform global computation operations on distributed data; and to synchronize with one another at specific points in a program flow. Some embedded parallel applications, like network or media processors, are characterized by independent data streams or by a small amount of inter-process communications [2]. However, many general-purpose parallel applications display a bulk synchronous behavior: the processing nodes access the network according to a global, structured communication pattern

The performance of these collective communications (CC for short) has a dramatic impact on the overall efficiency of parallel processing. The most efficient way to switch messages through the network connecting multiple cores makes use of wormhole (WH) routing, in which each message is divided into small pieces that are pipelined through the network. Wormhole routing reduces the effect of path length on communication time, but if multiple messages exist in the network concurrently (as is happens in CCs), contention may be a source of waiting times. Provided that computation times are known, as is usually true in case of application-specific systems, the only thing that matters in obtaining the highest performance are durations of various collective communications.

Logarithmic diameter networks, e.g. hypercube, butterfly and fat tree, provide enough bandwidth for all-to-all communications, but do not map well into two dimensions provided by a silicon chip: the length of some interconnection wires increases proportionally to the number of processors. This will decrease the clock frequency dramatically and degrade the performance. In this paper we therefore investigate two NoC topologies with only local interconnection among processors, namely the hierarchical rings and the 2D-mesh, which are under consideration for Tera-scale computing platform [3].

The paper is structured as follows. In the following Section 2 we analyze time complexity of collective communications in WH networks, namely the lower bounds on the number of start-ups for general networks (uniform messages, non-combining model). In Section 3 we investigate upper bounds on number of startups and on link occupancy for two topologies of our choice, hierarchical rings and 2D-mesh, with

message combining. Performance prediction and scalability of four typical CC is a subject of Section 4. Results and possible extensions are commented on in Conclusions.

2. Time complexity of collective communications in WH networks

A collective operation is usually defined in terms of a group of processes. The operation is executed when all processes in the group call the communication routine with matching parameters. We classify collective operations into three types according to their purpose: CCs (One-to-All, OA, All-to-One, AO, All-to-All, AA), global computation (reduction AOR or AAR and scan) and synchronization (barrier).

The CCs are most important, as other collective operations are closely related to them. In a broadcast (OAB), one process sends the same message to every group member, whereas in a scatter (OAS), one process sends a different message to each member. Gather (AOG) is the dual operation of scatter, in that one process receives a message from each group member. These basic operations can be combined to form more complex operations. In all-to-all broadcast (AAB), every process sends a message to every other group member. In complete exchange, also referred to as all-to-all scatter-gather (AAS), every group member sends a different message to every other group member. Permutation operations, such as shift and transpose, are also CCs. Since complexities of some communications are similar (AOG ~ OAS, AOR ~ OAB, AAR ~ AAB), we will focus only on 4 basic types (OAB, OAS, AAB, AAS). Also, from now on, when we refer to „collective communications", then we will assume only CCs involving the group of all processors.

Performance of CCs is closely related to their time complexity. The simplest time model of point-to-point communication in direct WH networks takes the communication time composed of a fixed start-up time t_S at the beginning (SW and HW overhead), a serialization delay – transfer time of m message units (words or bytes), and of a component that is a function of distance h (the number of channels on the route or hops a message has to do):

$$t_{WH} = t_S + m\,t_1 + h\,t_r \ , \qquad (1)$$

where t_1 is per unit-message transfer time and t_r includes a routing decision delay, switching and inter-router latency. The dependence on h is rather small, (since $t_r << m\,t_1$), so that WH switching is considered distance-insensitive. Possible synchronization overhead involved in communication steps, be it hardware or software-based, should be included in the start-up time t_S. According to frequency of CCs and an amount of interleaved computation in a certain application, efficiency of parallel processing can be estimated.

In the rest of the paper we assume that the CC in WH networks proceeds in synchronized steps. In one step of CC, a set of simultaneous packet transfers takes place along complete disjoint paths between source-destination node pairs. If the source and destination nodes are not adjacent, the messages go via some intermediate nodes, but processors in these nodes are not aware of it; the messages are routed automatically by the routers attached to processors.

Complexity of collective communication will be determined in terms of the number of communication steps or equivalently by the number of "start-ups" $\tau^{CC}(G)$ (upper bound). Neglecting the hardware overhead in routers along the traversed path (term ht_r) and excluding contention for channels, CC communication times can be obtained approximately as number of start-ups plus the sum of associated serialization delays $m_i\,t_1$,

$$t_{CC} = \sum_{i=1}^{\tau^{CC}(G)} (t_S + m_i t_1) = t_S \tau^{CC}(G) + m t_1 \sum_{i=1}^{\tau^{CC}(G)} k_i \qquad (2)$$

The above expression takes into account the fact that message length m_i can vary from one step to another, e.g. when k_i messages of length m are combined together in step i.

The port model of the system defines the number k of CPU ports that can be engaged in communication simultaneously. This means that there are $2k$ internal unidirectional (DMA) channels, k input and k output channels, connecting each local processor to its router that can transfer data simultaneously. Always $k \le d$, where d is a node degree; a one-port model (k=1) and an all-port router model (k=d) are most frequently used. Fig. 1 shows a one-port and an all-port router in a 2D-mesh. In a one-port system, a node must transmit (and/or receive) messages sequentially. Architectures with multiple ports alleviate this bottleneck. In an all-port router every external channel has a corresponding port. The port model is important in designing collective operations as it determines the number of required start-ups and thus the CC performance.

Moreover, the CC performance is influenced by the fact whether or not the nodes can combine/extract partial messages with negligible overhead (combining model) or can only re-transmit/consume original messages (non-combining model). Finally, the lower bound on number of steps $\tau_{CC}(G)$ depends on a channel type; we have to distinguish between unidirectional (simplex) channels and bi-directional (half-duplex HD, full-duplex FD) channels. Typically $\tau_{CC}(G)$ will be 2-times larger for HD channels than for the FD ones. Further on we will consider FD channels and the most frequent one-port router model only.

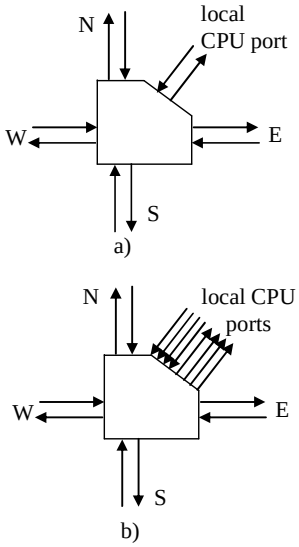

Fig.1. Port models for a 2D-mesh. a) one-port router b) all-port router

One of the key design factors of an interconnection network is its topology. The lower bounds $\tau_{CC}(G)$ for the network graph G depend on node degree d, number of nodes P, and bisection width B_C, Tab.1.

As far as the broadcast communication (OAB) is concerned, the lower bound on the number of steps $\tau_{OAB}(G) = s = \lceil \log_{k+1} P \rceil$ is given by the number of nodes informed in each step, that is initially 1, $1+1\times k$ after the first step, $(k+1)+(k+1)\times k = (k+1)^2$ after the second step, etc.,…, and $(k+1)^s \geq P$ nodes after step s.

In case of AAB communication, since each node has to accept $P-1$ distinct messages, the lower bound is $\lceil (P-1)/k \rceil$ steps. A similar bound applies to OAS communication, because each node can inject into the network not more than k messages in one step.

The lower bound for AAS can be obtained considering that one half of messages from each processor cross the bisection, whereas the other half do not. There will be altogether $\lceil 2(P/2)(P/2)/ B_C \rceil$ of such messages in both ways, where B_C is the network bisection width [4].

Table 1. Lower bounds on complexity of CCs on non-combining networks

CC	WH, k-port, full duplex links, without message combining
OAB	$\lceil \log_{k+1} P \rceil = \lceil (\log P) / \log (k+1) \rceil$
AAB	$\text{Max} (\lceil \log_{k+1} P \rceil, \lceil (P-1)/k \rceil)$
OAS	$\lceil (P-1)/k \rceil$
AAS	$\text{Max} (\lceil P^2/(2B_C) \rceil, \lceil (P-1)/k \rceil)$

For the network topologies potentially useful in a NoC the lower bounds of selected CCs are given in Tab.2. Wormhole switching and full duplex links have been assumed everywhere except unidirectional 1-way rings. Let us note that all the lower bounds can be reached for a simple ring topology and most of them also for a mesh topology by means of known algorithms [5]. Non-minimum routing would be necessary for further improvement of some upper bounds in meshes [5].

Table 2. Lower bounds $\tau_{CC}(G)$ from Tab.1 for selected networks

WH, FD, 1-port non-combining	OAB	AAB	OAS	AAS
1-way ring, P=4	2	3	3	5
2-way ring, P=4	2	3*)	3	3[+])
1-way ring, P=8	3	7	7	16
2-way ring, P=8	3	7	7	8
H. rings, P=16	4	15	15	32
H. rings, P=64	6	63	63	512
2D mesh 4 x 4	4	15	15	16
2D mesh 6 x 6	6	35	35	54
2D mesh 8 x 8	6	63	63	128

***)** three rotations
[+]) pairwise exchange – horizontal, vertical, diagonal

The bidirectional ring topology, though very simple, is not free from routing deadlock, because the channel dependency graph is not acyclic [4]. This can be seen on a common permutation called cyclic shift. The problem can be solved by introduction of virtual channels [4] and by implementing rules on channel usage. We assume that these rules are

adhered to in all our CC schedules and thus the deadlock is avoided.

As far as 2D meshes is concerned, the dimension-ordered deterministic routing (first in x, then in y direction) on meshes and tori is known to be deadlock-free. A certain degree of adaptiveness can be obtained by more relaxed routing, such as North-last or West-first strategy [4].

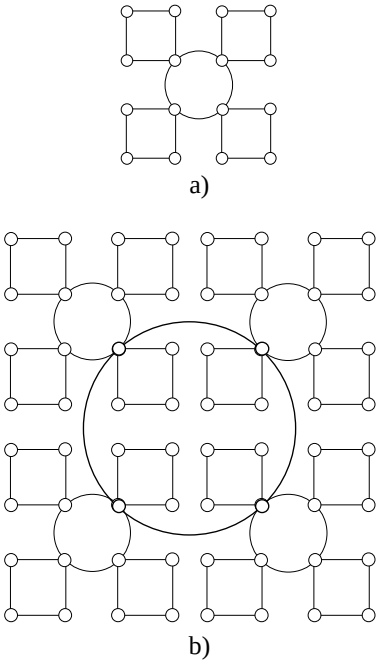

Fig.2. Hierarchical rings (H- rings)
a) P=16 b) P=64

3. Time complexity of real CC algorithms with message combining

The lower bounds on number of start-ups for uniform messages are easily translated into communication times by means of the linear communication model (2). E.g. AAS on the 8-node bidirectional ring would require 8 steps. Provided that a message has m = 100 byte, time per byte t_1 = 1ns and start-up time t_S = 10 ns (typical NoC parameters from [6]) , we get the total time

$$T_{AAS} = 8\ (10 + 100*1)\ \text{ns} = 880\ \text{ns}.$$

However, in large networks we can reduce the number of startups (further denoted by #) if we combine several messages locally and send them as a single message to a remote node representing another local group. In this case the total communication time must be calculated as

$$T_{CC} = \#\text{startups} \times t_S + \sum k_i\, m\, t_1$$
$$= \#\text{startups} \times t_S + m\, t_1\ \text{TCO}, \qquad (3)$$

where message size $k_i \times m$ is generally different in every communication step i over which we do summation. Term TCO is the Total Channel Occupancy time normalized by a transfer time mt_1 of an elementary message of size m bytes. So if we could replace the above AAS communication by another one with 3 steps and with transmission of 100, 200 and 400 bytes in successive steps, then m = 100, TCO = 1+2+4 = 7 and

$$T_{AAS} = 3*10 + (100+200+400)*1\ \text{ns} =$$
$$= 3*10 + 100*(1+2+4)*1 = 730\ \text{ns}.$$

Here the overhead of message combining is neglected as a second-order quantity.

3.1. CC algorithms on one-port combining rings and 2D-meshes

The combining model has no effect on OAB communication pattern because there are no distinct messages to be combined. The logarithmic lower bound on the number of start-ups in Table 1 (k=1) is reachable by binary jumping algorithm [4]. Since meshes are not node-symmetric, a direction that the message is sent along has to be determined depending on source node type (corner, edge, inner), unlike the torus networks.

AAB algorithm in meshes alternates odd-numbered steps and even-numbered steps. In former steps odd-even pairs of processors exchange and then combine data, and in even-numbered steps even-odd pairs do the same. This is done first in all rows simultaneously, then in all columns. The total number of startups is a plain sum of their count in two phases, since the message-combining overhead is ignored. The size of messages grows from one to $\sqrt{P}$ in rows, then from $\sqrt{P}$ to $P/2$ in columns, so that altogether

TCO = $1+2+4+\ldots+\sqrt{P}+2\sqrt{P}+4\sqrt{P}+\ldots+P/2 = P-1$.

For a ring we have a better way of AAB: all processors send their data in one direction around the ring, one hop at a time, in P-1 steps.

The combining OAS on a ring can use again the binary jumping. Unlike OAB, the size of messages now varies (decreases) as the nodes extract their data:

TCO = $P/2 + P/4 + \ldots + 1 = P-1$.

In 2D-meshes we perform first OAS within the row of the source node with messages of size $\sqrt{P}$ (a combination of messages destined for a whole

column). Then OAS follows in all columns in parallel.

The combining AAS pattern is the same as for AAB, but the contents and size of messages are different. On a ring, all processors simultaneously rotate messages around in one direction, extracting and accumulating their part in each step. The size of messages thus gets reduced linearly,

TCO = $(P\text{-}1)+(P\text{-}2)+\ldots+1= P(P-1)/2$.

In 2D-meshes we can use the same rotating strategy first in rows and then in columns (with WH switching we can rotate messages even if there is no wrap-around link). The normalized message size will decrease linearly from $\sqrt{P}\,(\sqrt{P}\text{-}1)$ to $\sqrt{P}$ during rotation in rows, but each node will accumulate data of size $\sqrt{P}\,(\sqrt{P}\text{-}1)$ plus $\sqrt{P}$ left behind for its column. The same situation will repeat in the column rotation except the data left behind in a node are now $\sqrt{P}\text{-}1$ messages from partners in the row. Therefore

TCO = $2\,[\sqrt{P}\,(\sqrt{P}\text{-}1) + \sqrt{P}\,(\sqrt{P}\text{-}2)+\ldots+ \sqrt{P}\,] =$

$= P\,(\sqrt{P}\text{-}1)$

All the results are summarized in Table2.

Table 2. Parameters of CCs on a combining P-ring and a $\sqrt{P}\times\sqrt{P}$ mesh (upper bounds)

CC on a ring	WH, 1-port, full duplex, message combining	
	# of start-ups	TCO
OAB	$\lceil \log_2 P \rceil$	$\lceil \log_2 P \rceil$
AAB	$P-1$	$P-1$
OAS	$\lceil \log_2 P \rceil$	$P-1$
AAS	$P-1$	$P(P-1)/2$

CC on a mesh	WH, 1-port, full duplex, message combining	
	# of start-ups	TCO
OAB	$2\lceil \log_2 \sqrt{P} \rceil$	$2\lceil \log_2 \sqrt{P} \rceil$
AAB	$2(\sqrt{P}-1)$	$P-1$
OAS	$2\lceil \log_2 \sqrt{P} \rceil$	$P-1$
AAS	$2(\sqrt{P}-1)$	$P\,(\sqrt{P}-1)$

3.2. Scheduling CCs on the hierarchical ring network topology

Optimum scheduling CCs on the hierarchical ring topology at Fig.2 requires careful consideration. Generally hierarchical networks are a good match for combining one-to-all CCs as combined messages can flow between adjacent levels of hierarchy with only one start-up time. For all-to-all communications the number of start-up times may get reduced in comparison to non-combining CCs, but with longer messages the total occupancy of links is higher. In Table 3 we give both #of start-ups and TCO.

There is no advantage of message combining in OAB on hierarchical rings. The broadcast pattern on P = 16 processors is shown in Fig. 3. The source node has the message in time 0, other nodes get the message in the time step given inside table cells. In 4 steps all nodes get informed.

1	4	4	2
4	3	3	4
4	3	3	4
2	4	4	0

a)

3	4	4	3
4	1	2	4
4	2	0	4
3	4	4	3

b)

Fig. 3. OAB on hierarchical rings in 4 steps a) a corner source 0 b) an inner source 0

The combining OAS communication has the same complexity (in steps) as OAB. Unlike the OAB, the message size recursively halves from P/2 to 1.

The AAB pattern is more interesting. It can be done by gathering messages in 4 representative nodes and then broadcasting from them. E.g. for P=16:

1. Combining OAG on basic (level 0) rings, 2 steps, messages of size m and $2m$
2. Non-combining AAB by rotation on level 1 rings, 3 steps, messages of uniform size $4m$. A representative node on each basic ring has all 16 messages.
3. OAB within a basic ring. Two steps, message size $16m$.

The result: 7/47.

Non-combining AAB among P =16 nodes can be done in two phases:

1. AAB among corresponding nodes in rings at level 0. Rotation in three super-steps, each super-step 4 steps. Message size always $1m$.
2. AAS on 0-level rings: 3 steps, message size $4m$.

The result: 15/24.

The most complex AAS communication can be done either as non-combining or combining.

The combining AAS among P=16 nodes (7/138):

1. AOG in 2 steps, with message size $15m$, $30m$

2. AAS among 4 nodes at level 1 in 3 steps, message size always 16.
3. OAS from nodes at level 1 to 3 local nodes in 2 steps, message size 30*m*, 15*m*.

Non-combining AAS among *P*= 16 nodes (51/51):

1. AAS among rings at level 0. Three super-steps, 32 pair-wise exchanges in each super-step, 2 in each step. Message size is always 1*m*.
2. The local AAS in level 0 rings. 3steps, message size 1*m*.

All-to-all CC algorithms for *P*=64 nodes use the similar strategy as those for *P*=16. The results of all algorithms given above are listed in Table 3.

Table 3. Upper bounds of #startups/TCO for CCs on hierarchical rings and $\sqrt{P} \times \sqrt{P}$ meshes

WH, FD, 1-port	OAB	AAB	OAS	AAS
H.rings, *P*=16, non-combining	4	15/24	15	51
H. rings, *P*=64 non-combining	6	99/252	63	819
H. rings, *P*=16 combining	4/4	7/47	4/15	7/138
H. rings, *P*=64 combining	6/6	11/319	6/63	11/2658
2D mesh 4 x 4 combining	4/4	6/15	4/15	6/48
2D mesh 8 x 8 combining	6/6	14/63	6/63	14/448

Table 4. The best CC times on hierarchical rings and a 2D- mesh (upper bounds)

time [ns]	Mesh 4×4	Mesh 8×8	Rings 16	Rings 64
OAB	56	84	56	84
AAB	120	392	246	1386
OAS	100	312	100	312
AAS	252	1932	622	10742

5. Conclusions

The results, #start-ups and TCO parameters, are summarized in Table 3. For typical NoC parameters t_S = 10ns, t_1 = 1ns and for message size m = 4 bytes the minimum CC times are highlighted in Table 3 and for illustration numerically calculated in Table 4. It is seen that CCs perform generally better on 2D-meshes. Whereas OAB and OAS perform equally well on 2D-meshes and H-rings, AAB and AAS are much slower on H-rings. This is so under our implicit assumption that all the links in the network of Fig.2 have the same bandwidth (what may not be true in some cases). Also the combining mode is always faster than non-combining except AAB on small H-rings (*P*=16) with message size $m \geq 4$ bytes.

In future we want to use the obtained CC times for performance prediction of complete parallel applications on both considered NoCs. Another research direction should concentrate on a more scalable AAB and AAS algorithms on hierarchical rings.

Acknowledgement

This research has been carried out under the financial support of the research grants “Design and hardware implementation of a patent-invention machine”, GACR, GA 102/07/0850, Grant Agency of Czech Republic, “Architectures of Embedded Systems Network“, GA102/05/0467, and “Security-Oriented Research in Information Technology” Ministry of Education, MSM 0021630528.

References

[1] A. Ivanov, G. De Micheli, “Guest Editors’ Introduction: The Network-on-Chip Paradigm in Practice and Research”, *IEEE Design&Test of Computers*, Vol.22. No.5, Sept.-Oct. 2005, pp. 399-403.

[2] A. Jantsch, H. Tenhunen, *Networks on Chip*, Kluwer Academic Publ., Boston, 2003.

[3] Intel Tera-scale Computing, 2007. URL: *http://www.intel.com/research/platform/terascale/index.htm*

[4] Duato, J., Yalamanchili, S.: *Interconnection Networks – An Engineering Approach*, Morgan Kaufman Publishers, Elsevier Science, 2003.

[5] Jaroš Jiří, Ohlídal Miloš, Dvořák Václav: Complexity of Collective Communications on NoCs, In: Proc. of 5th International Symposium on Parallel Computing in Electrical Engineering, IEEE CS Press, 2006, Los Alamitos, CA, US, pp. 127-132.

[6] Hennessy, J.L., Patterson, D.A.: Computer Architecture - A Quantitative Approach. 4th Edition, Morgan Kaufman Publishers, Inc., 2006.