Evolutionary Design of Fault Tolerant Collective Communications

Jiri Jaros [1]

[1] Brno University of Technology, Faculty of Information Technology,
Bozetechova 2, 612 66 Brno, Czech Republic
jarosjir@fit.vutbr.cz

Abstract. Scheduling of collective communications (CC) in interconnection networks possibly containing faulty links has been done with the use of the evolutionary techniques. Inter-node communication patterns scheduled in the minimum number of time slots have been obtained. The results show that evolutionary techniques often lead to ultimate scheduling of CC that reaches theoretical bounds on the number of steps. Analysis of fault tolerance by the same techniques revealed graceful CC performance degradation for a single link or node fault. Once the faulty region is located, CC can be re-scheduled during a recovery period.

Keywords: evolutionary design, fault tolerance, collective communications, wormhole switching.

1 Introduction

High performance computing platforms have been recently dominated by clusters of multi-core processor nodes [1] or for embedded applications by many cores interconnected by NoC (network on chip). The basic requirement for building the HPC systems turned out to be the low power consumption, in order that system parts can be close together and communication time thus minimized. For the same reason the CPU cores should be simple and processing nodes should be interconnected directly, without intermediate switches and routers. A class of interconnection networks of interest in this paper covers therefore direct networks, which for performance-driven environments converge on the use of pipelined cut-through (CT) message transmission, whose special case is wormhole routing (WH) [11] and source-based routing algorithms.

Since HPC systems include many processors, interconnection links and other units, their failure rate is much higher than the failure rate of the single-processor computers. Many fault-tolerant switching methods have been proposed to solve this problem. Unfortunately, the existing methods have many drawbacks such as low communication throughput, low fault-tolerant capability, and large hardware overhead.

In this paper, we consider faults to be permanent (i.e. damaged microcontrollers or communication links), as opposed to, say, transient, intermittent or even malicious

faults. As such, we are dealing with issues of fault tolerance. We wish to assess system behavior in the presence of a set of faults which is fixed for the duration of any routing attempt (defect in manufacture).

Generally, there are two kinds of faults in HPC systems, faulty links and faulty nodes. The first one is a damaged link interconnecting two parts of a HPC system. In this case, when the faulty link is located, it must be excluded from all routing algorithms (CC schedules). After new CCs are re-scheduled, the system is able to work properly only with little loss of performance. A node fault can be thought of as implying that all of that node's communication links are faulty.

In this paper, we want to analyze the complexity of collective communications in faulty networks. We employ an evolutionary algorithm, which is able to re-scheduled CCs after a single or multiple link or node fault with minimal possible loss of communication throughput. Of course, the network has to continue connected, i.e. at least one path for each source-destination pairs has to remain. The results of evolutionary techniques applied already to CC scheduling problem of medium size (tens of nodes) [2] are comparable well to optimum solutions obtained by mathematical means. However, networks in a faulty state are neither symmetrical nor regular, and analytic methods for scheduling do not exist. The results can be compared to theoretical lower bounds only.

The paper is structured as follows. Section 2 specifies the scheduling problem for CC on faulty networks while Section 3 presents an improved evolutionary algorithm for its solution. The results of CC scheduling in various faulty network topologies are summarized and discussed in Section 5. It also deals with fault tolerance of interconnection networks and possible recovery from a faulty state. The obtained results of evolutionary approach are discussed in Conclusion and possible future improvements are suggested.

2 Scheduling of Collective Communication in Faulty Networks

Pair-wise (point-to-point) as well as collective (group) communications involving all processors are frequently used in parallel processing and their timing complexity has a dramatic impact on performance. Since processors are connected only sparsely, the message can reach a destination processor directly, if source and destination processors are neighbors, or else through some intermediate nodes. The communication time from issuing the send request by one CPU until receiving data by another CPU represents an overhead of parallel processing which has to be minimized. The pipelined message transmission is considered only little sensitive to the source-destination distance; however, accumulating delays at traversing several nodes on the way should be minimized as well.

In this paper we are going to analyze only the frequently used collective communications involving all processors: one-to-all broadcast (OAB), all-to-all broadcast (AAB), one-to-all scatter (OAS, a private message to each partner), all-to-all scatter (AAS). Some other CCs, like all-to-one gather (AOG), have the same complexity as the basic four types.

Each CC can be seen as a set of point-to-point communications. The CC scheduling problem can be simply described as partitioning this set into as few subsets as possible that follow one another in sequence of synchronized steps and all communications in one subset proceed in parallel. The main goal is to avoid any conflicts in shared resources – links (channels). Several messages between source-destination pairs, not necessarily the neighbors, can proceed concurrently and can be combined into a single subset if their paths are link-disjoint. 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 (see Fig. 1).

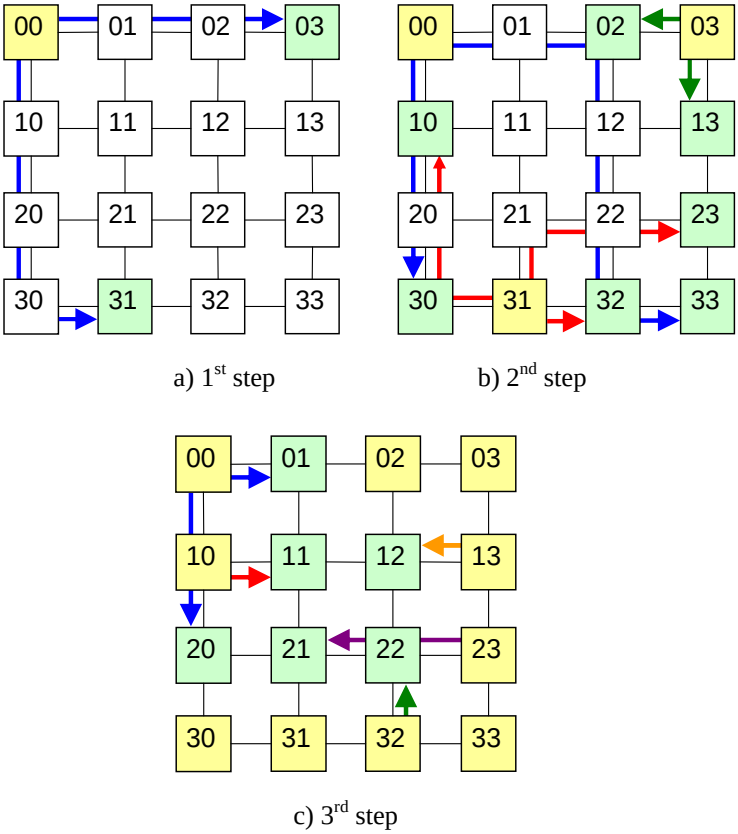

Fig. 1. One-to-all broadcast communication performed in three communication steps on 4x4 mesh. The initiator is node no. 00. Possible distributors of the broadcasted message in each step are marked by light yellow color. Newly informed nodes during the step are marked by light green color.

The number k of bi-directional channels between the CPU and a router (ports), that can be engaged in communication simultaneously, has a decisive impact on the number of communication steps; 1-port (k=1) or all-port (k=d) models are most

common, see Fig. 2. For the sake of performance we will consider only the all-port model (k=d).

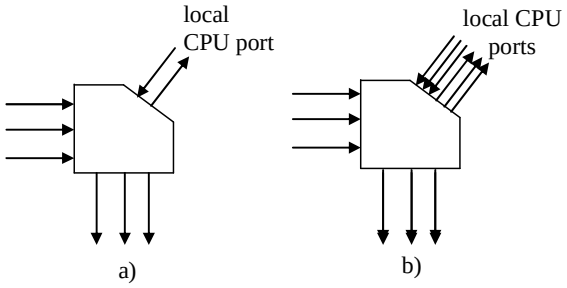

Fig. 2. Port models for 3-regular networks a) one-port router b) all-port router

Regardless the graph topology, there are known theoretical lower bounds on the number of communication steps. The broadcast communication (OAB) in WH network with P nodes and node degree d cannot be done in less than s steps, where $s = \lceil \log_{d+1} P \rceil$ is given by the number of nodes informed in each step, that is initially 1, $1+1\times d$ after the first step, $(d+1) + (d+1) \times d = (d+1)^2$ after the second step, etc.,…, and $(d+1)^s \geq P$ nodes after step s.

In case of AAB communication, since each node has to accept $P-1$ distinct messages, the lower bound is $\lceil (P-1)/d \rceil$. A similar bound is applied to OAS communication, because each node can inject into the network not more than d messages in one step. $P-1$ pair-wise communications, d of them per step, must be packed into the lowest number of steps in such a way that paths traversed in the optimum broadcast tree are edge-disjoint in each step.

For AAS communication pattern each of P processor sends an individual message to each of $P-1$ partners. A lower bound for AAS can be obtained considering that one half of messages from each processor cross the bisection and the other half do not. There will be altogether $2(P/2)(P/2)$ of such messages in both ways and up to B_C messages in one step, where B_C is the network bisection width [3]. In case of digraphs (graphs), B_C is taken as the (double) the number of (un)directed edges crossed by the bisection. However, some Δ messages originating and terminating in either half of a network cross the bisection as well. This gives the bound $(P^2/2 + 2\Delta)/B_C$ communication steps, since Δ messages cross the bisection twice. Another bound that concerns AAS used to be applied to SF routing only. If Σ denotes the sum of all shortest paths in a graph (from any source to any destination node) and if we can utilize only Pd channels in one step to avoid conflicts, then we cannot schedule AAS in less than $\lceil \Sigma/Pd \rceil$ steps. We have found that for the considered class of networks this latter bound is sharper, even for WH routing.

Table 1 summarizes the lower bounds for general graphs. Of course, communication bounds for AAB and AAS cannot be ever shorter than those ones for OAB and OAS respectively, if it applies.

Table 1. Lower bounds on complexity of CC in d-regular networks with P nodes

CC	CT (WH)
OAB	$\lceil \log_{d+1} P \rceil = \lceil (\log P)/\log(d+1) \rceil$
AAB	$\lceil (P-1)/d \rceil$
OAS	$\lceil (P-1)/d \rceil$
AAS	$\max[\lceil (P^2/2+2\Delta)/B_C \rceil, \lceil \Sigma/(Pd) \rceil]$

There are two major methods of increasing reliability with respect to faults in a system; namely, fault prevention and fault tolerance. Preventing all faults from a system is in most cases impossible or may cause some problems such as the delay or difficulty in maintenance. So we rely on fault tolerant algorithms to handle faults according to the following scenario [4]:

- In many fault tolerance approaches, the detection of an error is the first step of the recovery.
- When an error is detected, its origin and the possibility of containment of the proper maintenance activities are assessed.
- After that, the error recovery procedure comes. Its goal is to bring the system to an error-free consistent state, which means avoidance of a message loss and guarantee of conflict-free condition.
- After taking maintenance steps, the system will resume operation.

Our technique supposes that faulty links or nodes have been already detected, and the faulty region have been bordered. Conceptually, the faulty region may be considered as an island of faults in a sea of communication channels and nodes. In the same manner a ship is navigated around an island, it should be feasible to route a message around faulty region, see Fig. 3. It can be done using adaptive routing algorithms that re-route paths from source-destination pairs. However, these algorithms achieve only suboptimal results (i.e. possible faster CC schedules may exist, but they are not discovered, whereas only deterministic principles are used for rescheduling). In addition, many of fault-tolerant adaptive routing algorithms are not deadlock free, which introduce another delays and congestions. To relax all these restriction an evolutionary based technique was developed and used for finding CCs schedules on faulty networks.

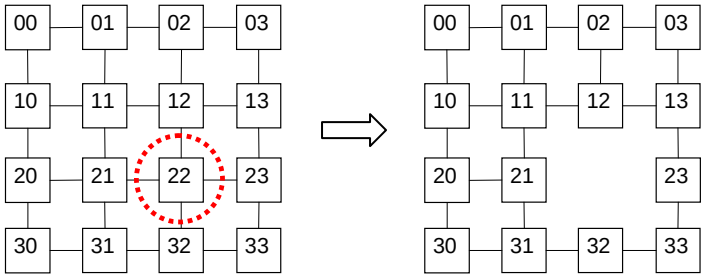

Fig. 3. Isolation of the faulty node 22

3 Evolutionary Scheduling of Collective Communications

We employ an Evolutionary Algorithm (EA) which is a powerful, domain-independent search technique inspired by Darwinian theory, to perform our search. Although a new methodology of designing near-optimal CC schedules is independent of the particular evolutionary algorithm, we restricted ourselves only to a simple EDA evolutionary algorithm without gene dependencies (UMDA) in this work.

Univariate Marginal Distribution Algorithm (UMDA) [5] is a very simple EDA [6] (Estimation of Distribution Algorithm) which does not reflect any interaction between genes (variables/solution parameters). The main advantage of this algorithm is better mixing of genetic material than is possible in standard GA [7], very simple implementation and much faster execution than a more complex EDA like BOA algorithm. Of course, any other EA can be used. Basic comparison of a success rate and time complexity of other types of EA applied to CC scheduling problem can be found in [8], [9].

This section describes, in more details, the elements of our evolutionary approach. Section 3.1 shows the global data structure and a preprocessing phase. Section 3.2 describes how the dataset is encoded, Section 3.3 presents the evaluation function used in EA, and Section 3.4 briefly describes acceleration and restoration heuristics used to increase a success rate and reduce execution time required to reach a good result. Parameters of used EA (UMDA) are outlined in Section 3.5.

3.1 Preprocessing Phase

An input data structure stores a topology description, a definition of CC, and a set of senders and receivers. The topology description is saved in a form of a neighbors list for each node, where the nodes are considered to be neighbors only if they are connected by a simple direct link. If a failure occurs, the topology description is changed to reflect a new topology of the interconnection network that arises by excluding of a faulty region.

After an input file is loaded, the data have to be preprocessed. The preprocessor takes the topology and finds all paths (shortest ones in the case of minimal routing) between all source-destination node pairs and stores them into a special data structure. This task is performed by a modified well known Dijkstra's algorithm.

3.2 Encoding

As broadcast and scatter CCs are completely different communication services, candidate solutions are encoded in separate ways.

A direct encoding has been designed for OAS chromosome; i.e. a chromosome contains an exact description of a schedule, see Fig. 4. The chromosome contains *P* genes; each one represents a particular point-to-point communication between the initiator and a destination node. A gene consists of two items: a utilized source-destination path (the first component) and the used time slot (the second component).

An AAS chromosome is created by extending the vector to a matrix, each row of which corresponds to one of OAS communications.

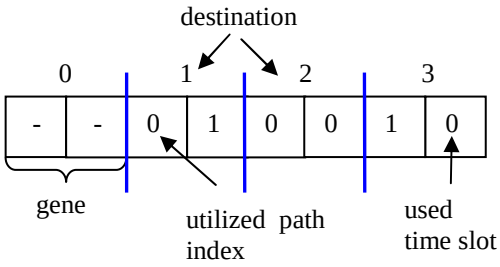

Fig. 4. The structure of OAS chromosome (0 is the initiator of OAS)

An indirect encoding has been designed for OAB; a chromosome does not include a broadcast tree, but only instructions how to create it. Each chromosome consists of *P* genes, one for each destination node, see Fig. 5. Individual genes are composed of three items: a source node for this pint-to-point communication, the index of a utilized path, and a step number. During each communication step, some new nodes are informed. These nodes can become distributors for next steps, and thus help the initiator of OAB to broadcast the message (all nodes receive the same message). That is why the additional component representing the message source must be incorporated into chromosome in this case.

The main disadvantage of this encoding is possible formation of some inadmissible solutions during the process of genetic manipulation. Simply said, a solution is inadmissible if it cannot lead to a correct broadcast tree (e. g. the situation when in a certain step a node should receive a message from a node that has not received it yet). That is why admissibility has to be verified for each chromosome before evaluating fitness and if it is necessary, the chromosome is restored. The AAB chromosome is then a collection of *P* OAB chromosomes, a kind of a matrix chromosome.

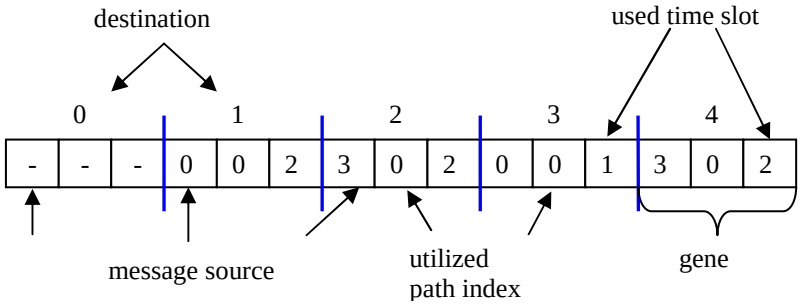

Fig. 5. The structure of OAB chromosome (0 is the initiator of OAB)

3.3 The Conflict Count Fitness Function

The main idea of fitness function is based on testing a conflict-free condition. We say two communications are in conflict if and only if they share the same channel in the same communication step (see Fig. 6). The fitness function is based on counting conflicts between all point-to-point communications realized in the same steps. The valid communication schedule for a given number of communication steps must be conflict-free. Valid schedules are either optimal (the number of steps equals the lower bound) or suboptimal. Evolution of a valid schedule for the given number of steps is finished up as soon as fitness (number of conflicts) drops to zero. If it does not do so in a reasonable time, the prescribed number of steps must be increased.

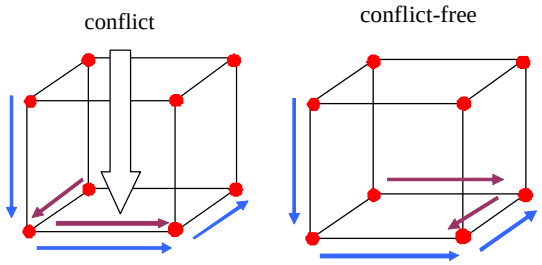

Fig. 6. Two point-to-point communications

3.4 Acceleration and Restoration Heuristic

A new heuristic has been developed for improving OAS/AAS optimization speed taking into account a search space restriction due to a limited message injection capability of network nodes. Because no node can send more than *k* messages in one communication step (*k*-port model), an acceleration heuristic checks this condition in the whole chromosome and redesigns ports' utilization in all communication steps before the fitness function is evaluated.

The second OAS/AAS heuristic replaces the mutation operator in an employed EA. It randomly swaps time slots of two point-to-point communications. These simple heuristics dramatically decrease the initial conflict count and lead to the better convergence of EA.

New heuristics for OAB/AAB chromosome restoration have been also developed and employed. The restoration (a repair of the broadcast tree) proceeds in subsequent communication steps. A check is made for every node whether the node receives the message really from the node already informed. If not so, the source node of this point-to-point communication is randomly replaced by a node that has already received the message. A change of the source node has naturally an impact on utilized channels. Hence the original path is replaced by newly chosen one from a list of exploitable paths between new source-destination pair.

To accelerate the convergence of the EA, OAB/AAB specific heuristic has been developed. It injects good building blocks into the initial population. For all point-to-

point communications of OAB, the time slot is set initially to the same value (step no. 0). By selecting correct time slots, the restoration heuristic produces corrected broadcast trees that violate the conflict-free condition in much fewer cases.

3.4 Parameters of EA

The simple UMDA evolutionary algorithm has been used for the search for near optimal communication schedules. The value of the population size was set to 60 individuals because higher values did not improve the quality of founded schedules and did not justify an increased computation time. The binary tournament selected the better half of the current population to form the parent subpopulation. The univariate marginal probabilistic model was created according to the parent subpopulation in each generation. New chromosomes were generated by the sampling of the estimated probabilistic model. Each chromosome was mutated by a simple mutation operator with probability of 90%. This operator is responsible for testing and changing possible source-destination paths for particular point-to-point communications. The mutation rate is very high due to huge number of source-destination pairs (thousands) whose amount growth exponentially with network diameter *D*. Finally, the newly generated solutions replace the worse half of the current population.

4 Results of Evolutionary Design

The evolutionary algorithm described previously has been applied to two networks that either already found the commercial application such as scalable Kautz networks, [10] and well known 2D-mesh, Fig 7.

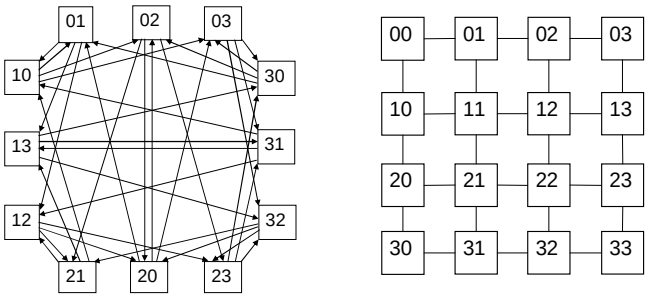

Fig. 7. Investigated topologies: Kautz12 network (left) and 4x4 2D-mesh (right)

First, we verified the ability of EA to discover optimal CC schedules for fail-safe interconnection networks, see Table 2. Obtained CC schedules for Kautz network met the theoretical lower bounds for all class of collective communications and thus cannot be improved anymore. Since 2D mesh is an irregular topology, 3 different situations depending on a source node position (corner, boundary, and center) were investigated. In all these cases, the theoretical lower bounds were reached.

Table 2. Achieved time complexity of CC schedules

all-port model	OAB	AAB	OAS	AAS
Kautz12	2	4	4	7
4x4 2D Mesh	3, 2, 2	8	8, 6, 4	16

As the Kautz network is known for its fault tolerance, we have tested performance degradation under a single link fault. A fault diameter of the Kautz12 network is D+2, meaning that among multiple links between any two nodes the longest path is 4. The network performance under a single link fault is given in Table 3 (with node 01 as the source node for OAB and OAS), but the network could operate even under a double link fault. In any case, when the link fault is detected, the new schedule could be computed in 20 seconds on a single processor and then the cluster could continue with a lower performance.

Table 3. Performance of Kautz12 network with a single faulty link (in # steps). A reduced performance is in bold.

Link	OAB	AAB	OAS	AAS
No fault	2	4	4	7
01-10	**3**	**6**	**6**	**9**
01-12	**3**	**6**	**6**	**9**
01-13	**3**	**6**	**6**	**9**
02-20	2	**6**	4	**9**
02-21	2	**6**	4	**9**
02-23	2	**6**	4	**9**
03-30	2	**6**	4	**9**
03-31	2	**6**	4	**9**
03-32	2	**6**	4	**9**
10-01	2	**6**	4	**9**
10-02	2	**6**	**5**	**9**
10-03	2	**6**	**5**	**9**
12-20	2	**6**	4	**9**
All other	2	**6**	4	**9**

The 2D meshes are very suitable interconnection network for System on Chips (SoC) because they need only very simple link arrangement on a 2D silicon chip. For the 4x4 2D mesh, performance degradation under a single link fault and a single node fault has been tested. Table 4 shows the performance degradation under a single link failure. The source node of OAB and OAS was appointed the node 00. Any link failure in 4x4 2D mesh increases the time overhead of AAS about 37%. The OAS and AAB communication will be delayed twice but only in two cases and OAB communication will be influenced noways. In all link failures, the proposed technique discovers the optimal communication schedule for a given CC.

Table 4. Performance of 4x4 2D Mesh network with a single faulty link (in # steps). A reduced performance is in bold.

Link	OAB	AAB	OAS	AAS
No fault	3	8	8	16
00-01	3	**15**	**15**	**22**
00-10	3	**15**	**15**	**22**
All other	3	8	8	**22**

Finally, we have tested the performance degradation under a single node failure (Table 5). A node fault can be thought of as implying that all of that node's communication links are faulty. From this table, it can be observed the same performance degradation under a single node fault as under a single link fault.

Table 5. Performance of 4x4 2D Mesh network with a single faulty node (in # steps). A reduced performance is in bold.

Node	OAB	AAB	OAS	AAS
No fault	3	8	8	16
01	3	**15**	**15**	**22**
10	3	**15**	**15**	**22**
All other	3	**15**	8	**22**

5 Conclusions

The evolutionary technique has been applied successfully to Kautz and 2D mesh interconnection topologies and quite general collective communications. Scheduling CC in the minimum number of steps without creating a conflict (a common channel in two transfers in the same step) led to optimal solutions or nearly optimal solutions.

The proposed technique can be with advantage used for failure recovery. CC schedules designed by the presented evolutionary technique are targeted for micro-programmed DMA engines residing in nodes of the network. They can be easily re-programmed in case of a link failure so that CC can sustain the highest possible performance even under limited connectivity.

Some of the found CC schedules attain the theoretical lower bound on the number of communication steps and thus there is no way to improve them further. Future research may reveal limits on a size of networks that can be handled by parallel implementation of evolutionary techniques.

Another direction for future research could explore a combining model for CC or fault tolerance of fat interconnection networks.

References

1. van der Steen, A.J., Dongarra, J.J.: Overview of Recent Supercomputers. TOP 500® Supercomputer Sites, Nov. 2007 Edition, http://www.arcade-eu.org/overview/
2. Jaroš, J., Ohlídal, M., Dvořák, V.: An Evolutionary Approach to Collective Communication Scheduling. In: 2007 Genetic and Evolutionary Computational Conference, pp. 2037-2044 ACM, New York (2007).
3. Duato, J., Yalamanchili, S.: Interconnection Networks – An Engineering Approach, Morgan Kaufman Publishers, Elsevier Science (2003).
4. Levi, S.T, Agrawala, A.K.: Fault Tolerant system design, McGraw-Hill Inc. (1994)
5. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Parallel Problem Solving from Nature – PPSN IV. LNSC, vol. 1411, pp. 178-187, Springer, Heidelberg (1996)
6. Larrañaga, P., Lozano, J.A: Estimation of Distribution Algorithms. Kluwer Academic Publishers, ISBN 0-7923-7466-5, London (2002)
7. Goldberg, D.: Genetics Algorithms in Search, Optimization, and Machine Learning, Addision-Wesley Publishing Company, (1989)
8. Jaroš, J., Dvořák, V.: Speeding-up OAS and AAS Communication in Networking System on Chips. In: Proc. of 8th IEEE Workshop on Design and Diagnostic of Electronic Circuits and Systems, pp. 4, ISBN 9639364487, Sopron, HU, UWH (2005)
9. Ohlídal, M., Jaroš, J., Dvořák, V.: Schwarz, J. Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks. In: EvoCOMNET. LNSC, vol. 3907, DE, pp. 267-278, Springer, Heidelberg (2006)
10. Stewart, L.C., Gingold, D.: *A* New Generation of Cluster Interconnect. White Paper, SiCortex Inc., (2006)
11. Dally W.J., Seitz, C.L.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, IEEE Trans. Computers, vol. C-36, no. 5, pp. 547-553, (May 1987).

Acknowledgements

This research has been carried out under the financial support of the research grants "Design and hardware implementation of a patent-invention machine", GA102/07/0850 (2007-9), "Safety and security of networked embedded system applications", GA102/08/1429 (2008-10), both care of Grant Agency of Czech Republic, and "Security-Oriented Research in Information Technology", MSM 0021630528 (2007-13).