Understanding the Interconnection Network of SpiNNaker

Javier Navaridas†, Mikel Luján*, Jose Miguel-Alonso†, Luis A. Plana*, Steve Furber*

† Dpt. of Computer Architecture and Technology
The University of the Basque Country, Spain
Manuel de Lardizabal, 1. 20018 San Sebastian
(+34) 943 018019

* School of Computer Science
The University of Manchester
Oxford Road, Manchester M13 9PL, UK.
(+44) 161 306 9280

{javier.navaridas, j.miguel}@ehu.es {mikel.lujan, plana, steve.furber}@manchester.ac.uk

ABSTRACT
SpiNNaker is a massively parallel architecture designed to model large-scale spiking neural networks in (biological) real-time. Its design is based around *ad-hoc* multi-core System-on-Chips which are interconnected using a two-dimensional toroidal triangular mesh. Neurons are modeled in software and their spikes generate packets that propagate through the on- and inter-chip communication fabric relying on custom-made on-chip multicast routers. This paper models and evaluates large-scale instances of its novel interconnect (more than 65 thousand nodes, or over one million computing cores), focusing on real-time features and fault-tolerance. The key contribution can be summarized as understanding the properties of the feasible topologies and establishing the stable operation of the SpiNNaker under different levels of degradation. First we derive analytically the topological characteristics of the network, which are later confirmed by experimental work. With the computational model developed, we investigate the topology of SpiNNaker, and compare it with a standard 3-dimensional torus. The novel emergency routing mechanism, implemented within the routers, allows the topology of SpiNNaker to be more robust than the 3-dimensional torus, regardless of the latter having better topological characteristics. Furthermore, we obtain optimal values of two router parameters related with livelock and deadlock avoidance mechanisms.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Network topology*
C.4 [**Performance of Systems**]: – *Fault tolerance, Performance attributes*

General Terms
Performance, Design, Reliability, Experimentation.

Keywords
Analytical Evaluation, Biologically Inspired Architecture, Fault Tolerance, Interconnection Networks, Massively Parallel Architecture, Performance Evaluation, Real-time Applications, Spiking Neurons, Systems on Chip.

ICS'09, June 8–12, 2009, Yorktown Heights, New York, USA.

1. INTRODUCTION
The SpiNNaker system is a biologically-inspired massively parallel architecture of bespoke multi-core System-on-Chips (SoC) designed with the aim of simulating up to a billion spiking neurons in (biological) real-time. Its main applications are to be the "mind" of a robot providing real-time *stimulus-response* behavior [8], and as an experimental platform to improve our understanding of the brain architecture. Biological spiking neural networks communicate by means of spike events which occur when a neuron is stimulated beyond a given threshold and fires. Spike events are communicated to all connected neurons, with typical fan-outs of the order of 1000. Fortunately, applications such as these have abundant parallelism and no explicit requirement to maintain consistency in shared memories. Another characteristic of the biological process is its inherent resilience to failures; neurons may die and spikes may be missed. Furthermore, the biological process [6, 10] advances at very low pace when compared to standard computer components: milliseconds *vs.* microseconds, and with neurons spiking in average 10 times per second (average firing rate of 10Hz).

SpiNNaker takes advantage of these characteristics to deploy a well-balanced, low-power massively parallel architecture. The largest configuration houses 2^{16} nodes creating a system with over one million computing cores, able to simulate neural nets with more than one billion (10^9) neurons. Its design is based around bespoke multi-core SoC which are interconnected using a two-dimensional (2D) toroidal triangular mesh. Neurons are modeled in software and their spikes generate packets that propagate through the on- and inter-chip communication fabric relying on bespoke on-chip multicast routers.

This paper focuses on the Interconnection Network (IN) that implements the inter-chip communication used to simulate synaptic connections. We describe SpiNNaker and its topology (Section 2) and derive analytically its most interesting characteristics (Section 3). SpiNNaker sits at a rather different design point compared with High-Performance Computing (HPC) systems, such as those in the TOP500 list [7]. Those are commonly built with very fast processors over not-so-fast networks, while SpiNNaker uses low-power cores working at 200MHz with an IN capable of communicating at 1 Gbps. These features dictate that our experimental study (Section 4) is far from a traditional one in HPC; latency and throughput become secondary figures of merit. In fact the main concern is to be able to understand the balance between injection load and the number of dropped packets, as SpiNNaker implements a packet dropping

mechanism to avoid deadlock and livelock. Given the large scale of the system, throughout the experimental study we consider scenarios in which the IN suffers different levels of failures. As the IN traffic is dependent on the specific features of the spiking neural network, the experiments consider a worst case scenario where the traffic disregards locality and pushes the injection load beyond the biological 10Hz average neuron firing rate. Finally, Section 5 presents related work, while Section 6 summarizes the work.

Recapitulating, the main contributions of this paper are:

- an analytical characterization of SpiNNaker's IN topology and its properties;
- a computational model of the system used to carry out simulation-based studies that validate the analytical study;
- the characterization of optimal parameters for the deadlock and livelock avoidance mechanisms;
- a worst-case study of the stability of the SpiNNaker IN under pessimistic levels of failures and injection load scenarios beyond the expected normal operation; and
- an assessment of the efficiency of emergency routing to keep the system operating stably.

2. SPINNAKER ARCHITECTURE

To emulate the very high connectivity of biological systems, SpiNNaker uses a self-timed, packet-switched network which supports efficient multicast, high bandwidth, and low-delay communications. The heart of the communication infrastructure is an on-chip router and the self-timed implementation of the fabric that allows the seamless extension of the on-chip communications to include the inter-chip links. Further details of the communication fabric can be found in [19].

2.1 SpiNNaker Node

The basic block or node is the SpiNNaker chip. It contains one multi-core SoC with 20 low-power ARM968 cores. Each core has a tightly-coupled dedicated memory that can hold 32KB of instructions and 64KB of data. Each core runs an independent event driven neural process with events generated by modules such as timer, vector interrupt controller (VIC), communication controller (CC) and DMA controller. All the cores in a chip share a SDRAM of up to 1 GB storing, for example, synaptic connection information. Access to this shared storage space is carried out by means of a self-timed NoC [9] which is used to connect resources in the chip. The router can be accessed through the NoC, but only for configuration purposes; during normal execution the ARM cores use the communication controller to send or receive packets. This NoC provides higher communication bandwidth (8 Gbps), lower contention and lower power consumption than any typical bus-based interconnect [20]. A model of the SpiNNaker chip is depicted in Fig. 1. Detailed simulations of the chip using Verilog and SystemC showed that each core can model in biological real-time up to around 1000 individual neurons [13].

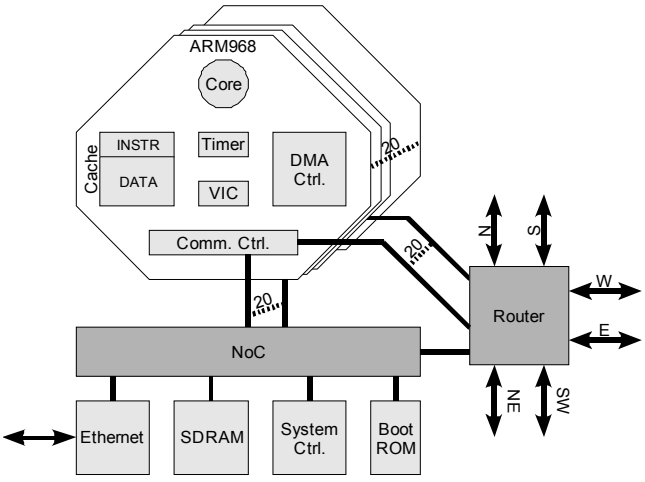

Figure 1. Schematic model of the SpiNNaker chip with all its components depicted.

2.2 On-chip Router

Each chip incorporates a router [9] that allows inter- and on-chip communications. The router is the heart of the NoC and occupies approximately 10% of the chip area. Its primary role is to direct each neural event packet to those cores where the connected neurons are located. Given the area constraints of the SpiNNaker chip, popular NoCs based on 2D meshes are not feasible because of the large area they occupy.

A depiction of the router is shown in Fig. 2. It has 20 ports for internal use by the ARM cores and six ports to communicate with six adjacent chips. All ports are full-duplex and implement self-timed protocols. The organization within the router is hierarchical; ports are merged in three stages before using the actual routing engine. Note that the router is able to forward a single packet at once, but it works faster than transmission ports. Thus, most of the time routers will be idle, and router delay barely affects the pace at which packets are processed.

The router is designed to support point-to-point and multicast communications using small packets (5 bytes, normally 40 bits). The multicast engine helps reducing pressure at the injection ports, and reduces significantly the number of packets that traverse the network, compared to a pure point-to-point alternative. Routers make routing decisions based on the source address (neuron identifier) of the packets.

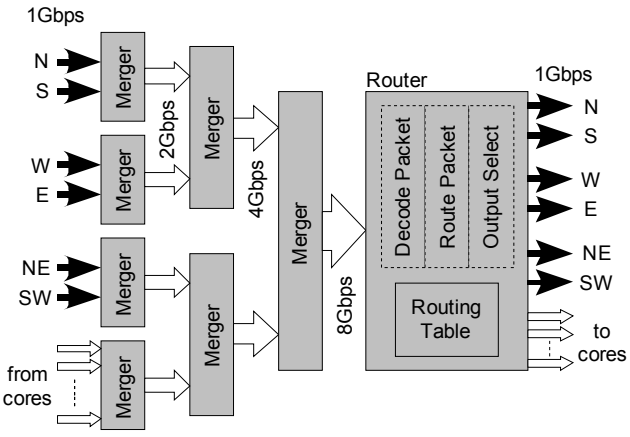

Figure 2. Architecture of the router. Black arrows represent links outwards from the chip. White arrows represent hard-wired links within the chip.

In other words, a neural-event packet does not contain any information about its destination(s), only the neuron that has fired.

The information necessary to deliver these packets to all the relevant cores and chips is compressed and distributed across 1024-word routing tables. Each router contains one routing table, and the tables have to be preloaded using application-specific information. To allow further compression, the tables offer a masked associative route look-up and the routers are designed to perform a *default routing* (no entry needed in the tables) that sends the packet to the port opposite to the one the packet comes from. For example, if the packet comes from the North it will be sent to the South. Thus, the expected shape of the routes between chips is by means of two straight lines with one inflection point [14].

The network topology allows two-hop routes among neighbor chips (see Fig. 3) which are denoted as *emergency routes*. These routes may be invoked to bypass problematic links due to transient congestion states or link failures. In practice, only one of the two possible turns is implemented in the router to minimize chip area.

The flow-control mechanism is very simple. When a packet arrives to an input port, one or more output ports are selected and the router tries to transmit the packet through them. If the packet cannot be forwarded, the router will keep trying, and after a given period of time it will also test the clockwise emergency route. It will try both the regular and the emergency route. Finally, if a packet stays in the router for longer than a given threshold (*waiting time*) the packet will be dropped to avoid deadlock scenarios. To avoid livelock situations, packets have an *age* field in their header. When two ages pass and the packet is still in the IN, it is considered *outdated* and dropped. The ages are global to the whole system and its time-span is arbitrary, a router configuration parameter. Section IV-*C* provides bounds for the recommended values for these two network parameters (*waiting time* and *age length*).

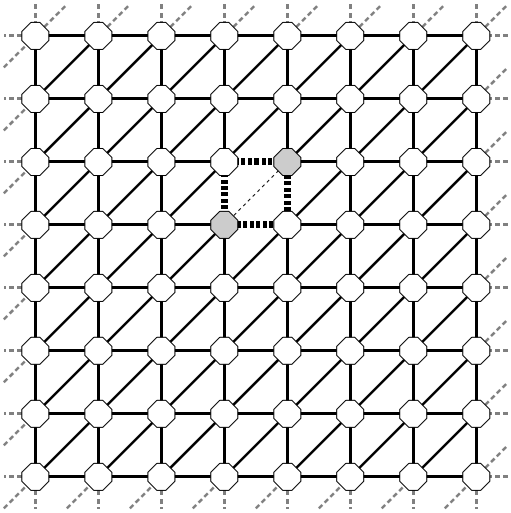

Figure 3. Example of an 8×8 SpiNNaker topology. Peripheral wrap-around connections are not depicted for the sake of clarity. The regular route (thin and slashed line) and the two emergency routes (thick and dotted lines) among the two shaded nodes are shown.

Emulating the behavior of biological neural networks, dropped packets in SpiNNaker are not re-sent. Losing neurons (one per second in human brains) or signals does not impede the normal functioning of the biological processes; however, dropping level must be kept (*very*) low.

2.3 Topology of the Interconnection Network

SpiNNaker chips are arranged in a 2D mesh topology with links to the North, South, East, West, Southwest and Northeast neighbors. An 8×8 instance of this topology is depicted in Fig. 3. Note that chips at the network boundaries are connected by means of peripheral, wrap-around links not shown in the figure for the sake of clarity.

The external 6-ports in the SpiNNaker chip could also allow for a three-dimensional (3D) torus arrangement. In fact, the topological properties of a 3D torus, such as bisection bandwidth and distance related characteristics, are better than those of the SpiNNaker topology. Nonetheless, the topology chosen for SpiNNaker has some interesting advantages:

- a two-dimensional system is easier to manufacture and deploy,
- the diagonal links add redundancy to the design, and
- the previously described emergency routing can be easily implemented.

A three-hop emergency routing could be implemented in the on-chip router to support a 3D torus configuration, but at a significant cost in terms of chip area. Note also that routing in a 3D torus requires more entries in the routing tables, as regular routes are composed by three straight lines instead of two. This increases the entries in the routing tables roughly by a 33%, which may force an increase in the number of entries per table and, therefore, the chip area. In Section 4.4 we compare the behavior of the two topologies, illustrating how the greater stability provided by emergency routing tips the scale in favor of the SpiNNaker topology.

3. TOPOLOGICAL PROPERTIES

An analytical study of the IN allows us to derive some representative characteristics: the maximum theoretical throughput for uniform load (computed from the bisection bandwidth), the average distance between nodes, and the diameter of the network. For the sake of simplicity, we consider only square topologies with an even number of nodes per dimension. Expressions for other dimension ratios and/or odd number of nodes per dimension could be derived analogously.

3.1 Definitions

The following conventions are used to denote elements and characteristics of the IN under study:

n: The number of nodes per dimension.

N: The total number of nodes in the system; $N=n^2$.

B_B: The bisection bandwidth defined as the bandwidth of a minimum cut joining two equal partitions of the network, measured as the number of links in such cut.

Θ: The maximum theoretical throughput measured as the number of packets per cycle each node can inject, when using uniform traffic.

D: Diameter of the network, *i.e.* the maximum distance between two nodes considering only minimal paths.

δ: Average distance between the nodes of the network. Again, only minimal paths are considered.

3.2 Bisection Bandwidth

According to [5], in networks with uniform channel bandwidth, as the one studied here, the bisection bandwidth is proportional to the channel count of the minimum cut of the network. For an $n \times n$ SpiNNaker topology, the minimum cut of the network in two halves is the one that divides it in two adjacent rectangles; note that both the horizontal and vertical cuts lead to the same scenario. In both cases, the cut crosses $4 \cdot n$ links ($2 \cdot n$ internal and $2 \cdot n$ peripheral links). This means that at most $4 \cdot n$ packets can traverse these links per cycle in each direction. Under random uniform traffic, the $N/2$ nodes located in one half of the network generate packets with probability 0.5 to the other half; therefore a maximum of $\frac{4 \cdot n}{N/4} = \frac{16 \cdot n}{N}\ packets/cycle$ can pass through the bisection. The B_B puts an upper bound on the maximum injection rate per node under uniform traffic assumption. Thus, the maximum theoretical throughput is:

$$\Theta = \frac{16 \cdot n}{n^2} = \frac{16}{n}\ packets/cycle/node \cdot$$

Recall that the link bandwidth was selected as to keep communication latencies below the biological real-time constraint of 1ms, even in peaks of network utilization. Since the link bandwidth is 1 Gbps, this leads to:

$$\Theta = \frac{16}{n}\ Gbps/node \cdot$$

However, the network is not expected to reach this theoretical limit. Research in neuroscience provides an estimation of average neuron firing rates of 10Hz in active populations [6]. Thereby, for SpiNNaker, we can estimate the average network usage required by each node as:

$$10Hz \cdot 20cores \cdot 1000neurons \cdot 40bits = 8Mbps\ .$$

For the largest configuration of SpiNNaker (256×256), we obtain a limit of the theoretical throughput

$$\Theta = \frac{16}{n} = \frac{16}{256} = \frac{1}{16} \approx 62Mbps\ ,$$

which indicates that the system is expected to operate below 15% of network capacity. Therefore no emphasis should be put on (nor conclusions extracted from) the behavior of the IN under saturation.

3.3 Distance-Related Characteristics

The average and maximum distance between nodes in a system have a definite impact on the latency of packets that travel along the network. For this reason, designers try to minimize these properties. Note that the SpiNNaker topology is *vertex-symmetric* and therefore all nodes have identical view of the network. Consequently, the routing space of *any* node in this topology is the same and can be seen as a set of hexagons centered in the node, plus two symmetric sets of triangles. Fig. 4 depicts an example of the routing space for a node in an 8×8 topology; hexagons represent those nodes at distance 1, 2, 3 and 4. The nodes forming the two symmetric triangles are at distance 5. To simplify the computation of the distance-related characteristics we consider a single node.

In general, the maximum network distance is the number of hops needed to reach any node in the two smallest triangles and can be computed as:

$$D = \frac{n}{2} + \left\lfloor \frac{n}{6} \right\rfloor = \left\lfloor \frac{2 \cdot n}{3} \right\rfloor \cdot$$

Note that $6 \cdot i$ nodes are located in the hexagon at distance i when i belongs to $\left[1,\ \frac{n}{2}\right]$. However, when n is even, the edges of the outmost hexagon can be reached going either through the positive or the negative dimensions. Thus, there are three nodes at distance $\frac{n}{2}$ that are counted twice. Finally for distance i in $\left[\frac{n}{2}+1,\ \ D\right]$, the two symmetric triangles at distance i contain $9 \cdot \left(\left\lfloor \frac{n}{6} \right\rfloor - \left(i - \frac{n}{2}\right)\right) + 3 \cdot \left(\frac{n}{2} \bmod 3\right)$ nodes each. When n is multiple of six the last triangle contains a single node despite this expression returning zero. Thus, the average distance is calculated as:

$$\delta = \frac{\sum_{i=1}^{\frac{n}{2}} \left(6 \cdot i^2\right) - 3 \cdot \frac{n}{2} + \sum_{i=1}^{\left\lfloor \frac{n}{6} \right\rfloor} 2 \cdot \left(\frac{n}{2} + i\right) \cdot \max\left(1, 9 \cdot \left(\left\lfloor \frac{n}{6} \right\rfloor - i\right) + 3 \cdot \left(\frac{n}{2} \bmod 3\right)\right)}{n \cdot (n-1)}\ .$$

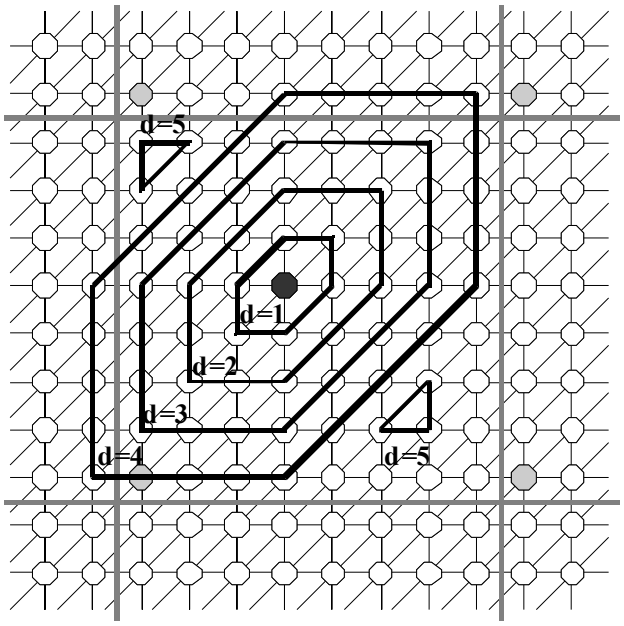

Figure 4. Example of a routing space for a SpiNNaker node – represented by a dark dot – in an 8×8 topology (tessellated). Light grey dots represent the origin of coordinates (0, 0).

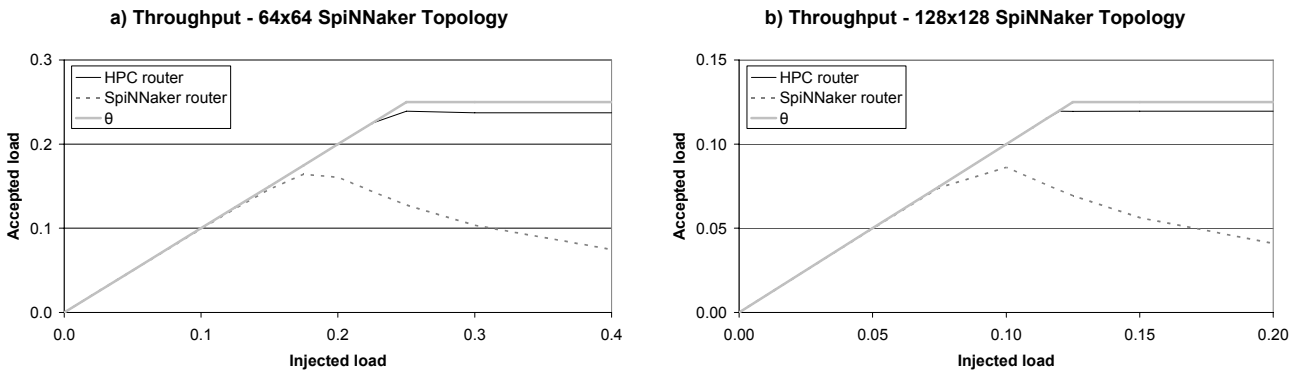

Figure 5. Throughput under uniform traffic of the SpiNNaker topology. Injected and accepted load (in packets/cycle/node). Plots for the SpiNNaker router, for an HPC router and for the theoretical maximum throughput. a) 64×64. b) 128×128.

4. EXPERIMENTAL WORK

4.1 Model of the System

A detailed model of the SpiNNaker IN was implemented in INSEE [23], a time-driven simulator that has been previously used in several high performance computing environment studies [4, 16, 17]. Note that SpiNNaker has a completely asynchronous design, so the use of a time-driven approach introduces a very fine-grain time discretization, with little influence on the results. The computational model includes most of the features of the router, and also the topological description of the system. However, to be able to execute simulations of large-scale systems, some modeling simplifications are needed. We model a cycle as the time to route and forward a packet (1 cycle corresponds approximately to 10 processor cycles). Since routing is faster than transmission, we allow the router to process several packets in a single cycle, provided that all the involved input and output ports are different.

As the routing tables need to be configured differently for each biological neural network, to avoid tying the evaluation to any particular application, the table-based routing is not used (which also reduces the computing resources required to perform simulations). As the regular routes between chips in the actual system will attempt to use a minimal path with a single inflection point, packets are sent through minimal routes using Dimension Order Routing (DOR) which emulates the expected shape of actual communications in SpiNNaker. Note that, when applying DOR, diagonal links are considered a third dimension (Z), thus the routes followed by packets are always XY, XZ or YZ; note that XYZ is not a minimal path.

The system is evaluated under point-to-point traffic only; the multi-cast engine is not used. Nonetheless, the traffic used disregards the inherent locality of communication in biological system [6, 10] and examines injection loads above the expected limit of 15% of the network capacity. Furthermore we want to remark that, while DOR is unaware of network failures, SpiNNaker is aware of these failures and can modify the routing tables to avoid sending packets to areas pinpointed as conflictive. Hence the results of the experimental study should be taken as worst-case results.

The nodes are modeled as independent traffic sources that inject packets following a Poisson temporal distribution, in which the injection rate (packets/cycle/node) can be tuned to any desired value. Furthermore, as all the ports from the cores inside a chip are merged, we model all of them as a single injection queue with room for four packets. If this queue is full and a core tries to inject, packets will be dropped.

4.2 Validation of Topological Characteristics

Using the described model of SpiNNaker, we simulate different sizes of the system ranging from 32×32 (1024 nodes) to 256×256 (65536 nodes). To allow the validation of the analytical study performed in Section 3, the interconnection networks are fed with uniform traffic.

Fig. 5 shows classical throughput graphs for networks of 64×64 and 128×128 nodes. Note that the threshold of 8Mbps corresponds to an injection load of 0.01 packets/cycle/node. The throughput values of the SpiNNaker router are almost indistinguishable from the calculated theoretical limit up to an injection load of 0.12 in Fig. 5a) and of 0.07 in Fig. 5b). In other words, the obtained throughput follows the theoretical limit for injection loads up to 7 times the expected load. We can also see that the throughput figures with higher injection rates cannot reach the theoretical value. Just to verify the analytical results, in the figures we also present the throughput results of the SpiNNaker topology but using a router specifically designed for HPC scenarios (IBM BlueGene/L torus network router [3, 21]). This kind of router is normally off-chip and incorporates mechanisms such as multiple virtual channels per physical link to increase throughput and prevent deadlock, and a congestion-control technique based on the prioritization of in-transit traffic. It does not implement, though, the emergency routing mechanism, nor is it viable for a SpiNNaker chip due to chip area constraints.

Table 1. Computed (comp.) and measured (meas.) average and maximum distances for different network sizes.

network size	comp. δ	meas. δ	comp. D	meas. D
32×32	12.4516	12.4527	21	21
64×64	24.8923	24.8878	42	42
128×128	49.7795	49.7824	85	85
256×256	99.5564	99.5576	170	170

Table 1 compares the analytically derived (*comp.*) and measured (*meas.*) average distance and diameter for the different networks. Note that, to accurately measure the distance-related characteristics, when an emergency route (two hops) is required, this counts as a single hop. We can see that analytically derived and measured diameters are the same. For the average distance, computed and measured values are equal up to the second decimal.

4.3 Optimization of Timeout Parameters

We evaluate the largest configuration of SpiNNaker (256×256) under a wide range of injection rates from 0.001 to 0.068 packets/cycle/node, which roughly represent 1.6% and 109% of the theoretical maximum throughput. This provides a picture of the behavior of the system under different levels of communication requirements. Note that most of the simulated scenarios are noticeably above the expected utilization of the IN. With this wide range we can study network behavior under utilization peaks. This study considers uniform distribution of packet destinations, although the actual system is expected to use optimized mapping of the neurons keeping communicating neurons in close proximity [14].

Different values of the *waiting time* (from 0 to 8) are tested to elucidate an optimal value for the actual system. Note that zero-waiting means that, if a packet cannot be transmitted, it tries the emergency route and, if it is also unavailable, the packet is dropped immediately. In the rest of the cases, the emergency route is tested in the last half of the *waiting time*. The figures of merit are the ratio of dropped packets, *i.e.* the amount of dropped packets normalized to the number of injected packets, and the injection rate at which each configuration is forced to drop packets. Note that maximum latency figures help to select a good value for the age-based packet dropping mechanism to avoid livelock in the actual system. Although not mandatory, it is preferable to keep latency low.

Furthermore we study the system under different degrees of network failures. The experiments are repeated in systems with one, two and 64 random link failures. The reason to test with one and two failures is that SpiNNaker can adapt to avoid the non-working components of the system and, thus, only a small amount of failures are expected to occur at once. Moreover, if we consider a pessimistic scenario of mean time between failures of 5 years with a sigma of 2 years, the region of interest is [30..60] failures. Given this interval and to test a worst case configuration, we tested a system with 64 random link failures.

Fig. 6a) shows the ratio of dropped packets in systems without failures. The lines disappearing at the bottom of the graphs are those with this ratio equal to zero; *i.e.*, no packet lost. Note that the higher the *waiting time*, the higher the load the network is able to manage without dropping packets. In contrast, the lower values lead to higher dropped ratios as soon as the network becomes saturated. This may lead into thinking that the higher the *waiting time*, the better the performance.

However, when looking at scenarios with link failures—Fig. 6b), c) and d)—the picture changes drastically. Those configurations with high waiting times start dropping packets before those with medium values, and reach unacceptably high dropped ratios

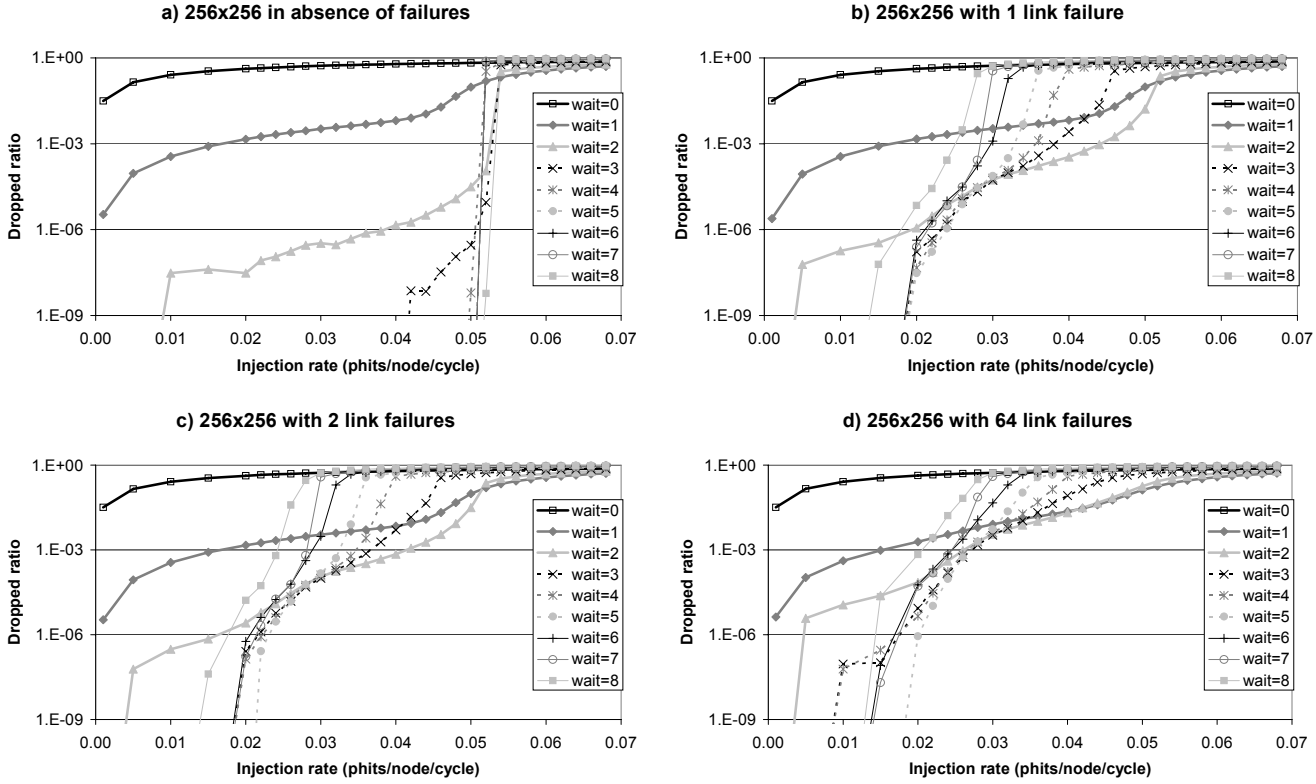

Figure 6. Packet dropped ratio for the different values of waiting time and number of link failures. a) Failure-free system. b) System with 1 failure. c) System with 2 failures. d) System with 64 failures.

Table 2. Maximum latencies measured for different values of the *waiting time* and different number of network failures.

configuration	wait=0	wait=1	wait=2	wait=3	wait=4	wait=5	wait=6	wait=7	wait=8
0 failures	174	373	790	890	1353	1411	1809	1975	2226
1 failure	174	373	789	888	1352	1411	1811	1974	2227
2 failures	174	373	789	889	1353	1409	1812	1973	2227
64 failures	174	371	787	891	1350	1415	1807	1968	2222

faster. This occurs because long waiting times generate congestion zones around the faulty links, which spread along the whole network. The clearest example is *wait=8*, which, in all the faulty scenarios, starts dropping packets noticeably before *wait* values 4 to 7. Note also how the lower the waiting time, the lesser the effect failures have on the dropped ratio. For *wait=0* or *wait=1* we observe the same behavior independently of the number of failures. In these cases, even the slightest contention for the use of the resources leads to packet dropping. Therefore, congestion zones are neither formed nor spread. Obviously, the penalty to pay is a higher dropping ratio that surpasses the acceptable threshold. The overall best performer *waiting time* is *wait=5*. Nevertheless we have to keep in mind that, in scenarios with high network pressure, unlikely to occur, it could perform poorer than other smaller timeout values.

The experiments also search for the optimal length of an *age* in terms of cycles, a requirement for a properly working livelock avoidance mechanism. All packets being in the network for more than two ages will be dropped. An age duration that allows dropping outdated packets as soon as possible, but without dropping useful, slowly-advancing packets is desirable. Age duration could be fixed to the maximum packet latency value obtained via simulation, which depended on the *waiting time*. As ages are global to the whole network, a packet that is injected in the last cycle of an age will be tagged with that age and, therefore, is under the risk of being dropped as soon as the next age finishes, so it will only have one age length plus one cycle to be delivered. Selecting a lower value of age length may lead to unnecessary packet dropping. For example, in the case of *wait=1* an age length of 373 cycles would be a good choice. Nonetheless, an outdated packet may wander around the network for up to 746 cycles. Table 2 summarizes the measured maximum latencies, for the different *waiting time* values and number of link failures.

4.4 Stability of SpiNNaker

Due to the real-time nature of SpiNNaker, our next set of experiments focuses on *stability*, understood as *low* variability of performance indicators as time evolves. Another distinguishing feature of SpiNNaker is the emergency routing mechanism, and part of these experiments focus on assessing its contribution to keep the system stable.

Experiments start with a fully functioning system, fed with uniform traffic at 0.02 packets/cycle/node load (~32% of network throughput, more than twice the expected worst-case scenario). An increasing amount of failures is introduced every 5K network cycles, simulating a system that degrades progressively, from 0 to 1024 link failures. At the beginning of every 5K-cycle block failures are introduced at once. Performance metrics are measured at intervals of 10 network cycles. In the graphs we plot accepted load (packets/cycle/node), number of dropped packets, and packet latency (average and maximum). We have fixed *waiting time* to 5 as suggested by the experiments reported in the previous subsection. To better understand the impact of emergency routing in system stability, we plot the evolution of three different systems: a 256×256 SpiNNaker configuration with this feature deactivated, SpiNNaker chips arranged as a 64×32×32 3D torus which does not allow emergency routing, and the actual SpiNNaker using emergency routing.

Fig. 7 plots all the obtained results. Notice that the horizontal axis shows simulation clock (in cycles). The labels on the top (1, 2, 4, …, 1024) indicate the total number of failures at the corresponding interval: during the first 5K cycles the network was fully operative, from 5K to 10K there was a single link failure, from 10K to 15K there were two failures, and so on. Each performance metric has its own unit, indicated in the vertical axes: packets (for the dropped packets line; on the left axis), cycles (for the latency related figures; left axis) and packets/cycle/node (for the accepted load line; right axis).

Figures 7a) and b) show how the progressive introduction of failures in the two systems without emergency routing resulted in a high variability of the performance metrics. This effect is reduced, but still very significant, for the 3D topology. Remember that these are SpiNNaker chips arranged in a 3D torus topology, but that are not using emergency routing because the on-chip routers do not implement it due to chip area constraints. If we look at results with emergency routing—Fig. 7c)—the system performs very stably.

Focusing on the amount of dropped packets—which is the key figure of merit, as its value *must* be kept low—we can see that the systems without emergency routing start dropping packets as soon as a single link fails. In the most extreme scenario (1024 failures) the SpiNNaker topology without emergency routing dropped roughly *25%* of packets, the 3D torus dropped *8%* of packets and SpiNNaker *with* emergency routing dropped only *0.2%* of packets.

The conclusion of these experiments is that SpiNNaker has a highly stable network for the real-time simulation of spiking neurons, even under very pessimistic scenarios. The system does not show significant performance fluctuations, and degrades gracefully. Furthermore we have showed the potential of the emergency routing mechanism to keep the system operating stably.

5. RELATED WORK

Research in simulating biologically plausible neural networks (brain-like systems) is not new and has remained a hot topic for the last decades.

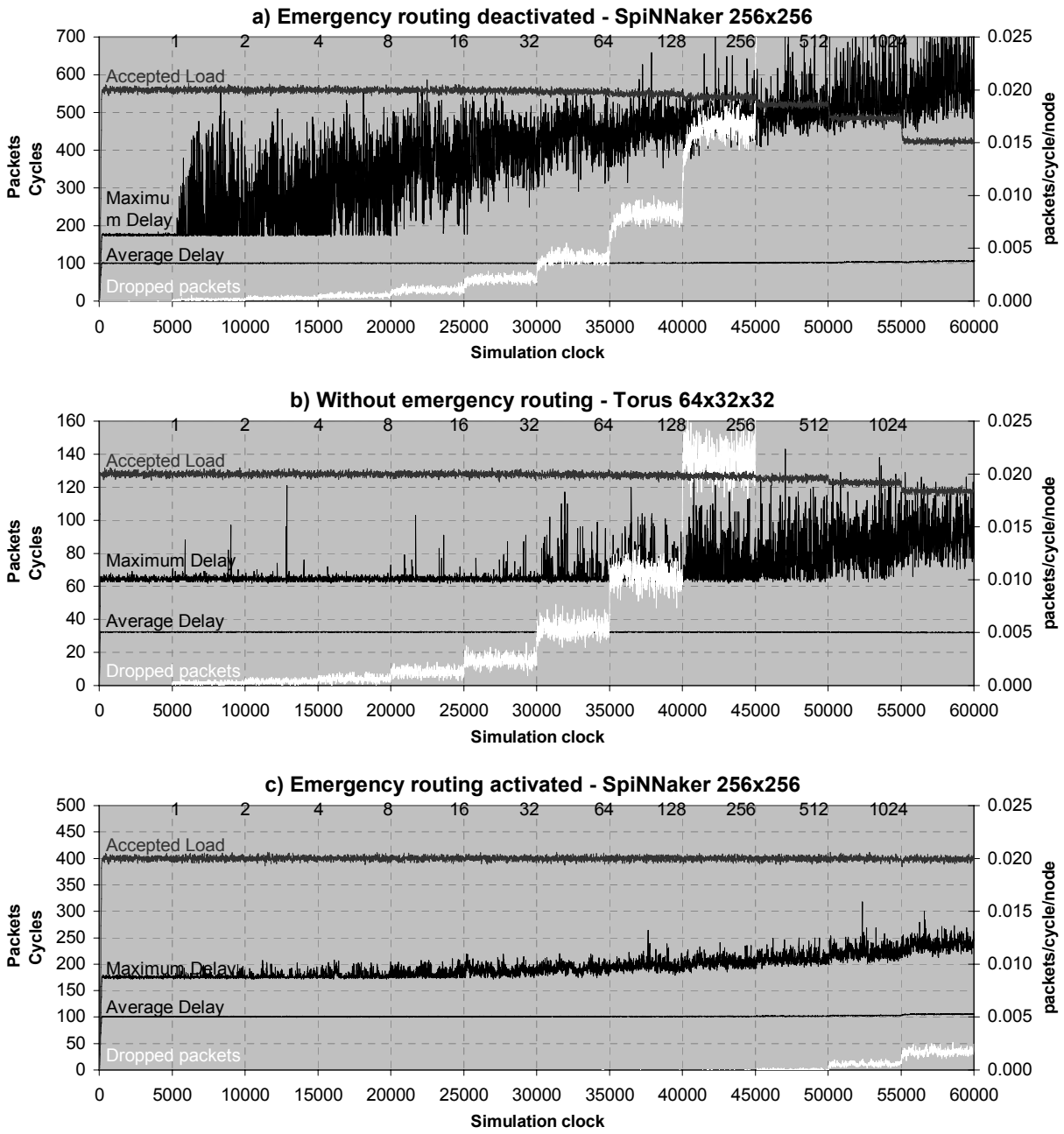

Figure 7. Evolution of the different systems under uniform traffic at a given load of 0.02 packets/node/cycle. a) SpiNNaker without emergency routing. b) Torus 3D, emergency routing not allowed. c) SpiNNaker with emergency routing.

In the early nineties a team at U.C. Berkeley worked in the Connectionist Network Supercomputer [1]. This project aimed to build a supercomputer specifically tailored for neural computation as a tool for connectionist research. The system was designed to be implemented as a 2D mesh, with a target size of 128 nodes (scalable to 512). Each node would incorporate a general-purpose RISC processor plus a vector coprocessor, 16MB of RAM and a router. To our knowledge, a prototype of the node was built (under the codename T0), but the system never operated as a network. Experiments using up to five nodes in a bus configuration were discussed in [18].

More recently, the Microelectronics Division at the T.U. of Berlin worked in a project [15] whose objectives were similar to those of SpiNNaker. Part of this project is an acceleration board, called SSE, implemented with a collection of FPGAs interconnected via an on-board bus. An SEE accelerator is able to perform neural computations 30 times faster than a desktop PC [12]. Other projects used FPGAs for similar purposes, obtaining speedups of up to 50 compared to software-only implementations. However, as these boards cannot be connected to form a network, they are not able to scale to the magnitudes of SpiNNaker.

As far as we know, the only active project comparable to SpiNNaker in terms of simulation scale is the Blue-Brain project [2] which aims to create a biologically accurate functional model of the brain. However, the high complexity of its neuronal model does not allow it to work in real-time. Furthermore, in contrast with the biologically-inspired SpiNNaker architecture, the BlueBrain project does not contemplate the construction of any specific computing system but uses a general-purpose supercomputer, the IBM BlueGene [3].

The SpiNNaker emergency routing mechanism has been shown to be very effective for fault tolerance. Implementing fault tolerance in HPC interconnection networks (such as the well-known 3D torus) is a hot research topic but current solutions are neither easy nor cheap to implement in silicon; see for example [11, 22].

6. CONCLUSIONS AND FUTURE WORK

This paper has studied the IN of SpiNNaker, a biologically-inspired massively parallel system of bespoke multi-core SoC designed with the aim of simulating up to a billion spiking neurons in (biological) real-time. To be a robust system the SpiNNaker architecture relies on redundancy both in terms of computing and communicating elements. SpiNNaker chips are interconnected using a 2D toroidal triangular mesh. Neurons are modeled in software and their spikes generate packets that propagate through the on- and inter-chip communication fabric relying on the specifically-designed on-chip multicast routers.

Through simulation, we have examined the temporal evolution of the system in order to test the stability of SpiNNaker under worst-case scenarios and with high levels of degradation due to faults. Three different networks have been tested: the actual network with and without the emergency routing activated and a 3D torus which does not allow the use of emergency routing due to hardware constraints. This study has showed that SpiNNaker has a highly stable network for the real-time simulation of spiking neurons, even under very pessimistic scenarios. However the two systems without emergency routing are not able to keep operation stable. For example using the largest configuration (over 65 thousand nodes) and 1024 failed links, the SpiNNaker topology without emergency routing dropped roughly 25% of packets, the 3D torus dropped 8% of packets and SpiNNaker with emergency routing dropped only 0.2% of packets.

An analytical evaluation of the system IN has been carried out and validated via simulation. The analysis has obtained expressions to compute the topological characteristics of the network: theoretical throughput and distance-related properties. Furthermore, the experimental study has allowed the selection of optimal values for the timeout mechanism to avoid deadlock. The focus to select these parameters has been on reducing packet dropping ratios and increasing the injection rate at which the system starts dropping packets. Moreover, the measured values of maximum latencies have been provided in order to help in the selection of the appropriate values for the age-based packet dropping mechanism, implemented to avoid livelock. Our results lead to the conclusion that keeping in-transit packets waiting for too long for the allocation of output ports is counterproductive. This contention results in a backpressure that causes the dropping of packets at the injection queues. In most of the experiments, a waiting time of five cycles provides the best balance between the number of dropped packets at any injection rate and the injection rate that started losing packets, both in the properly-working scenario and in scenarios with link failures. However, note that our simulated routing model was unaware of the network failures. On the other hand the actual SpiNNaker is aware of these failures and would route the packets through trusted paths. Thus, a lower degree of system degradation is expected.

As future work we expect to perform more evaluations of the system with different failure models (*e.g.*, bisected system, small areas with high density of failures, etc.) and different traffic models both in terms of the spatial distribution of the communications using distance distributions that favor short distance communication and in terms of temporal and causal relationships among packets (spikes).

7. ACKNOWLEDGMENTS

The Spinnaker project is supported by the Engineering and Physical Sciences Research Council, through Grants EP/D07908X/1 and GR/S61270/01, and also by ARM and Silistix. Steve Furber holds a Royal Society-Wolfson Research Merit Award. Authors from the University of the Basque Country are supported by the Spanish Ministry of Education and Science, grant TIN2007-68023-C02-02, and by Basque Government grant IT-242-07. Javier Navaridas is supported by a doctoral grant of the UPV/EHU.

8. REFERENCES

[1] K Asanovic, et al. “A supercomputer for neural computation.” In Proc. 1994 Intl. Conf. on Neural Networks (ICNN94).

[2] BlueBrain project. Available (January 2009) at: http://bluebrain.epfl.ch/.

[3] M Blumrich, et al. “Design and Analysis of the BlueGene/L Torus Interconnection Network” IBM Research Report RC23025 Dec. 2003.

[4] JM. Camara et al. "Mixed-radix Twisted Torus Interconnection Networks". Proc. 21st IEEE International Parallel & Distributed Processing Symposium - IPDPS '07, Long Beach, CA, March 26-30, 2007.

[5] WJ Dally and B Towles, “Principles and Practices of Interconnection Networks”, Morgan Kaufmann Series in Computer Architecture and Design, 2004.

[6] P Dayan and L Abbott, “Theoretical Neuroscience”. Cambridge: MIT Press, 2001.

[7] JJ Dongarra, HW Meuer, E Strohmaier. “Top500 Supercomputer sites”. Nov. 2008 edition. Available at: http://www.top500.org/

[8] T Elliott and N Shadbolt, “Developmental robotics: Manifesto and application,” Philosophical Trans. Royal Soc., vol. A, no. 361, 2003.

[9] S Furber, S Temple, and A Brown, “On-chip and inter-chip networks for modelling large-scale neural systems,” in Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece, May 2006.

[10] S Furber, S Temple, “Neural Systems Engineering”. Journal of The Royal Society Interface 4(13), pp 193-206, April 2007

[11] ME Gomez, et al. “A routing methodology for achieving fault tolerance in direct networks”. IEEE Transactions on Computers, 55(4), 2006.

[12] HH Hellmich, et al. “Emulation engine for spiking neurons and adaptive synaptic weights”. In Proc. IEEE International Joint Conference on Neural Networks (IJCNN), 2005.

[13] X Jin, SB Furber, and JV Woods. “Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor”. In Proc. of the International Joint Conference on Neural Networks, 2008.

[14] MM Khan et al. "SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor". Proc. 2008 International Joint Conference on Neural Networks (IJCNN2008).

[15] Microelectronics Division T.U. of Berlin. “Design and implementation of spiking neural networks.” Available (January 2009) at: http://mikro.ee.tuberlin.de/spinn.

[16] J. Miguel-Alonso, C. Izu, J.A. Gregorio. "Improving the Performance of Large Interconnection Networks using Congestion-Control Mechanisms". Performance Evaluation 65 (2008) 203–211.

[17] J Navaridas et al. "Reducing Complexity in Tree-like Computer Interconnection Networks". Technical report EHU-KAT-IK-06-07. Department of Computer Architecture and Technology, the University of the Basque Country. Submitted to Elsevier’s Journal on Parallel Computing.

[18] P Pfaerber and K Asanovic. “Parallel neural network training on multispert”. In Proc. IEEE Third International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP’97), 1997.

[19] LA Plana et al. “A GALS Infrastructure for a Massively Parallel Multiprocessor”. IEEE Design & Test of Computers, Volume: 24 , Issue: 5, pp. 454 - 463, Sept.-Oct. 2007

[20] LA Plana et al. “An on-chip and inter-chip communications network for the spinnaker massively-parallel neural net simulator”. Proc. Second ACM/IEEE Intl. Symposium on Networks-on-Chip (NoCS 2008), 2008, pp. 215 – 216.

[21] V Puente, et al. “The Adaptive Bubble router”, Journal on Parallel and Distributed Computing, vol 61, Sept. 2001.

[22] V Puente, JA Gregorio. “Immucube: Scalable fault-tolerant routing for k-ary n-cube networks”. IEEE Transactions on Parallel and Distributed Systems, 18(6), 2007.

[23] FJ Ridruejo, J Miguel-Alonso. “INSEE: an Interconnection Network Simulation and Evaluation Environment”. Lecture Notes in Computer Science, Volume 3648 / 2005 (Proc. Euro-Par 2005).