Evolutionary-Based Conflict-Free Scheduling of Collective Communications on Spidergon NoCs

Jiri Jaros
Brno University of Technology
Bozetechova 2
612 66 Brno, Czech Republic
+420 54114-1207
jarosjir@fit.vutbr.cz

Vaclav Dvorak
Brno University of Technology
Bozetechova 2
612 66 Brno, Czech Republic
+420 54114-1149
dvorak@fit.vutbr.cz

ABSTRACT

The Spidergon interconnection network has become popular recently in multiprocessor systems on chips. To the best of our knowledge, algorithms for collective communications (CC) have not been discussed in the literature as yet, contrary to pair-wise routing algorithms. The paper investigates complexity of CCs in terms of lower bounds on the number of communication steps at conflict-free scheduling. The considered networks on chip make use of wormhole switching, full duplex links and all-port non-combining nodes. A search for conflict-free scheduling of CCs has been done by means of evolutionary algorithms and the resulting numbers of communication steps have been summarized and compared to lower bounds. Time performance of CCs can be evaluated from the obtained number of steps, the given start-up time and link bandwidth. Performance prediction of applications with CCs among computing nodes of the Spidergon network is thus possible.

Categories and Subject Descriptors

I.2.8 [**Artificial intelligence**]: Problem Solving, Control Methods and Search – *heuristic methods, scheduling*.

General Terms

Algorithms, Performance, Design.

Keywords

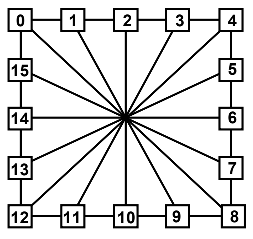
Collective communications, communication scheduling, evolutionary design, Spidergon, fat topologies, wormhole switching.

1. INTRODUCTION

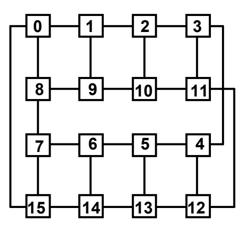
Networks on Chip (NoCs) ever more replace traditional on-chip communication architectures based on shared communication medium - a bus. A number of CPU cores, memory modules and other hardware units in Systems on a Chip (SoCs) or cores in many-core systems with memory physically distributed among computing nodes communicate by sending data through a NoC[1]

Classical logarithmic diameter networks, e.g. hypercubes, butterflies and fat trees, provide enough bandwidth for all-to-all communications, but do not map well into the 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 work we therefore restrict our attention to the Spidergon NoC topology with mostly local interconnection among processors.

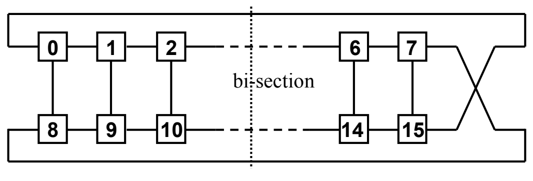
The Spidergon depicted in Fig. 1 is the novel interconnection network architecture suitable for the on-chip communication demands of SoCs in several application domains [2]. The Spidergon NoC first reported in [10], and later in [3], [4], has been recently adopted by STMicroelectronics [5] with the objective to realize low cost multiprocessor SoC implementation with topology opened for application-specific optimization. Spidergon is somewhere between the ring and mesh topologies: an even number of nodes is connected into a bidirectional ring and pairs of nodes are connected by a cross connection. Each edge in Fig. 1 represents two unidirectional physical links, one for each direction. In order to avoid deadlock, two virtual channels are multiplexed on each physical link. Fig. 1 depicts the 16-node Spidergon topology and its layout on a chip resembling a sparse mesh. Each node represents a router/switch (Fig. 2) and a CPU core.



(a) basic Spidergon topology



(b) the on-chip layout



(c) a ladder-like Spidergon topology

Figure 1. Isomorphic graphs of the 16-node Spidergon topology.

This topology is regular (constant node degree $d = 3$), node-symmetrical and edge-asymmetrical. Node symmetry implies identical routers within the entire network and simple routing algorithms. Moreover, the Spidergon scheme employs packet-based wormhole switching which can provide low message latency at a low cost. The network graph in the form of Möbius strip (Fig. 1c) demonstrates that bisection width B_C as defined in [6] is constant, $B_C = 8$, which becomes a bottleneck with a larger number of processors. The actual on-chip layout requires only a single crossing of metal layers, Fig. 1b, c. If the router at each node of the Spidergon NoC is a typical one-port router, the communication performance suffers. Enhancing the one-port router architecture to the all-port router architecture increases the cost but improves performance a great deal [7].

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 processing (BSP) behavior: the processing nodes access the network according to a global, structured communication pattern.

The performance of the collective communications (CCs 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) switching, in which each message is divided into small pieces (flow control digits, flits) that are pipelined through the network. Wormhole switching reduces the effect of path length on communication time, but if multiple messages exist in the network concurrently (as is happens in CCs), contention for busy links may be a source of congestion and waiting times. To avoid congestion delays, it is necessary to organize CC into separate steps in time and to put into each step only non-conflicting pair-wise communications whose paths are disjoint. The conflict-free scheduling of CCs is therefore important, because it leads to congestion-free CCs.

The deterministic shortest path routing algorithms proposed for the Spidergon architecture are so called Across First (aFirst) and Across Last (aLast) [4], [9]. Both algorithms are minimal source routing. An analytical performance model has been analyzed in [8] and the average message latency evaluated. Regarding CCs, only the broadcast and multicast CCs on Spidergon were studied in the past [7]. Other CCs, especially all-to-all communications have not been analyzed in the literature as yet.

In this work, we want to improve the performance of Spidergon NoC by designing such communication schedules that prevent any possible link contention. Optimized communication schedules can be uploaded into switch routing tables and boost the performance of many parallel algorithms. For this reason, four common CC patterns based on broadcast and scatter services will be analyzed.

The optimization of CC scheduling is based on evolutionary techniques. These techniques applied already to CC scheduling problem on hypercubes of medium size (tens of nodes) [11] were able to find the already known optimum solutions obtained analytically. However, for some CCs studied in this work analytic methods to find optimum schedules do not exist, so that the results can be compared only to theoretical lower bounds.

The paper is structured as follows. In the following Section 2 we analyze time complexity of CCs in WH networks, namely the lower bounds on the number of start-ups for general networks under the assumption of uniform non-combined messages. In Section 3 we present an evolutionary search for optimum CC schedules on simple Spidergons as well as on Spidergons with fat nodes. The time complexity of the evolutionary search is also discussed. Results and possible extensions are commented on in Conclusions.

2. TIME COMPLEXITY OF COLLECTIVE COMMUNICATIONS

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 (OA - One-to-All, AO - All-to-One, AA - All-to-All),
- 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 to 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 four 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 processes.

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 of a sender and a receiver), a serialization delay, i.e. the 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):

$$t_{WH} = t_s + mt_1 + ht_r, \qquad (1)$$

where t_1 is per unit-message transfer time and t_r includes a routing decision delay, switching and inter-router latency. A relatively small dependence on h may be taken into account by including $h_{max}t_r$ into t_s, so that only two parameters t_s and mt_1 are sufficient.

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" τ^{CC} (upper bound). Provided that the term $h_{max}t_r$ is included in t_s and excluding contention for channels, CC times can be obtained approximately as the sum of start-up delays plus associated serialization delays $m_i t_1$ in individual communication steps

$$t_{CC} = \sum_{i=1}^{\tau^{CC}} (t_S + m_i t_1) = \tau^{CC}[t_S + mt_1] \ . \qquad (2)$$

The above expression assumes that the nodes can only re-transmit/consume original messages, so that the length of messages $m_i = m$ remains constant in all communication steps. This is true in the so called non-combining model of communication; on the contrary, in the combining model the nodes can combine/extract partial messages with negligible overhead. The combining/non-combining model influences CC performance and either one can outperform the other in some cases. Further on we will consider the non-combining model only.

Possible synchronization overhead involved in communication steps, be it hardware or software-based, should be included in the start-up time t_s. Let us note that with uniform messages and a single clock signal domain, one barrier synchronization before CC might be sufficient to synchronize the whole CC. Communication steps would then follow in the lockstep. According to frequency of CCs and an amount of interleaved computation (BSP model) in a certain application, efficiency of parallel processing can be estimated.

The port model of the system defines the number k of CPU ports that can be engaged in communication simultaneously. This means that beside $2d$ network channels, 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 $k \leq d$, 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. 2 shows a one-port and an all-port router for the Spidergon network. In the one-port system, a node must transmit (and/or receive) messages sequentially. The messages may block on occupied injection channel, even when their required network channels are free. Architectures with multiple ports alleviate this bottleneck. In the all-port router architecture, there are as many local CPU channels as there are network channels that reduce the message blocking latency during CC operations. The port model determines the number of required communication steps and thus the CC performance.

Finally, the lower bound on the number of steps τ_{CC} depends on a channel type; we have to distinguish between unidirectional (simplex) channels and bi-directional (half-duplex HD, full-duplex FD) channels. Typically τ_{CC} will be twice as large for HD channels than for the FD ones. Further on we will consider FD channels and the all-port router model only.

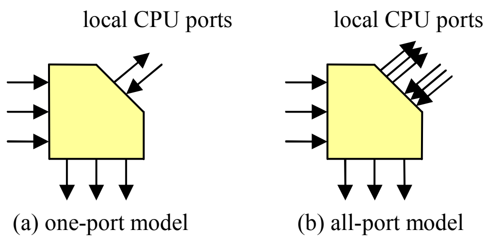


Figure 2. Port models for 3-regular Spidergon network.

The number of communication steps is in the first place influenced by the topology of an interconnection network. Generally 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, Table 1.

Table 1. Lower bounds on the number of communication steps τ_{CC} (WH, k-port, non-combining networks)

CC	τ_{CC} [steps]
OAB	$\lceil \log_{k+1} P \rceil$
AAB	$\lceil (P-1)/k \rceil$
OAS	$\lceil (P-1)/k \rceil$
AAS	$\max(\lceil P^2/(2B_C) \rceil, \lceil \Sigma/(Pk) \rceil, \lceil (P-1)/k \rceil)$

As far as the broadcast communication (OAB) is concerned, the lower bound on the number of steps $\tau_{\text{OAB}}(G) = s = \lceil \log_{k+1} P \rceil$ 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. Since the broadcast message is the same for all the nodes, each node once informed can help with distributing of the message in following steps.

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 cannot inject into the network more than k messages in one step.

The lower bound for AAS can be obtained considering that half the 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 channel bisection width [4]. Sometimes a stronger lower bound may be obtained considering the count of channels from all sources to all destinations (Σ) and the limited count Σ_1 of channels available for one step. In regular networks with constant node degree $\Sigma_1 = Pk$. As each node has to accept $P-1$ distinct messages, $\lceil (P-1)/k \rceil$ bound has to be also obeyed.

Specialized AAS lower bounds for the Spidergon network with $P = 4n$ nodes are easily derived from the following parameters:

$$B_C = 8,\ \Sigma_1 = 3P \text{ and } \Sigma = (P^2/2)(P/4+1);$$
$$\tau_{CC} = \max(\lceil P^2/(2B_C) \rceil, \lceil \Sigma/(Pk) \rceil, \lceil (P-1)/k \rceil) = \max(P^2/16, (P/2k)(P/4+1), \lceil (P-1)/k \rceil). \qquad (3)$$

Which lower bound takes effect depends on a particular network topology and the port model. For the network topologies potentially useful in NoCs are the lower bounds of selected CCs, computed from the formulas in Table 1 and from (3), given in Table 2. Wormhole switching and full duplex links have been assumed everywhere; port model $k = 1$ and $k = 3$ (all-port).

Table 2. Lower bounds τ_{CC} obtained from Table 1 for the Spidergon network (1-port/all-port)

Topology	OAB	AAB	OAS	AAS
6-gon	3/2	5/2	5/2	7/3
8-gon	3/2	7/3	7/3	12/4
12-gon	4/2	11/4	11/4	24/9
16-gon	4/2	15/5	15/5	40/16
20-gon	5/3	19/7	19/7	60/25
24-gon	5/3	23/8	23/8	84/36
28-gon	5/3	27/9	27/9	112/49
32-gon	5/3	31/11	31/11	144/64
36-gon	6/3	35/12	35/12	180/81

The Spidergon, as well as the bidirectional ring topology, though very simple, is not free from routing deadlock, because the channel dependency graph is not acyclic [4], [6]. This can be seen on a common permutation called the cyclic shift. The problem can be solved by the introduction of virtual channels [4] and by implementing rules on channel usage. However, in conflict-free CCs all source to destination paths are disjoint and therefore there is no competition for shared resources, no danger of deadlock and no need for escape virtual channels. When implementing CCs, we therefore use either one of two virtual channels.

3. EVOLUTIONARY SEARCH FOR THE OPTIMAL CONFLICT-FREE SCHEDULES

The selection of Evolutionary Algorithms (EA) for the scheduling problem has been justified already in [11]. Although the proposed methodology of designing near-optimal CC schedules is independent of the particular evolutionary algorithm, we restricted ourselves in this work only to a simple EDA evolutionary algorithm without gene dependencies (UMDA).

Univariate Marginal Distribution Algorithm (UMDA) [12] is a very simple EDA [13] (Estimation of Distribution Algorithm) which does not reflect any interaction between genes (variables/solution parameters). The main advantages of this algorithm are better mixing of genetic material than it is possible in standard GA [14], very simple implementation and much faster execution than more complex EDAs like BOA (Bayesian Optimization Algorithm [13]) algorithm. Of course, any other EA can be employed. Basic comparison of a success rate and execution time of other types of EA applied to CC scheduling problem can be found in [15] and [16].

The following subsections detail the evolutionary approach. Section 3.1 shows the global data structure and a preprocessing phase. Sections 3.2 and 3.3 describe how the dataset is encoded, Section 3.4 presents the evaluation function used in EA, and Section 3.5 briefly describes acceleration and restoration heuristics used to increase a success rate and to reduce the execution time required to reach a satisfactory result. Parameters of used EA (UMDA) are outlined in Section 3.6.

3.1 Preprocessing Phase

An input data structure maintains a description of a Spidergon topology, a definition of CC and sets of senders, receivers and intermediate routers. The topology description is saved in the form of node/router neighbor lists, where the nodes/routers are considered to be neighbors only if they are connected by a simple direct link.

After the input file is loaded, the data have to be preprocessed. In the first phase, the preprocessor divides the set of all nodes V^* into a set of transmitters T and a set of receivers R. Then, a set of terminal nodes $V \subseteq V^*$ is determined as the union $T \cup R$. The terminal nodes can inject/consume messages to/from the network, while the non-terminal nodes (routers) can only retransmit the messages. Finally, all the sets are ordered based on the node index.

The preprocessor generates all the shortest paths (the set R_{xy}) between all transmitter-receiver pairs x-y and saves them into a specific data structure in the operating memory for the second phase. This task is performed by a modified version of the well known Dijkstra's algorithm [17].

3.2 Scatter Encoding

As broadcast and scatter are completely different communication services, candidate solutions are also encoded differently. The definition of scatter encoding will be introduced first.

Let's consider a scatter based CC communication between M transmitters from set T and N receivers from set R.

1) The CC can be defined as a set $COMM$ of pair-wise transfers $comm_{src,dst}$ originating in $src \in T$ and terminating in $dst \in R$, where $src \neq dst$:

$$COMM = \{comm_{src,dst} : src \in T, dst \in R, src \neq dst\}. \qquad (4)$$

2) A direct encoding can be designed for the scatter-based communication schedules (i.e. an exact description of the schedule is stored in a chromosome). A chromosome can be formalized as n-tuple of genes:

$$chromosome = \begin{pmatrix} gene_{0,0} & \cdots & gene_{0,N-1} \\ \vdots & \ddots & \vdots \\ gene_{M-1,0} & \cdots & gene_{M-1,N-1} \end{pmatrix}, \qquad (5)$$

where M is the number of transmitters and N is the number of receivers while n is the total number of genes. Notice that

$$M, N \leq P \text{ and } n = M \cdot N. \qquad (6)$$

3) A gene $gene_{i,j}$ represents a single message transfer from the transmitter (source) $x_i \in T$ to the receiver (destination) $x_j \in R$, where $x_i \neq x_j$. The source and the destination are identified by the genes' indexes i and j. A gene is the ordered couple:

$$\begin{aligned} & gene_{i,j} = (l_{i,j}, s_{i,j}), 0 \leq i < M,\ 0 \leq j < N, i \neq j \\ & l_{i,j} \in R_{i,j} \\ & 0 \leq s_{i,j} < Steps \end{aligned} \qquad (7)$$

The first component $l_{i,j}$ represents a chosen path from transmitter x_i to receiver x_j stored in set $R_{i,j}$. The second component $s_{i,j}$

determines a selected communication step of the transfer between x_i and x_j. The total number of communication steps is given by the predefined parameter *Steps*.

4) The (shortest) path is defined as an ordered set of unidirectional channels. The length of the path is equal to h (hop count):

$$l_{i,j} = \{c_1, c_2, c_3, \ldots, c_h\},\ |l_{i,j}| = h, \tag{8}$$

where $src(c_1) = x_i$ and $dst(c_h) = x_j$.

5) Next consider a set *Genome* containing all the genes included in a *chromosome*:

$$Genome = \{gene_{i,j} : 0 \le i < M \text{ and } 0 \le j < N, i \ne j\}. \tag{9}$$

6) Finally, we can define a bijective mapping f from set *Genome* into set *COMM* meaning that each gene corresponds to a unique pair-wise transfer and also vice versa:

$$f : gene_{i,j} \in Genome \mapsto comm_{src,dst} \in COMM \tag{10}$$
$$\text{iff } x_i = src,\ x_j = dst.$$

3.3 Broadcast encoding

Consider a broadcast based CC communication between M transmitters from set T and N receivers from set R. Unfortunately, the set *COMM* cannot be constructed for broadcast based CCs, since each node already informed could also become a distributor of the broadcast message.

1) Therefore, the definition of CC is based on a set of messages *MSG* that have to be delivered during a given CC to each destination.

Let

$$MSG = \{msg_{src,dst} : src \in T,\ dst \in R,\ src \ne dst\} \tag{11}$$

be a set of messages originating in transmitters *src*, transported through the network via an intermediate distributor, and consumed by receivers *dst*. Notice that the message distributors are not known in advance.

2) A direct encoding has been designed to store broadcast-based communication schedules. A broadcast CC schedule is represented by the chromosome in the form of n-tuple of genes

$$chromosome = \begin{pmatrix} gene_{0,0} & \cdots & gene_{0,N-1} \\ \vdots & \ddots & \vdots \\ gene_{M-1,0} & \cdots & gene_{M-1,N-1} \end{pmatrix} \tag{12}$$

where M is the number of transmitters and N is the number of receivers while n is the total number of genes. Again, notice that

$$M, N \le P \text{ and } n = M \cdot N. \tag{13}$$

3) A gene $gene_{i,j}$ determines the way in which a receiver $x_j \in R$ obtains the broadcast message $msg_{i,j}$ from the transmitter $x_i \in T$ via a distributor $d_{i,j} \in V$. The producer and consumer of the message are identified by the genes' indexes i and j. Individual genes are represented by the ordered triplet:

$$\begin{aligned} & gene_{i,j} = (d_{i,j}, l_{i,j}, s_{i,j}),\ 0 \le i < M,\ 0 \le j < N,\ i \ne j \\ & 0 \le s_{i,j} < Steps \\ & d_{i,j} \in D_{s_{i,j},j} \subset V \\ & l_{j,i} \in R_{d_{i,j},j} \end{aligned} \tag{14}$$

The first component of the gene selects a distributor $d_{i,j}$ of the message $msg_{i,j}$. Besides the transmitter, the distributor can be any node from set $D_{s_{i,j}}$ that includes all nodes informed during all $s_{i,j} - 1$ steps; see the extended domination set theory [18]. The second component $l_{i,j}$ represents a chosen path between the distributor $d_{i,j}$ and the receiver x_j from the set $R_{d_{i,j},j}$. Analogously, the last component $s_{i,j}$ determines a selected communication step of the transfer between $d_{i,j}$ and x_j. The total number of steps is given by the predefined parameter *Steps*.

4) The (shortest) path was defined in the same way as in (8) as an ordered set of unidirectional channels.

5) Next, consider a set *Genome* containing all genes included in a *chromosome*

$$Genome = \{gene_{i,j} : 0 \le i < M \text{ and } 0 \le j < N, i \ne j\}. \tag{15}$$

6) Finally, we can define a bijective mapping f from set *Genome* into set *MSG*; each gene thus corresponds to a unique src-dst transfer of the message and also vice versa.

$$f : gene_{i,j} \in Genome \mapsto msg_{src,dst} \in MSG$$
$$\text{iff } x_i = src,\ x_j = dst \tag{16}$$

3.4 The Conflict Counting Fitness Function

This section proposes a formal description of the fitness function. The definition is the same for scatter and broadcast CC.

1) Let *SS* (Same Step) be a binary relation on the set *Genome*. Let a, $b \in COMM$ (scatter) or a, $b \in MSG$ (broadcast) message transfers be represented by $gene_{i,j}$ and $gene_{k,l}$, then

$$gene_{i,j}\ SS\ gene_{k,l} \text{ iff } s_{i,j} = s_{k,l} \tag{17}$$

Thus, two transfers are in relation *SS* if and only if they are executed within the same time step.

Now, we show that *SS* is an equivalence relation:

a) *SS* is reflexive, since no transfer can be performed in more than one time slot.

b) *SS* is clearly symmetric considering $s_{i,j} = s_{k,l}$, then $s_{k,l} = s_{i,j}$.

c) *SS* is transitive. Let a, b, c be elements of *Genome*. Whenever a *SS* b and b *SS* c, then also a SS c (a is executed within the same step as c whenever a is executed within the same step as b and b is executed within the same step as c).

Thus SS is an equivalence relation.

2) The equivalence relation *SS* induces the partition on the set *Genome*. Each equivalence class $[g_s]$ includes all transfers performed in the same slot.

3) Let $E_{a,b}$ be a set of all channels shared in two transfers a, b represented by genes $gene_{i,j}$, $gene_{k,l} \in Genome$, which utilizes the paths $l_{i,j}$ and $l_{k,l}$:

$$E_{a,b} = l_{i,j} \cap l_{k,l} \tag{18}$$

The number of conflicts between a and b can be obtained as the cardinality of the set $E_{a,b}$.

4) Define a multiset E_s including channels shared by all transfers within a given time slot, then

$$E_s = \bigcup_{a,b \in [g_s], a \neq b} E_{a,b} \tag{19}$$

5) The multiset E, covering all shared channels within the whole CC, can be obtained by a union over all equivalence classes. Thus

$$E = \bigcup_{[g_s] \in Genome/SS} E_s \tag{20}$$

6) The total number of conflicts can be obtained as the cardinality of multiset E. Thus

$$Fitness = |E|. \tag{21}$$

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 completed 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.

3.5 Acceleration and Restoration Heuristics

New heuristics have been developed to improve 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 a communication step, an acceleration heuristic checks this condition in the whole chromosome and redesigns terminal node utilization in all communication steps before evaluating the fitness function. The implementation of the heuristic is based on a node utilization histogram.

The second OAS/AAS heuristic replaces the mutation operator in the employed EA. This heuristic swaps associated communication steps of two transfers originating in the same transmitter. Actually, this heuristic performs a local search on a candidate solution based on the time domain. The conflicting transfers are rescheduled in spite of their timing to reduce total congestion.

Since illegal solutions of the OAB/AAB problem can appear during the process of genetic manipulation with OAB/AAB chromosomes, (a gene violates the condition $d_{i,j} \in D_{s_{i,j}}$ in eq. 14), the restoration heuristic has to be applied. This heuristic proceeds in subsequent communication steps and constructs a correct broadcast schedule. 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 replaced by a node that has already received the message. The replacement is made taking into account the node utilization histogram. An exchange of the distributor node $d_{i,j}$ has naturally an impact on utilized links $l_{i,j}$. Hence the original path is replaced by the one newly chosen from a list of exploitable paths $R_{d_{i,j},j}$ between new input-output pair $d_{i,j}$ and x_j.

To accelerate the convergence of the EA, an OAB/AAB-specific heuristics have been developed. First, good building blocks are injected into the initial population. First, the communication step $s_{i,j}$ is initially set to the same value ($s_{i,j} = 0$). Then the restoration heuristic corrects the time slots and produces the correct broadcast trees with significantly lower numbers of conflicts.

3.6 Parameters of EA

The simple UMDA [12] evolutionary algorithm has been used for the search for near optimal communication schedules. The value of the population size was set to 80 individuals because higher values did not improve the quality of found schedules and did not justify an increased computation time. The binary tournament selects the better half of the current population to form the parent subpopulation. The univariate marginal probabilistic model is created according to the parent subpopulation in each generation. New chromosomes are generated by the sampling of the estimated probabilistic model. Each chromosome is then mutated by a mutation operator that includes acceleration heuristics as a local search technique. The chromosome mutation probability is 0.9. 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 a large number of source-destination pairs that grows exponentially with the source-destination distance. Finally, the newly generated solutions replace the worse half of the current population.

4. EXPERIMENTAL RESULTS

The evolutionary optimization described previously has been applied to several slim all-port Spidergon networks as well as to several fat one-port Spidergon networks (Fig.3) with 2, 3 and 4 CPUs connected to a single router. Our goal is to find (near) optimal schedules for four basic CC patterns not studied on Spidergon networks as yet.

4.1 Scheduling CCs on the All-port Spidergon

We will use the all-port Spidergon NoC with P = 6, 8, 12, 16, …, 4n retaining the original topology [7], even though other extensions have been proposed [10]. We will comment the CC algorithms shortly on the 8-processor Spidergon network (Octagon), Fig. 3. One-to-all communications are done the same way for every source node due to node symmetry. OAB clearly can be done in 2 steps and OAS needs $\lceil 7/3 \rceil = 3$ steps. In order to implement AAB, we have to use such a broadcasting tree that is time-arc-disjoint (TADT) and can be used by all nodes simultaneously without creating conflict. For example node 0 could use this TADT:

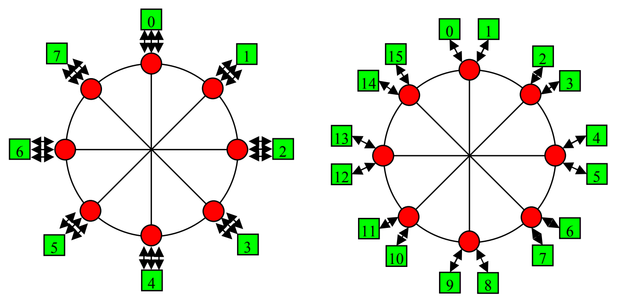
Step 1: $0 \rightarrow 7$, $0 \rightarrow 4$, $0 \rightarrow 1$
Step 2: $7 \rightarrow 6$, $1 \rightarrow 2$
Step 3: $4 \rightarrow 5$, $4 \rightarrow 3$.

We cannot join steps 2 and 3 though, because it would create conflict - one node cannot use more than 3 channels in a single step, because there is not more than 24 channels altogether.

The most complex AAS communication is not performed the same way by all nodes - there is no analogy to the TADT. In the design of AAS schedule, it is necessary to exploit some kind of combinatorial optimization (EA). Four steps were needed for AAS on Octagon with all-port (3-port) nodes, which is equal to the lower bound in Table 1.

For a Spidergon with the number of nodes $P = 4n$, we can find optimum CC schedules similarly as for Octagon using the evolutionary algorithm. The best obtained results from 25 runs are given in Table 3. The cases where the upper bound differs from the known lower bound are denoted by digits in bold.

Table 4 gives the time complexity of an evolutionary process for all the obtained schedules listed in Table 3. It summarizes the number of evolved generations with the constant population size of 80 individuals. From the table, it can be concluded that that the evolutionary design of one-to-all CC is pretty fast. By contrast, finding of a high quality all-to-all schedule is a much more complex problem that can take up to a few millions of generations, especially for AAS. A fluctuation of the numbers of generations is given by different distances between lower and upper bounds.



(a) slim all-port Octagon (b) fat one-port Octagon

Figure 3. Two different Octagon networks.

4.2 Scheduling CCs on the Fat Spidergon

Fat nodes with several CPUs connected by single port to the router could provide cheaper solution for Spidergon networks of a larger size, similarly as fat hypercubes do. However, it is a trade-off between such measures as cost and performance. As it is seen in Table 3, we can e.g. divide 12 cores in Spidergon network among 12, 6 or 4 nodes with 1, 2, or 3 cores per node, respectively. If we measure the cost by the total number of ports, it will be 48, 30, and 24 in these three cases. However, looking at the limits of the best possible CC performance, the impact is rather high. The difference between 2 or 3 cores per node is then negligible.

It should be noted that neither lower bounds for one-port, nor for all-port model apply here. The reason is that we cannot assign 3 network ports of a node explicitly to internal cores. Let us also note that the optimal schedules are not known for the fat Spidergon networks so far. Accordingly, Table 3 presents the best know upper bounds of CCs for the fat Spidergon networks found by evolution. Consequently, Table 4 summarizes the mean time complexity of evolving such schedules from all successful runs in terms of the number of evaluated generations. The number of successful runs varies from 25 in case of one-to-all CCs to only a single successful run in the case of all-to-all CCs.

Table 3. Numbers of steps in CCs for Spidergon networks reached by evolution

Topology	OAB	AAB	OAS	AAS
6-gon	2	2	2	3
8-gon	2	3	3	4
12-gon	2	4	4	9
16-gon	2	5	5	**17**
20-gon	3	7	7	**26**
24-gon	3	8	8	**37**
28-gon	3	9	9	**51**
32-gon	3	11	11	**68**
36-gon	3	12	12	**91**
2-fat_6-gon	4	12	11	12
2-fat_8-gon	4	16	15	17
2-fat_10-gon	5	20	19	25
2-fat_12-gon	5	24	23	37
2-fat_14-gon	5	29	27	50
2-fat_16-gon	6	37	31	66
2-fat_18-gon	6	42	35	86
3-fat_4-gon	4	11	11	11
3-fat_8-gon	5	24	23	38
3-fat_12-gon	6	41	35	82
4-fat_4-gon	4	16	15	16
4-fat_6-gon	5	24	23	38
4-fat_8-gon	6	33	31	64

Table 4. The number of evolved generations producing schedules summarized in Table 3 (in thousands).

Topology	OAB	AAB	OAS	AAS
6-gon	<1	<1	<1	<1
8-gon	<1	<1	<1	<1
12-gon	<1	2.3	<1	176
16-gon	<1	6.5	<1	101
20-gon	<1	4.1	<1	1266
24-gon	<1	12.4	<1	1942
28-gon	<1	24.9	<1	3221
32-gon	<1	74.1	<1	632
36-gon	833	148	<1	1031
2-fat_6-gon	<1	31.4	<1	31.5
2-fat_8-gon	2.1	786	<1	470
2-fat_10-gon	5.4	214	<1	1739
2-fat_12-gon	36.7	4330	<1	242
2-fat_14-gon	45.5	2740	<1	3048
2-fat_16-gon	58.9	1160	<1	6387
2-fat_18-gon	64	2540	<1	4281
3-fat_4-gon	<1	227	<1	604
3-fat_8-gon	15.4	2262	<1	3139
3-fat_12-gon	96.3	2060	<1	509
4-fat_4-gon	1.9	28.9	<1	1454
4-fat_6-gon	7.5	2270	<1	495
4-fat_8-gon	9.4	2056	<1	1412

5. CONCLUSIONS

This work has focused on the evolutionary design of near optimal schedules for slim all-port as well as fat one-port Spidergon networks. The paper has introduced a formal definition of the schedule encodings and the fitness function. The basic principle of acceleration and restoration heuristics has been also discussed.

The optimization of CC schedules has revealed that all-port Spidergon is very efficient in performing OAB, OAS and AAB. The evolution discovered the optimal schedules reaching the lower bounds for all Spidergon instances which cannot be improved any more. The weak point of all-port Spidergon network is AAS communication. Since parameter B_C does not scale with the number of nodes (always $B_C = 8$), the number of steps needed for AAS grows quadratically with P. For $P \geq 16$ the lower bounds on AAS steps have not been reached and it is not known whether they are reachable. However, the difference is only a few steps, good enough for engineering practice. The best found upper bounds are summarized in Table 3.

As far as the fat Spidergon is concerned, connecting 2, 3 or 4 one-port processors to a single router did not help much in CCs. Lower cost of this arrangement is gained at the expense of poorer performance of all CCs; whereas performance of OAB, OAS and AAB communications is close to one-port Spidergons with slim nodes and the same number of cores, the performance of AAS is rather close to all-port Spidergons. Since the lower bounds are not known for the fat one-port Spidergons, the presented results represent the most accurate estimation that we have obtained so far.

Future research will be oriented toward optimizing CCs on Spidergons modified for specific applications. Another research target is tolerance of faulty links in Spidergon network and their impact on CC performance.

6. ACKNOWLEDGMENTS

This research has been partially supported by the research grants "Safety and security of networked embedded systems applications", GA 102/08/1429 of Czech Science Foundation (2008-10), "Natural Computing on Unconventional Platforms", GP103/10/1517, Czech Science Foundation (2010-13), "Secured, reliable and adaptive computer systems", BUT FIT grant FIT-10-S-1 (2010) and the research plan "Security-oriented research in information technology", MSM 0021630528 (2007-13).

7. REFERENCES

[1] Ivanov, A., De Micheli, G. "Guest Editors' Introduction: The Network-on-Chip Paradigm in Practice and Research", *IEEE Design&Test of Computers*, vol. 22., no. 5, 2005, pp. 399-403.

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

[3] Coppola, M., Locatelli, R., Maruccia, G., Pieralisi, L., Scandurra, A. Spidergon: a novel on-chip communication network. *Int'l Symposium on System-on-Chip*, 2004.

[4] Coppola, M., Grammatikakis, M. D., Locatelli, R., Maruccia, G., Pieralisi, L. *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. Boca Raton, FL, USA: CRC CRC Press, Inc., 2008.

[5] STMicroelectronics. www.st.com.

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

[7] Moadeli, M., Vanderbauwhede, W., Shahrabi, A. A Performance Model of Communication in the Quarc NoC. In *14th IEEE International Conference on Parallel and Distributed Systems*, IEEE CS Press, 2008, pp. 908-913.

[8] Moadeli, M., Shahrabi, A., Vanderbauwhede, W., Ould-Khaoua, M. An Analytical Performance Model for the Spidergon NoC. In *21st International Conference on Advanced Networking and Applications (AINA'07)*, IEEE CS Press, pp. 1014 - 1021.

[9] Concer, N., Iamundo, S., Bononi, L. aEqualized: a Novel Routing Algorithm For The Spidergon Network On Chip. In: *Design, Automation and Test in Europe*, DATE 2009, Nice, 2009, IEEE CS Press, pp. 749-754.

[10] Karim, F., Nguyen, A. *An Interconnect Architecture for Networking Systems on Chips*. IEEE Micro, 2002, pp. 36-45.

[11] Jaros, J., Ohlidal, M., Dvořák, V. An Evolutionary Approach to Collective Communication Scheduling, In *2007 Genetic and Evolutionary Computation Conference*, New York, US, ACM, 2007, pp. 2037-2044.

[12] Mühlenbein, H., Paaß, G. From recombination of genes to the estimation of distributions I. Binary parameters. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature – PPSN IV*, pp. 178-187, 1996.

[13] Larrañaga, P., Lozano, J. A. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, London 2002, ISBN 0-7923-7466-5.

[14] Goldberg D. *Genetics Algorithms in Search, Optimization, and Machine Learning*, Addision-Wesley Publishing Company, 1989.

[15] Jaros, J., Dvorak, 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*, Sopron, HU, UWH, 2005, pages 4.

[16] Ohlidal, M., Jaros, J., Dvorak, V., Schwarz, J. Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks, In *Lecture Notes in Computer Science*, 2006, no. 3907, DE, EvoWorkshops 2006, pp. 267-278, ISSN 0302-9743.

[17] Dijkstra, E. W. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1954.

[18] Tsai, Y., McKinley, P. K. An Extended Dominating Node Approach to Broadcast and Global Combine in Multiport Wormhole-Routed Mesh Networks. In *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 1, 1997.