Optimizing Collective Communications on 2D-Mesh and Fat Tree NoC

Vaclav Dvorak and Jiri Jaros
Brno University of Technology
Brno, Czech Republic
{dvorak, jarosjir}@fit.vutbr.cz

***Abstract*—The paper investigates an impact of direct and combining collective communications models that may be critical for performance of parallel applications. Analysis provided for any given start-up time and message transfer time reveals the fastest collective communication mode in relation to the number of processing elements in 2D meshes and fat tree networks on a chip.**

Keywords-collective communications, 2D-Mesh; fat trees; message combining; graph embeddings

I. 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. 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 processing elements (PEs) makes use of wormhole (WH) switching. Wormhole switching reduces the effect of path length on communication time, but if multiple messages exist in the network concurrently (as it happens in CCs), contention for communication links may be a source of congestion and waiting times. To avoid congestion delays, it is necessary to organize CC into separated steps in time and to put into each step only such pair-wise communications whose paths do not share any links. The contention-free scheduling of CCs is therefore important.

The 2D-mesh topology has become the most commonly used choice in NoCs owing to its regularity which requires only short local links and allows simple deterministic or adaptive routing. However, one switch in every network node increases its cost and power consumption.

Topologies of the indirect networks can be more easily optimized for some particular application than those of the direct networks because the number of switches can be chosen independently of the number of PEs connected to the network. Therefore, they are more suitable for multiprocessor Systems on a Chip (SoC) in this respect.

Indirect networks with fat tree topology are constructed from smaller switches and approximate one centralized switch with a large number of ports. Switches in high-speed system area networks are constructed from distinct VLSI circuits where only the I/O pin count limits their size. In the NoCs, however, the silicon area is a limiting factor in addition to power consumption.

There are several classes of fat trees:

- Fat-trees FT(m, h) of height h built with m-port switches; all internal nodes are of degree m, where m as well as processor count P is an even integer. Each node has $m/2$ children and $m/2$ parents, see Fig. 1a.
- Generalized fat trees GFT(h, m, w) of height h, where each node has m children and w parents. Processor count P is limited to powers of m. Examples are at Fig. 1b and c.
- Extended generalized fat trees XGFT(h, m_1, m_2, …, m_h, w_1, w_2, …, w_h) where nodes in level i have m_i children and w_i parents. Processor count P is a product of all m_i's. One example is in Fig. 1d.

In the sequel, term "fat tree" will denote any of three classes above.

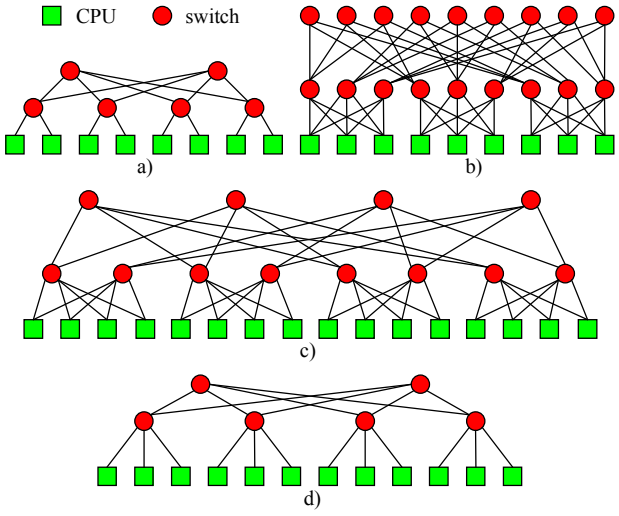


Figure 1. Fat trees a) 8-way FT(4, 2), b) 9-way GFT(2, 3, 3), c) 16-way GFT(2, 4, 2), d) 12-way XGFT(2, 3, 4, 1, 2).

978-0-7695-3979-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ICN.2010.12

The choice of a particular topology depends not only on the required speed of CCs, but also on fault tolerance and related cost of switches; e.g. robust topology in Fig. 1b has the highest fault tolerance, but the cost of switches is also very high.

Routing algorithms for fat trees and complexity of CC algorithms for some classes of fat trees have been analyzed in the literature [3], as well as the lower bounds and the achieved number of communication steps in direct implementation [4]. Combining messages in CCs on fat trees presented in [5] targeted only one particular system (Lemieux) interconnected by QsNet from Quadrics. Here we want

- to present improved upper bounds for 2D-meshes and fat trees up to 32 PEs obtained by evolutionary algorithms,
- to find the performance breakpoint between combining and direct communication model and
- to compare CC performance of 2D-meshes and fat trees.

The paper is structured as follows. In the following Section 2, we explain the communication model used in the paper. Section 3 summarizes the time complexity of CCs in WH networks: lower bounds on a number of communication steps for general networks and upper bounds obtained by authors for 2D-meshes and fat trees in case of uniform messages, without their combining (direct strategy). In Section 4, we first investigate combining CC algorithms on 2D-meshes and then transfer of known message combining CC algorithms of hypercube and 2D-mesh to fat trees. A performance boundary between direct and combining mode is the original contribution of authors. Analytical results are interpreted and commented on in Conclusions.

II. COMMUNICATION MODEL

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. From now on, when we refer to „collective communications", then we will assume only CCs involving the group of all processing elements (PEs).

In a single node broadcasting (One-to-All Broadcast, OAB) one PE sends the same message to all other PEs. Gossiping (All-to-All Broadcast, AAB) means that each PE sends the same message to all other PEs. In single node scattering, (One-to-All Scatter, OAS) one PE sends a personalized message to each other PE. The total exchange (All-to-All Scatter, AAS) is a parallel scattering of all PEs at a time. Since complexities of other collective operations and communications (such as gather or reduction) are similar to some of the above four types, we will not consider them explicitly.

Performance of CCs is closely related to their time complexity. The simplest time model of point-to-point communication in WH networks takes the communication time composed of a fixed start-up time t_s at the beginning (SW and HW overhead and synchronization), 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):

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

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 $h\, t_r << t_s + m\, t_1$), so that WH switching is considered distance-insensitive.

In the rest of the paper we assume that the CC in WH networks proceeds in synchronized steps. In each step of CC, a set of simultaneous packet transfers takes place along complete paths between source-destination node pairs that can be modeled by (1). We assume that paths travelled in every step are edge-disjoint, so that there is no contention for links and thus we do not consider contention delay in (1). 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 in a network represented by graph G will be determined in terms of the number of communication steps ("start-ups"), namely by $R(G)$, the upper bound on this number. Neglecting the hardware overhead in routers along the traversed path (term ht_r) and excluding contention for channels, CC times can be obtained approximately as the number of start-up delays Rt_s plus the sum of associated serialization delays $m_i\, t_1$,

$$T_{CC} = \sum_{i=1}^{R} (t_S + m_i t_1) = R t_S + m t_1 \sum_{i=1}^{R} k_i = R\, t_s + m t_1 \ \mathrm{TCO} \qquad (2)$$

where we do summation over all communication steps. Term TCO is the Total Channel Occupancy time normalized by a transfer time mt_1 of an elementary message of size m bytes, $m_i = m\, k_i$.

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. In the one-port model, 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 CCs 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, direct strategy). Message combining reduces the total number of messages, making each node send fewer messages of larger size. Reducing the number of start-ups can improve communication performance in case of

short messages when the start-up delays dominate in CC times. In non-combining model we have messages of the same size in all communication steps so that TCO = R in (2).

Finally, the lower bound on number of steps $R(G)$ depends on a link type; we have to distinguish between unidirectional (simplex) links and bi-directional (half-duplex HD, full-duplex FD) links. Typically R will be twice larger for HD links than for the FD ones. Further on we will consider FD links and the most frequent one-port/all-port router models.

III. Lower Bounds on $R(G)$ for Non-combining, Congestion-free CCs

One of the key design factors of an interconnection network is its topology. The lower bounds $R(G)$ for the network graph G depend on number of nodes P, port model (k), and channel bisection width B_C [6], Table I. Which one of three AAS lower bounds will dominate depends G on the topology.

The upper/lower bounds of selected CCs for the mesh and fat tree network topologies potentially useful in a NoCs are given in Table II. Wormhole switching and full duplex links have been assumed everywhere. Since meshes, unlike tori, are not node-symmetric, there are no elegant algorithms for them. Evolutionary algorithms can discover CC schedules that either match the already known lowest number of steps or provide new schedules close to lower bounds [7]. All the results in Table II have been obtained by evolutionary optimization tool [8]. The obtained upper bounds that match lower bounds are in bold, otherwise the lower bounds are given in brackets (in bold).

IV. Message Combining – Strategy for Short Messages.

As Table II shows, the number of communication steps without message combining can be quite high, especially when the number of PEs is large. It is then inefficient to store related data structures in routers and CC performance suffers as well. Message combining can improve that and we want to derive exactly when. In our analysis we will ignore the message-combining overhead which also slightly degrades the performance. Two cases are of interest:

- direct vs. combining strategy on a 2D-mesh and
- direct strategy on a fat tree vs. combining strategy on a hypercube or a 2D-mesh embedded into a fat tree.

Combining CC algorithms given below inherently assume 1-port 2D-meshes or hypercubes. All-port assumption would not decrease their running time.

A. CC Algorithms on Combining 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 I is pretty tight and not always reachable. Since meshes are not node-symmetric, OAB and OAS communication times depend on the source node type (corner, edge, inner), unlike the torus networks.

TABLE I. Lower Bounds on Complexity of CCs on Non-combining Networks

CC	**WH, k-port, FD Links, R = TCO [steps]**
OAB	$\lceil \log_{k+1} P \rceil$
AAB	$\lceil (P-1)/k \rceil$
OAS	$\lceil (P-1)/k \rceil$
AAS	$\max(\lceil P^2/(2B_C) \rceil, \lceil \Sigma/(Pk) \rceil, \lceil (P-1)/k \rceil)$

TABLE II. Upper Bounds on R from Table I for Selected All Port Networks

WH, FD, all-port, direct	**OAB**	**AAB**	**OAS**	**AAS**
FT-8	**3**	**7**	**7**	**7**
GFT-9	**2**	**3**	**3**	**3**
XGFT-12	**4**	12 (**11**)	**11**	15 (**11**)
GFT-16	**3**	**8**	**8**	**8**
FT -32	6 (**5**)	33 (**31**)	**31**	33 (**31**)
2D mesh 2 x 4	3 (**2**)	**4**	4 (**3**), **4**	**8**
2D mesh 3 x 3	**2**, **2**, **2**	**4**	**2**, **3**, **4**	**6**
2D mesh 3 x 4	**2**, **2**, **3**	**6**	**3**, **4**, **6**	**12**
2D mesh 4 x 4	**2**, **2**, **3**	**8**	**4**, 6 (**5**), **8**	17 (**16**)
2D mesh 4 x 8	**3**, **3**, **4**	**16**	**8**, **11**, **16**	**64**

AAB algorithm in meshes rotates messages first in rows and then in columns (with WH switching and FD links we can rotate messages even if there is no wrap-around link). Each node accumulates first B–1 messages of size m from partners in its row and then A–1 messages of size Bm from partners in its columns, so that

$$\text{TCO} = (B-1) + (A-1)B = AB-1 \tag{3}$$

The total number of startups is a plain sum of their count in two phases, i.e. $A + B - 2$.

In 2D-meshes we perform OAS first within the row of the source node by means of binary jumping with messages of size decreasing as follows:

$$m\hat{AB}/2,\ m\hat{AB}/4, \ldots, mA \tag{4}$$

where $\hat{X}$ is the nearest power of 2 greater or equal to X, $\hat{X} = 2^{\lceil \log X \rceil}$. At the end, all nodes in this row have the total data for all nodes in their columns and then they do in parallel vertical OAS within their columns. In this second phase the message size decreases as follows:

$$m\hat{A}/2,\ m\hat{A}/4, \ldots, m \tag{5}$$

so that in total

$$\mathrm{TCO} = A(\hat{B}-1) + \hat{A} - 1 \qquad (6)$$

The combining AAS pattern is the same as for AAB, but the contents and size of messages are different. Processors first form messages for one column each, combine them together into messages of size $A(B–1)m$ and pipeline them within all rows in parallel. Each receiver extracts A packets destined for its column, stores them and forwards the rest. After the row AAS is finished, each processor has messages from all B–1 colleagues within its row (on top of its own ones) destined for A–1 colleagues within its column. The message size decreases linearly from $A(B–1)m$ to Am in the first phase and from $B(A–1)m$ to Bm in the second phase. Therefore

$$\mathrm{TCO} = A\frac{(B-1)B}{2} + B\frac{(A-1)A}{2} = \frac{AB(A+B-2)}{2} \qquad (7)$$

All the results are summarized in Table III.

Conditions for superior performance of non-combining CCs over combining ones on all-port 2D meshes are derived from Table I and Table III:

$$\begin{aligned}
&\text{AAB:}\ \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/k \rceil - A - B + 2}{P - 1 - \lceil (P-1)/k \rceil},\ k = 2\\
&\text{OAS:}\ \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/k \rceil - \lceil \log A \rceil - \lceil \log B \rceil}{A(\hat{B}-1) + \hat{A} - 1 - \lceil (P-1)/k \rceil},\\
&\qquad k = 2,3,4\\
&\text{AAS:}\ \frac{mt_1}{t_s} \le \frac{\lceil P^2/(4A) \rceil - A - B + 2}{P(A+B-2)/2 - \lceil P^2/(4A) \rceil}
\end{aligned} \qquad (8)$$

Here we have used lower bounds of non-combining communication because upper bounds either match them or are reasonable close. The strongest lower bound for AAS on 2D-meshes is $P^2/(4A)$. All three functions $mt_1/t_s = f(P)$ are depicted in Fig. 2, where meshes up to size 4×8 from Table II are displayed as discrete points, among others.

It turns out, that depending on number of PEs, message combining makes sense if the message length m is in the range (0.1 to 0.9) (t_s / t_1). For example, if time per byte t_1 = 1ns and start-up time t_s = 10 ns (typical NoC parameters from [9]), we get m < (1 to 9) bytes.

B. Hypercube Strategy for CCs on Fat Trees

Design of combining CC algorithms for fat trees is based on graph embedding, simulating of one network by another. We can embed the n-cube (a guest graph G) into a fat tree with 2^n leaves (a host graph H) by mapping guest nodes into host nodes and guest edges into paths in the host. Combining CCs on the hypercube exchange data in a single dimension at a time in log P steps. To avoid contention, it is therefore important that paths connecting neighboring nodes in one dimension are edge disjoint, see Fig. 3 (connecting 0-subcubes is trivial).

TABLE III. PARAMETERS OF KNOWN CC ALGORITHMS ON A COMBINING 2D-MESH

CC on $A\times B$ mesh	WH, 1-port, FD, message combining	
	# of start-ups R	*TCO*
OAB	$\lceil \log A \rceil + \lceil \log B \rceil$	$\lceil \log A \rceil + \lceil \log B \rceil$
AAB	$A+B-2$	$AB-1$
OAS	$\lceil \log A \rceil + \lceil \log B \rceil$	$A(\hat{B}-1)+\hat{A}-1$
AAS	$A+B-2$	$AB(A+B-2)/2$

AAB communication is performed using channels of dimension 1, than channels of dimension 2, and so on. The size of messages will double each step. Hence

$$\mathrm{TCO} = 1 + 2 + \ldots + P/2 = P - 1 \qquad (9)$$

TCO in OAS communication is the same, because the size of the messages follows the same pattern, but in the opposite way. AAS is done again one dimension after another, but here the message size does not change. In each step, every processor exchanges $P/2$ messages destined for the PEs in the other half of the hypercube: its own messages + some already received in previous steps, starting with $P/2+0$, $P/4+P/4$, $P/8+(P/4+P/8)$…, and ending up with $1+(P/2–1)$ messages.

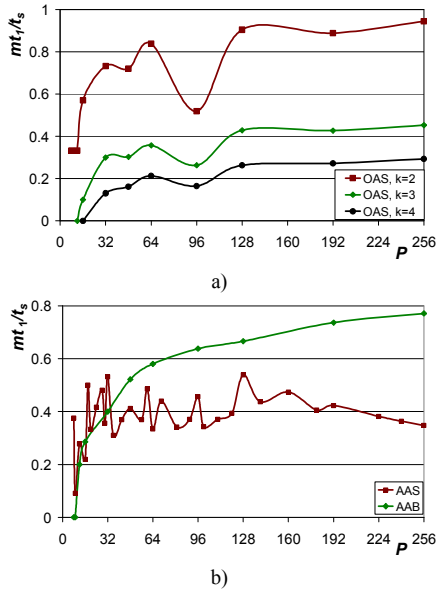


Figure 2. Performance breaking points between combining and non-combining CCs on 2D-meshes.

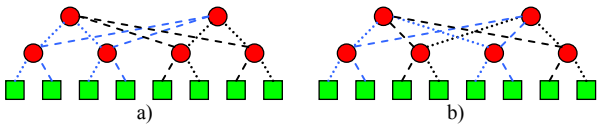


Figure 3. Connecting 1-subcubes (a) and 2- subcubes (b) by edge-disjoint paths in FT-8.

TABLE IV. PARAMETERS OF KNOWN CC ALGORITHMS ON A COMBINING A *P*-NODE HYPERCUBE

CC on a hypercube	WH, 1-port, FD, message combining	
	# of start-ups R	***TCO***
OAB	$\log P$	$\log P$
AAB	$\log P$	$P-1$
OAS	$\log P$	$P-1$
AAS	$\log P$	$(P/2)\log P$

Parameters of combining CC on hypercubes are listed in Table IV. Under the assumption of edge disjoint paths in each dimension we can combine Table I and Table IV and get conditions for faster non-combining mode in a form:

$$\text{AAB, OAS: } \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/w_1 \rceil - \log P}{P-1-\lceil (P-1)/w_1 \rceil} \qquad (10)$$

$$\text{AAS: } \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/w_1 \rceil - \log P}{(P/2)(\log P - \lceil (P-1)/w_1 \rceil)}$$

We have used the strongest AAS lower bound for fat trees which is $\lceil (P-1)/w_1 \rceil$ Combining AAB and OAS on embedded hypercube are always faster than non-combining modes on fat trees with w_1=1, whereas for w_1=2 it is not so. This case and two variants of AAS for $w_1 = 1$ and $w_1 = 2$ are depicted in Fig. 4.

C. *Mesh Strategy for CCs on Fat Trees*

In case of AAB and AAS we do rotation in one dimension and then in the other. It is therefore important if we are able to map paths connecting nodes in one mesh dimension to edge-disjoint path in the fat tree. Edge congestion ε in a certain dimension may be more than 1 and then we need ε-times more steps in that dimension. For example, embedding 3 x 3 mesh into 9-tree: $\varepsilon_1 = 1$, $\varepsilon_2 = 1$. Contrary, embedding 4 x 4 mesh into 16-tree: $\varepsilon_1 = 1$, $\varepsilon_2 = 2$.

Typically a group of adjacent PEs in a fat tree represents the nodes in one mesh dimension (B) with edge disjoint local neighbor-to-neighbor paths ($\varepsilon_1 = 1$). If the other dimension (A) has $\varepsilon_2 = 2$, the number of steps in Table III $R = A+B-2$ is to be replaced by

$$R = \varepsilon_1 (B-1) + \varepsilon_2 (A-1) \qquad (11)$$

This is the only correction that must be considered in Table III. In order to ensure the minimum value of R, we always try to assign $\varepsilon_1 = 1$ to the higher number of steps. The value of TCO remains the same, because some steps are repeated with shorter messages.

Binary jumping in OAS makes use only one non-local pair-wise communication in each mesh row or column. Provided that this binary jumping is congestion free, each topology will be characterized by the pair (ε_1, ε_2) and by the value of w_1. Conditions under which message combining on embedded meshes can outperform direct strategy on fat trees are obtained from Table I and Table III as follows:

$$\text{AAB: } \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/w_1 \rceil - \varepsilon_1 (A-1) - \varepsilon_2 (B-1)}{P-1-\lceil (P-1)/w_1 \rceil}$$

$$\text{OAS: } \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/w_1 \rceil - \lceil \log A \rceil - \lceil \log B \rceil}{A\varepsilon_2 (\hat{B}-1) + \varepsilon_1 (\hat{A}-1) - \lceil (P-1)/w_1 \rceil} \qquad (12)$$

$$\text{AAS: } \frac{mt_1}{t_s} \le \frac{\lceil (P-1)/w_1 \rceil - \varepsilon_1 (A-1) - \varepsilon_2 (B-1))}{(P/2)(\varepsilon_1 (A-1) + \varepsilon_2 (B-1)) - \lceil (P-1)/w_1 \rceil}$$

Fig. 5a and 5b show these conditions in a graphical form. As combining AAB and OAS on embedded meshes are for w_1=1 always faster than direct strategies, only the case (w_1=2, ε_1 =1, ε_2=1) is shown for illustration. The source of OAS is a corner node (k=2); the AAS condition displayed in Fig. 4b has been evaluated only for square meshes $n \times n$ to show the general trend. Moreover, embeddings of rectangular meshes into fat trees are topology-specific and their existence cannot be predicted easily. As it is seen from Fig. 3b, application of message combining is limited to very short messages and scarcely useful.

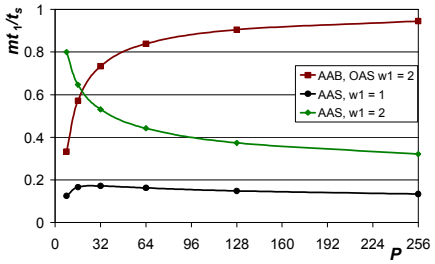
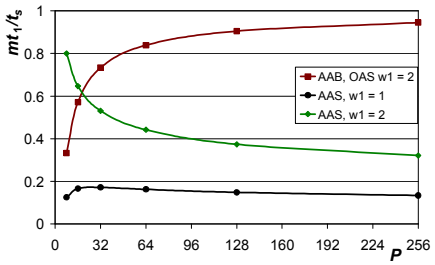
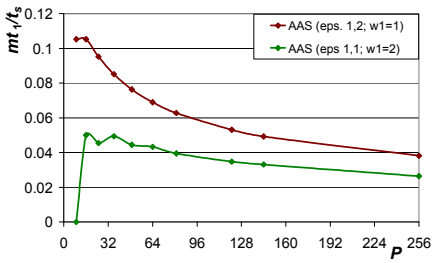


Figure 4. Performance breaking points for combining CCs on hypercubes embedded in fat trees.



a)



b)

Figure 5. Performance breaking points for combining CCs on meshes embedded in fat trees.

V. CONCLUSIONS

We addressed the problem "combine or not to combine messages?" with respect to performance of collective communications on selected NoCs. The results are summarized as follows:

1) CCs on 2D-meshes may profit from message combining only when message transfer time mt_1 is a fraction of start-up delay t_s. The fraction gets larger for more processors in the mesh.

2) Combining AAB and OAS on hypercubes embedded in fat trees with w_1=1 is always faster than related direct CCs on the fat tree. For short enough messages fat trees with w_1=2 can also benefit from message combining. However, combining AAS on embedded hypercube makes sense only for very small fraction $mt_1/t_s < 0.1$.

3) Combining CCs on meshes embedded in fat trees depend on edge congestion ε_1, ε_2 in two mesh dimensions. Combining AAB and OAS are always faster than native direct strategies on the fat tree with w_1=1, and with w_1=2 only if message length m is a fraction of t_s/t_1. Combining AAS will be rarely useful because this fraction must be less than 0.1.

In order to compare the performance of all-port 2D-meshes and all-port fat trees, we have used time per byte t_1=1ns, start-up time t_s=10 ns (typical NoC parameters from [9]) and two values of m; always the shortest time of the two modes (direct vs. combining mode) has been entered. Table V and VI give the best communication times, for combining mode in bold; embedded hypercubes and meshes are marked by symbols * and #, respectively.

It is seen, that for the selected parameters will message combining hardly improve performance of AAS on fat trees and of all CCs on 2D-meshes. It will be most useful for AAB and OAS on hypercubes or meshes embedded into fat trees with w_1=1 when the performance can improve even by more than 50%. In any case, for accurate evaluation one needs to know network topology, message length, time parameters t_s and t_1, the port model, and edge congestion for embedded networks. The obtained CC times can then be used to decide on message combining and to predict an overall performance of complete parallel applications with CC patterns.

In our performance optimization and comparison we did not take into account power consumption and cost of the NoCs, including their manufacturability. These are important attributes that could be considered in future research.

TABLE V. THE BEST CC TIMES [NS] REACHED ON ALL-PORT 2D-MESHES

***P*-meshes**	**t_1 = 1ns/byte, t_s = 10ns, m=8 bytes**			**t_1 = 1ns/byte, t_s = 10ns, m=64 bytes**		
	AAB	***OAS***	***AAS***	***AAB***	***OAS***	***AAS***
8	72	72	144	296	296	592
9	72	36	108	296	148	444
12	108	108	216	444	444	888
16	144	72	306	592	296	1258
36	288	144	1152	1184	592	4736

TABLE VI. THE BEST CC TIMES [NS] REACHED ON FAT TREES

***P*-fat trees**	**t_1 = 1ns/byte, t_s = 10ns, m=8 bytes**			**t_1 = 1ns/byte, t_s = 10ns, m=64 bytes**		
	AAB	***OAS***	***AAS***	***AAB***	***OAS***	***AAS***
8 (w_1=1)	**86***	**86***	126	**478***	**478***	518
9 (w_1=3)	54	54	54	222	222	222
12 (w_1=1)	**138#**	**136#**	270	**754#**	**808#**	1110
16 (w_1=2)	144	144	144	592	592	592
36 (w_1=1)	**298***	**298*#**	594	**2034***	**2034*#**	2442

ACKNOWLEDGMENT

This research has been carried out under the financial support of the research grant "Safety and security of networked embedded system applications", GA102/08/1429 (2008-10) of the Grant Agency of Czech Republic and "Security-Oriented Research in Information Technology", MSM 0021630528 (2007-13).

REFERENCES

[1] D. N. Jayasimha, B. Zafar, and Y. Hoskote, "On-Chip Interconnection Networks: Why They are Different and How to Compare Them". Platform Architecture Research, Intel Corporation, 2006. http://blogs.intel.com/research/terascale/ODI_why-different.pdf

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

[3] S. R. Őhring, M. Ibel, and S. K. Das, "On Generalized Fat trees", Proc. International Parallel Processing Symposium, IPPS 1995, pp.37.

[4] S. K. Das, S. R. Őhring, and M. Ibel, "Communication aspects of fat-tree-based interconnection networks for multicomputers", Proc DIMACS Workshop on Robust Communication Networks: Interconnection and Survivability, DIMACS Center, NJ, 1998, pp. 40-60.

[5] S. Kumar, and L. V. Kalé, "Scaling All-to-All Multicast on Fat-tree Network", Proc. 10th International Conference on Parallel and Distributed Systems, ICPADS 2004, pp. 205- 214.

[6] J. Duato, and S. Yalamanchili, "Interconnection Networks – An Engineering Approach", Morgan Kaufman Publishers, Elsevier Science, 2003.

[7] J. Jaroš, M. Ohlídal, and V. Dvořák, "Complexity of Collective Communications on NoCs", Proc. 5th International Symposium on Parallel Computing in Electrical Engineering, IEEE CS Press, 2006, Los Alamitos, CA, US, pp. 127-132.

[8] J. Jaroš, M. Ohlídal, V. Dvořák, "An Evolutionary Approach to Collective Communication Scheduling", Proc ACM Genetic and Evolutionary Computational Conference, New York, US, ACM, 2007, pp. 2037-2044.

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