Scalable NoC-based architecture of neural coding for new efficient associative memories

J-Ph. Diguet*, M. Strum†, N. Le Griguer*, L. Caetano* and M. J. Sepúlveda *†

* CNRS, UMR6285, Lab-STICC, Univ. Bretagne Sud, 56100 Lorient, France, email: jean-philippe.diguet@univ-ubs.fr
Microelectronics Lab.-EPUSP, Univ. of São Paulo, Brazil, email: jsepulveda@lme.usp.br

April, 2013

Abstract

We present the first NoC-based hardware implementation of Neural Coding (NC), which is a new approach that opens outstanding perspectives for the design of associative memories and learning machines. We first propose optimized architectures of memories and processing elements that allow for an efficient distributed implementation. Then we introduce different NoC architectures to interconnect all elements, it provides the required scalability and takes advantage of parallel transfer opportunities. Performance, cost and energy consumption tradeoffs of various NoC solutions are compared and discussed. Based on previous implementation results, we run SystemC-TLM that validate the behavior of the algorithm and of the efficiency of the dedicated architecture. This work demonstrates that this architecture can meet expected requirements in terms of scalability and hierarchy, and consequently that NC-based architectures are compliant with efficient hardware implementations of a new and promising model of associative memories.

1 Introduction

Neural Coding (NC) [4] is a new concept that associates neural network and information theory (error correcting codes). It is a very promising solution that pave the way to new kinds of associative memories, classifiers, data-mining or search engines that is to say systems with the ability to give access to an information given a partial knowledge of it. Basically NC is a learning system that integrates two phases: message learning and repairing that corresponds to information retrieval from partial knowledge. It relies on two main ideas: sparse coding and clique-based codewords that allow for impressive improvements compared to Hopfield-like neural network (HNN) in terms of capacity, efficiency and diversity. The *capacity* is the total number of information (i.e. nb of bits) a system can learn. The *efficiency* is the ratio of the amount of bits learned to the amount of bits used, which is close to optimal (optimal efficiency=1). Finally the *diversity* is defined as the total number of messages a network can learn as opposed to the capacity, it is a key point that distinguishes this approach to HNN. It makes a decisive difference since contrary to HNN, Neural Coding doesn't require the length of learnt messages to be equal to the number of neurons. This leads to the important property of *sparsity*, which means that NC requires a very limited number of neurons distributed over the NC network. A crucial innovation is the way these neurons are linked with a code that improves the resilience capacity that characterize the human neocortex. This code is basically a clique, namely a concatenation of simple codes comparable to Low-density parity-check codes (LDPC).

The improvement of diversity depends on network parameters and configurations but is typically in the range of 2 to 3 orders of magnitude. It opens new perspectives, for the design of efficient associative memories, that were not realistic with HNN-based solutions [6]. Learning and repairing processes are also simplified for implementation designers who are also greatly helped by the available formalization of the probability

of errors. An error is the retrieval of wrong data and the associated probability is given as a function of the number of learnt messages, the ratio of missing data and the NC sizing parameters (l, c detailed in Section 2).

Very few works have addressed the hardware implementation of NC. The only published work on that topic [6] identifies the interconnection architecture and memory access as the main performance and cost bottlenecks. In this study we address these issues and propose the first implementation of NC that takes advantage of the NoC concept to efficiently carry out communications with dedicated distributed memories and processing elements. The use of a NoC is essential to get the expected flexibility, it is actually necessary to have the possibility to configure a unique chip with parameters that fit application requirements.

This paper is organized as follows. In Section 2 we describe the NC algorithm, Section 3 details related studies in the domain of NC, neural network NoC and associative memories. Section 4 presents the different optimizations that have been applied to efficiently implement the NC algorithm and the architecture of the different components. Section 5 details traffic features and requirements and the associated optimized NoC architecture. In Section 6 we give implementation results and Section 7 presents the SystemC simulations that validate the algorithm and the architecture. Finally we draw perspectives and we conclude.

2 Neural Coding Algorithm

The neural network is based on c clusters, each cluster is composed of l fanals (that would be called neurons with the traditional HNN formulation), $n = l \cdot c$ is the total number of fanals. In practice c and l are chosen as power of 2 in order to simplify equations and implementations. A fanal could be associated to a neuron but the authors claim that a more plausible biological analogy with the neocortex would lead to consider a fanal as a micro-column of neurons, a cluster as a column and a NC network as a macro-column of the neo-cortex. For the sake of clarity we will use neurons and clusters in the following. In Fig. 1 the network example is such that $l = 8, c = 4, n = 32$ and 3 bits are required to code each neuron.

A message M is composed of c symbols (a symbol per cluster). The first step of the learning process is the mapping function that associates each symbol to a single neuron in each cluster. Thus each symbol is coded with $k = log_2(l)$ bits and a message is a binary sequence of $M \cdot k$ bits. This mapping illustrates the property of sparsity, that participates to confer diversity and resilience properties to NC. In a second step, each learnt message is associated to a clique of neurons, each clique materializes a codeword. The minimum distance d_{min} between two codewords is equal to $2(c-1)$, the coding rate[1] is such that $R = 1/(c-1)$ so the factor of merit $F = R \cdot d_{min} = 2$ is greater than 1, which means that it is a true error-correcting code (see [4] for details).

The repairing or retrieving process is based on a double-step algorithm. The first local step consists in the selection of the best candidate neuron in each cluster where an error occurs. This choice is made according to a ranking method, which is a winner-take-all algorithm in the first proposal of the NC authors. All clique edges are specified with binary weights such as $C_{XY}(ij) = 1$ means that at least one clique contains an edge from neuron i in cluster X to neuron j in cluster Y. The score of each neuron, in the target cluster, equals the number of clique edges that exist between this neuron and known neurons in other clusters. Fig. 1 presents an example where the partial message to be completed is: "001-110-xxx-101", the error occurs in cluster C, while known neurons in clusters A, B and D are "001", "110" and "101", respectively. In cluster C, neurons "001" and "110" obtain scores of 3 and 1, respectively, thus "001" is chosen as the best solution. The second step is the global decoding that gathers identified winners form all clusters through a message passing principle. In case of multiple errors, it may happen that some of winners cannot be decided, it means that some errors cannot be fixed, in such cases multiple iterations are necessary to converge. Simulations show that in practice the required number of iterations varies from 1 to 4, however most of errors can be solved with a single iteration.

It may happen that some errors cannot be solved but an interesting aspect of the model based on information theory, is that network parameters can be tuned in order to reach required performances. For

[1] Ratio between the minimum number of edges to specify a clique and the total number of edges

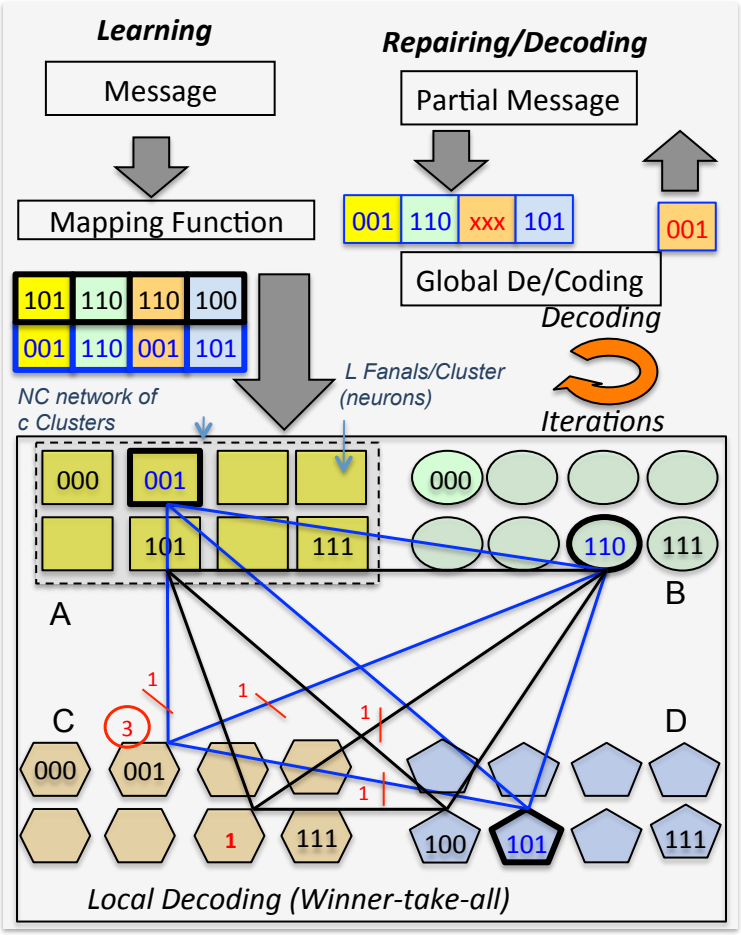


Figure 1: Neural Coding principle ({4,8,6} case)

instance n and c can be chosen according to a maximum threshold of missing information (e.g. 50%) and the expected error probability (e.g. 0.01). In the following, a NC system is noted as tuple: $\{c,l,m\}$, where c is the number of clusters, l is the number of fanals integrated at each clusters and m is the number of memories, e.g. Fig. 1 shows a {4,8,6} case.

3 State of the art

Standard content-addressable memories (CAM) provide current solutions for fast hardware associative memories, but at a cost of specific core cells that combine bit storage and comparison [7]. CAM-memories are adapted to fast lookup applications such as IP tables in network routers but are limited in terms of flexibility and scalability. Associative memories based on neural networks have been recently reconsidered as a possible solution to solve the increasing demand in content access applications. A lot of improvements have been proposed to increase the diversity of HNN. In [4] the authors explain that proposed alternative approaches such as [10] and the Bolzman machine [1] lead to complex learning rules and an increase of the number of neuron levels, moreover the main issue is the efficiency that remains low and tends to 0 when considering an infinite number of neurons while the NC approach is nearly optimal. To the best of our knowledge, [6] is the only published work about architecture for NC. This solution is mainly a proof of concept based on a straight-forward architecture with a point to point connection scheme. It includes a $\frac{c\cdot(c-1)}{l}{}^2 2$ weight

memory and is organized with c cluster modules, each module implements a flip-flop based memory of size $c \cdot (c-1)l^2$ bits with an OR-array including $3 \cdot l \cdot c$ k-bits adder/substractors for learning and repairing processes. Additional registers are also implemented to store sums required during the decoding process leading to $((c-1) \cdot l+1)c \cdot l$ bits of additional storage capacity. Our approach first optimizes both processing and memory aspects, it reaches the lower memory bound with parallelized memory accesses. Secondly it takes advantage of a NoC to obtain a fully distributed and scalable implementation.

Several hardware implementations of HNN have been proposed [2][11][8][3][12]. These works identified the system communication structure (CS) as the bottleneck to meet the system requirements. In order to handle the communication complexity of such systems bus and NoCs structures have been employed. The authors of [2] use bus-based CS to implement the data exchange among the different set of neurons. The broadcast property of the bus allows the improvement of the system performance. However, this solution is not scalable. Therefore, the overall system performance is degraded for HNN that integrates a large amount of neurons or that exchange large messages.

Several packet-based NoC architectures are proposed to support the tight communication requirements of HNN in [11][8]
[3][12]. These works take advantage of NoC highly parallel structure, scalability and reliability in order to meet the performance constraints and to reduce the cost of the system in terms of area and power. 2D mesh, torus and tree NoC topologies were explored in [11][8][12] to connect HNN up to 200 neurons. The authors of [12] have shown by means of analytical models, that the multicast provides the higher performance/cost ratio.

These previous works show the critical role of the CS in the HNN performances. However, the performance of such systems is highly dependent of the HNN application and the CS architecture must be tuned to achieve high performances whenever HNN application is changed.

4 NC Components & optimizations

Our architecture is composed of three types of optimized components connected to a dedicated network on chip: connection memory, winner-take-all processors (WTAP) and a global manager (GM). All components are specified in VHDL in a generic way so that c, l as well as NoC interface parameters can be decided at design time. In the following we consider a single manager per NC network, we will see later that this is a first step that provides a component to build hierarchical solutions, which fit with future complex and multi-message associative memories.

4.1 Connection memory

Memory cost is critical in NC as it is for associative memories and neural network implementations. So we came up with a solution that avoids any storage redundancy. It is applied as follows, a connection $v_x(i) - v_y(j)$ between neuron i in cluster x and neuron j in cluster y isn't stored twice in the two clusters but only once in a connection memory $M_{x,y}$ of l^2 bits (see Fig. 2), where $M_{x,y}(i,j) = 1$. It means that at least one connection exists between neuron i in cluster x and neuron j in cluster y. A memory point is binary and if multiple cliques use a common connection the value is equal to '1'. So if we consider c clusters, it means that $\frac{c \cdot (c-1)}{2}$ memories are implemented and connected to a NoC so that concurrent data transfers can be performed.

The second important point is the memory access time, we propose a solution based on l memory banks of l bits and a specific data mapping that allows parallel accesses. The principle is depicted in Fig. 2. In this example all connections from cluster C_0 to a specific neuron i in cluster C_1 can be retrieved as a row, from all memory banks in a cycle. The shifted mapping also authorizes to read concurrently all connections from cluster C_1 to a given neuron is cluster C_0 as diagonals with modulo indexing.

Then two methods can be applied. The first one consists in sending a full connection vector $V[0..l-1]$ where $V[i] = 1$ (resp. 0) means that at least one connection has been created (resp. no connection has ever been created). In this case data extraction is trivial, a full row or diagonal is transmitted to the WTAP that

processes data in order. In the second case only existing connections are transmitted, it means that a test has to be introduced within the memory reading step in order to detect '1' bits and that the location i (i.e. the address) of existing connections have to be transmitted. The best solution, in terms of hardware complexity, latency and traffic load, results from a tradeoff that depends on the cluster size. In this study we choose the first solution for the sake of implementation cost and discuss alternative architectures in perspectives.

The learning process is trivial, this is a simple set of write operations in connection memories. The GM sends the message to all memories that decode the address of the neuron they are in charge of and set the associated memory point.

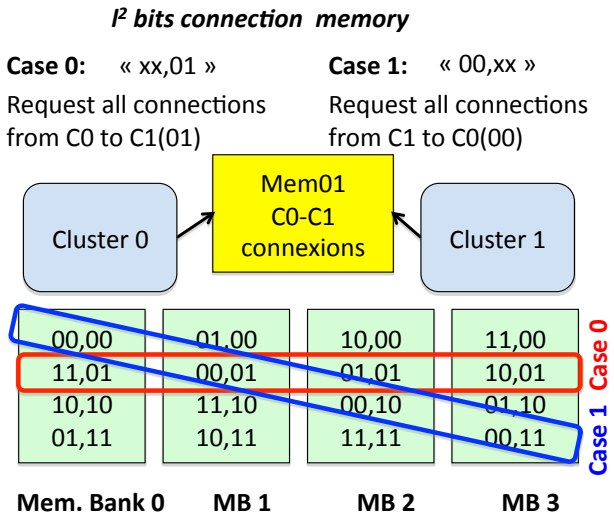


Figure 2: Connection memory

4.2 WTAP

A WTAP is in charge of the local decoding for a cluster where a neuron is missing. It means the reception of connection vectors sent by the connection memories, the update (+'1') of neuron scores and the sorting of the neurons in order to select the best candidate that will be transmitted to the GM. In case of a single error, $c-1$ memories will send connection vectors to the WTAP. A local memory is required to sort neuron candidates, the maximum best score is equal to $c-1$ so a memory of $l \cdot log_2(c-1)$ bits is implemented. A simple counter of length l is necessary to provide addresses of connections that arrive in order. In addition we have designed an optimized module dedicated to +1 addition and comparison with an output bitwidth of $l \cdot log_2(c-1)$ bits for score updates. A parallel sorting algorithm can be implemented if necessary, in that case all flit bits can be processed in parallel. In this first study a sequential implementation has been chosen to minimize the hardware cost. Each neuron is fully processed in a cycle (read, test, +1, write back / best score update), so l cycles are required to process data issued from each connection memory. Fig. 3 presents the WTAP architecture. Each WTAP must send back the best-scored neuron so that the GM can repair the initial message. Two schemes can be proposed, in the first one the WTAP sends back the new best score, in any, after processing each connection vector sent by a memory. In the second case, the best score is transmitted once all the connection memory have transmitted the connection vectors. The first case generates more short messages from WTAPs to the GM. The second case means more control since it requires that the GM indicates to each WTAP the exact number of errors, namely the number of connection vectors to be processed. In this work we have considered the first case so that each WTAP is fully independent from the GM with a very limited traffic overhead.

4.3 Global Manager

The GM is the interface of the NC system. It is a simple FSM structure that executes the mapping function and that transforms a learning or repairing request into a message of $c \cdot log_2(l)$ bits. The GM then sends messages to memories through a NoC interface. It also controls the iterations of the decoding process until the reception of the repaired message or the detection of an unsolvable ambiguity.

5 NC Network on chip

5.1 Communication Requirements

NoC turns an attractive alternative to implement the CS of this kind of systems. NoCs are scalable and perform parallel communication, allowing the integration of several processing and storage components without degrading the performance. NC traffic is composed by two main types of data flows: i) learning, to store the occurrence of connections between neurons from distinct clusters and forming neural cliques; and ii) repairing, to select from the cluster that presents an error the best neuron candidate by means of the WTAPs. Most of the execution time of the NC will be dedicated to the repairing process, first because reading are more frequent than writing in systems based on associative memories and secondly because the learning process is a simple write whereas a repairing operation requires the execution of a *winner-takes-all* algorithm and possibly multiple iterations. In the following section we present the communication requirements of the NC.

5.1.1 Learning flow

As explained in Section II, in the learning process, the incoming data M_{learn} represents the links between the c clusters of the system. For each message, a single neuron of each cluster is activated. Suppose that each cluster integrates l neurons, the number of bits b_{learn} of incoming M data is given by Eq. 1.

$$b_{learn} = log_2(l) \cdot c \tag{1}$$

When the manager receives the M_{learn}, it divides the incoming data and generates $(c(c-1)/2)$ packets, P_{Learn}. Each P_{learn} corresponds to the data that must be stored in a connection memory. Each memory stores the connections between two clusters. Therefore, the size S_{learn} of each P_{learn} depends on the number

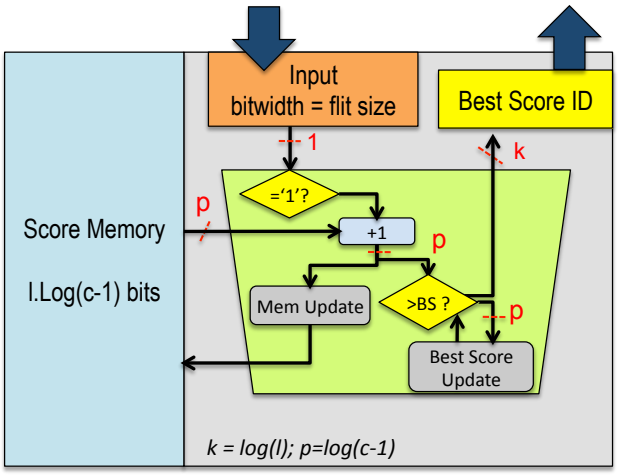


Figure 3: Dedicated and resource-optimized WTAP applying local decoding (sequential version)

of neurons l within each cluster, as shown in Eq. 2.

$$S_{learn} = log_2(l) \cdot 2 \tag{2}$$

The learn flow involves a set of writing transactions whose only initiator is the manager component and the destinations are the connection memories.

5.1.2 Repairing flow

In the repairing process, the incoming data M_{rep} represents the incomplete set of links between the c clusters of the system. The number of bits b_{rep} is the same that b_{learn}. An error represents an unknown neuron of the cluster in which the error is produced. As a result of the repairing process, the complete set of neurons is given. Therefore, with the partial known of the information, the complete word can be recovered. The communication of the repairing process can be divided in three stages:

From manager to connection memories Each time M_{rep} arrives, the manager identifies the error sections. It sends a set of packets P_{mame} to all the memories that store the link information of the cluster c_{rep} , which contains the unknown neuron memory. Suppose e, the number of errors at M_{rep}, the number of packets S_{mame} generated by the manager to the set memories is given by 3. P_{mame} contains the c_{rep} information and the M_{rep} section of the known connection between C_{rep} and the proper cluster indexed by the memory.

$$S_{mame} = (c-1) \cdot e \tag{3}$$

The size of each packet P_{mame} is given by $log_2(l) + 1$ bits. The extra bit is used to indicate the address of the c_{rep}. According to [4], 75% or errors can be considered as an acceptable maximum bound. Therefore, the number maximal of Pmame is given by Eq. 4.

$$P_{max} = (c-1) \cdot 0.75c \tag{4}$$

From memories to WTAP Memories which receive the data from the manager, retrieve the links information and send to the WTAP associated to cluster c_{rep} by means of one packet P_{mc}. The size of P_{mc} is given by l bits. The number of the packets generated by the memories depends of the number of errors in M_{rep}.

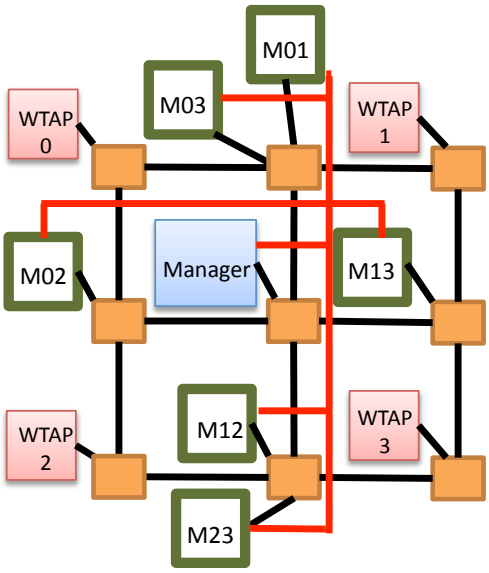
From WTAP to managers Each of the e WTAP is in charge of a local decoding, it waits the arrival of all the $(c-1)$ packets. After the processing procedure is finished, WTAP sends the packet P_{rep}, containing the best neuron candidate. The size of P_{rep} is given by $log_2(l)+1$, where the extra bit indicates the occurrence of an ambiguity. Ambiguity arises when several neurons have the same score, so there is no best single neuron, but a set of candidates. The occurrence of ambiguities requires extra repairing iterations.

The manager receives all the e packets P_{rep} from the WTAPs. The number of P_{rep} depends on the number of errors. If M_{rep} is repaired, the manager delivers the correct message $M_{solution}$, which size in bits is the same as b_{rep}. Otherwise, the manager evaluates if it is possible to find $M_{solution}$, by comparing the improvement of the current M_{rep} with the previous one. If it is possible, a new repairing cycle is initiated. If not, the manager will indicate that there is no possible solution.

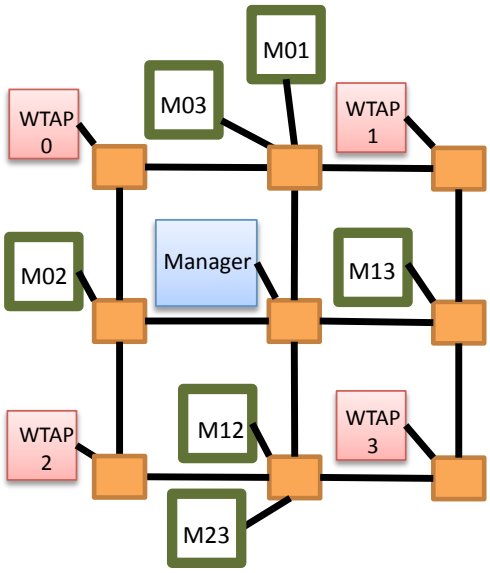
5.2 Communication Structure choices

In order to meet the communication requirements of the NC system, we propose 4 different architectures: i) Hybrid interconnection (HoC); ii) Regular mesh NoC ; iii) Semi-Torus; and iv) NoC 3D. Fig. 4 shows the proposed CS for a four cluster NC system.

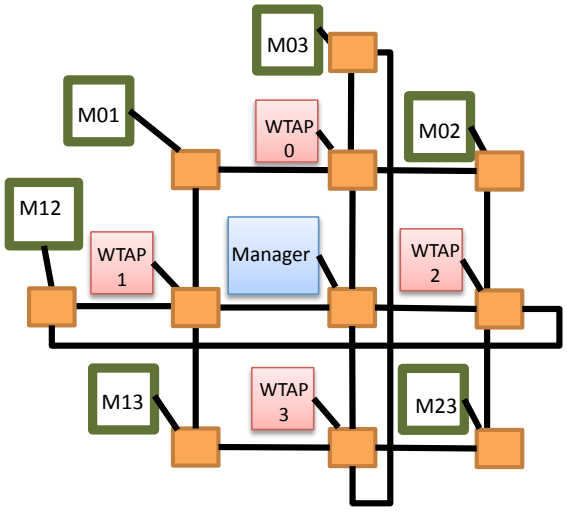
The objective is to create an optimized CS that minimizes the cost in terms of area and power while achieving the best response time. The mapping of the storage and processing components at the CS is made so that, they are equidistant from the manager and the hop distance between the memories and WTPAs is minimal.



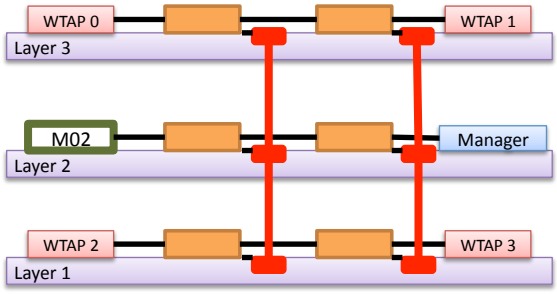
a) Hybrid interconnection



b) Regular mesh (unicast and multicast)



c) Semi-Torus (multicast and unicast)



d) NoC-3D

Figure 4: Target NoCs

5.2.1 Hybrid interconnection (HoC)

It combines buses and NoCs in the same communication structure. The bus is used to broadcast the information from the manager to the memories at the learning process and stage i of the repairing process. This architecture takes advantage of the fact that in these transactions, the only initiator is the manager. In such conditions the bus turns scalable. The NoC is used mainly to the repaired process at the stages ii and iii described in Section 2. HoC provides a high performance and low cost CS for NC systems. NoC routers implement wormhole switching, round-robin arbitration and XY routing algorithm. Bus and NoC links width are customized according to the size of the NC system.

5.2.2 Regular mesh (unicast and multicast)

It uses a mesh-based NoC to communicate all the NC components. The regular architecture allows the developing of high scalable systems. NoC characteristics are the same that the one employed by the Hoc architecture. There are two configuration possibilities: unicast or multicast. At the unicast configuration, the initiator sends the data to a single destination. At the multicast configuration, the initiator sends all the data in a single message to multiple destinations. The destination is in charge of filter the incoming data in order to obtain the parts of the message that are of its interest.

5.2.3 Semi-Torus (multicast and unicast)

It uses a partially torus architecture in order to decrease the hop distance between the different memories and the WTAP. NoC routers implement wormhole switching, round-robin arbitration and XYmin routing algorithm, which find the shortest path between a pair initiator-slave in the torus architecture. NoC links width depends on the size of the NC system.

5.2.4 NoC-3D

It uses buses and two dimensional NoCs (2D-NoCs) to implement the vertical and horizontal interconnection of the NC system. It can be implemented through the tree dimensional technology (3D). 3D promotes the vertical integration of several processing and storage components [5]. Our 3D-NCS implements a layered organization, that groups in each layer a single type of core, so that, storage components are stacked atop processing IPs [5]. 3D-NCS integrates three layers: 2 external layers of WTAPs (layer 1 and 3 in Fig. 4-D) and a memory layer (layer 2). 3D architecture is attractive for NC systems because it is highly scalable and increases the bandwidth between the memories and the WTAPs. The data flows through the NoC-3D as packets. 2D-NoCs uses a six-port router and an arbitrated bus spanning the entire vertical distance of the chip. NoCs and buses links widths depend on the size of the NC system.

6 Results

6.1 Experimental setup

We have developed a VHDL-RTL model of the NC system and of the six CS: i) Hybrid interconnection (HoC); ii) Regular mesh (unicast); iii) Regular mesh (multicast); iv) Semi-Torus (unicast); v) Semi-Torus (multicast); and vi) NoC 3D. Two NC systems were implemented: {4,4,6} (11 IPs) and {16,128,120} (137 IPs). The system was synthesized on a 65 nm Virtex 5 target to rapidly prototype the architecture. We assume that 25% of the execution time is dedicated to the learning process and 75% to the repairing process. At the repairing process, the percentage of errors presented in M_{rep} varies from 0% to 75%. Power and latency models from [9] were employed.

6.2 Performance results

Fig. 5, 6 and 7 show the results of the NC execution time for three different R = neurons/clusters ratios: i) R= 4; ii) R=8 ; and iii) R=16, respectively. The results show that the best configuration in terms of latency is the NoC-3D. It also presents the less dependency between the percentage of errors and the execution time. This is due to the short hop distance of this architecture. The multicasts versions of the regular mesh-NoC and the Semi-Torus outperform unicast versions. This is explained by the reduction of the time required for the packet creation. HoC systems result in interesting solutions for high R ratio and low percentage of error. This is because the broadcast of the buses, which achieves a fast communication between the manager and memories, present a significant advantage for larger packets. It's also important to keep in mind that the serial version of the WTAP is the component that dominates the execution time of the whole system; since once data are received, each processor must employ l cycles per cluster to obtain the result.

6.3 Cost results

Tables 1, 2 and 3 present the area and power results for the NC systems that use the different target CS. The NoC-3D provides the best performances but also the higher cost, it may be a solution for building hierarchical NCs and the other solutions offer close area/performance tradeoffs that can lead to different choices according to performances requirements. We observe that the main critical resources are obviously, as expected, the connection memories. The results show that the best implementation in terms of cost is the NoC regular mesh. A NC {4,4,6} can easily be implemented on small Virtex, but this is not the case for NC {16,128,120}, if the number of LUT can fit FPGA capacities, the number of single-bit registers if definitely too important. We propose a solution that reaches the minimum bound in terms of memory requirements but the number and size of memories will always represent the highly dominant resource. An

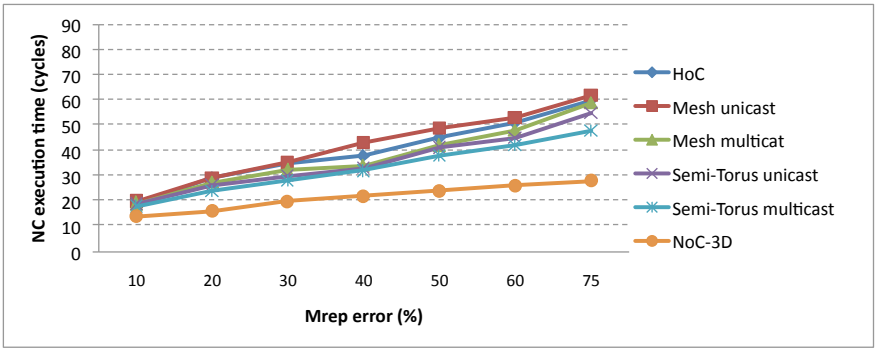


Figure 5: Exec. time vs error, neurons/cluster ratio=4

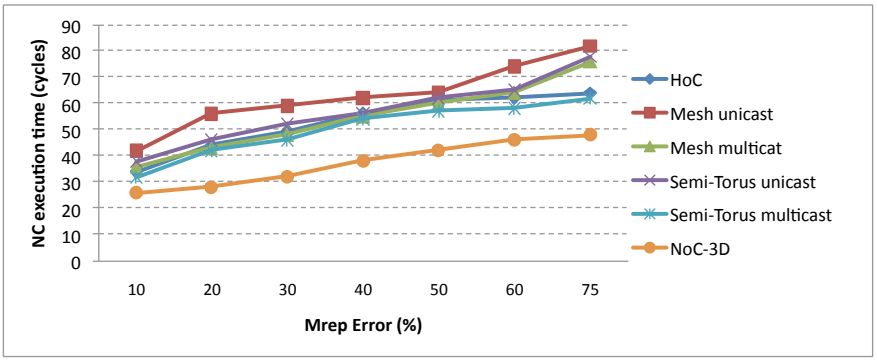


Figure 6: Exec. time vs error, neurons/cluster ratio=8

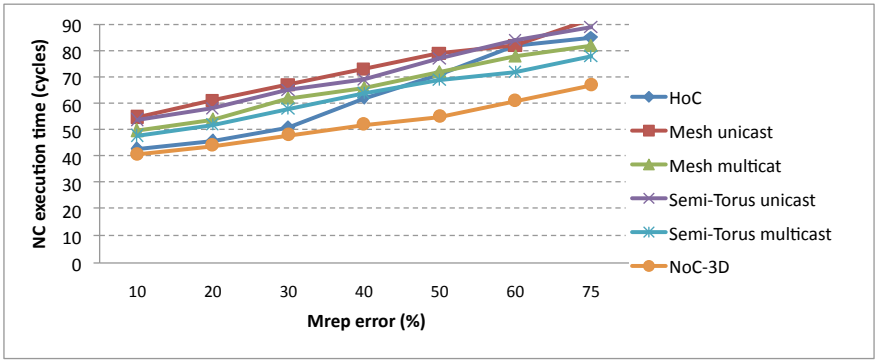


Figure 7: Exec. time vs error, neurons/cluster ratio=16

implementation of memory based on BRAM could a first solution for medium size NC, but the real target of NC-based associative memories is obviously ASIC technology.

NC	**Communication structure**		
	Configuration	**FFs**	**LUTs**
{4,4,6} (11 IPs)	HoC	514	721
	Regular mesh (unicast)	452	614
	Regular mesh (multicast)	502	746
	Semi-Torus (unicast)	622	936
	Semi-Torus (multicast)	734	1021
	NoC 3D	1248	1972
{16,128,120} (137 IPs)	HoC	5672	8435
	Regular mesh (unicast)	4956	7122
	Regular mesh (multicast)	5482	8504
	Semi-Torus (unicast)	6702	10483
	Semi-Torus (multicast)	8212	11537
	NoC 3D	14222	22086

Table 1: NoC Synthesis results, Virtex 5

NC	**Manager**		**WTAP**		**Memories**	
	FFs	**LUTs**	**FFs**	**LUTs**	**FFs**	**LUTs**
{4,4,6} (11 IPs)	5	52	28	36	23	30
	-		+1 BRAM		-	
{16,128,120} (137 IPs)	12	57	44	77	16384	28563
	-		+4 BRAM		-	

Table 2: NC components Synthesis results, Virtex 5

7 SystemC Simulations

7.1 Experimental setup

To be relevant, the demonstration of the NC concept requires an extensive set of simulations since it based on the learning and more importantly on the retrieval of a large number of messages with different parameters

NC	Communication structure	
	Configuration	**Power (mW)**
{4,4,6} (11 IPs)	HoC	10.8
	Regular mesh (unicast)	10.2
	Regular mesh (multicast)	11.6
	Semi-Torus (unicast)	12.4
	Semi-Torus (multicast)	12.8
	NoC 3D	18.2
{16,128,120} (137 IPs)	HoC	37.1
	Regular mesh (unicast)	35.3
	Regular mesh (multicast)	38.4
	Semi-Torus (unicast)	44.8
	Semi-Torus (multicast)	48.3
	NoC 3D	52.3

Table 3: Power consumption for NC configurations with 50% of error

including the number of messages, the number and the position (i.e. neuron) of errors. In order to speedup the simulation time and increase the number of experiments, SystemC-TLM models were developed. All NC components (Connection memories, WTAP, Global manager) and the NoC have been specified at a TLM level, the execution time of memories and WTAP are given as the number of cycles obtained with previous hardware implementations. The SystemC models of the Global Manager and of the NoC are cycle accurate. The impact on simulation time of the NoC specification level is significant but routers must be evaluated every cycle.

First different sets of complete messages are produced with a random law so that connection bits are uniformly distributed within connection memories. Then parts of the messages are erased according to controlled input error rate e, which means that the average number of erased neurons within a message is equal to e. If we consider for instance 16 clusters, $e = 25\%$ means that the average number of clusters, where information is missing, is equal to 4; in other words the decoding process will have to repair four neurons.

Then multiple simulations, based on distinct runs of the random process, are executed for different numbers of messages in order to demonstrate the relation between the success rate of the system to recover partially erased messages and the network density (i.e. the ratio between the number of learnt messages and the maximum number of messages that can be stored). In [4] the authors provide the analytical model to compute the error probability of recovering the initial message after a single iteration; this equation depends on the number of missing neurons, the number of learnt messages and the network parameters (c, l). They demonstrate the NC algorithm with software simulations that are compared to the theoretical values. We have followed the same process with our SystemC model.

7.2 Results

Fig. 8 presents our results for a NC architecture with 16 clusters of 128 neurons interconnected with the multicast mesh NoC. Three input error rates e have been simulated (10%, 30%, 40%) for different number of messages from 100 to 50,000. Each point of a curve is the average of four simulations with different random values. Each simulation requires from 9 to 30 minutes of CPU time to find a solution (or an ambiguity) according to the number of iterations, so each point of a curve was obtained after 36 to 120 minutes. This simulation time justifies the choice of high level SystemC-TLM models.

The dotted lines give the theoretical error probabilities for the three cases studies after a single iteration, it means that improvements can still be obtained after multiple iterations. On a curve of error rate vs number of learnt messages, this improvement corresponds to a shift to the right since the decoding process can repair messages with a higher density with multiple iterations.

The first observation is the demonstration of the correctness of the NC algorithm and of our architecture.

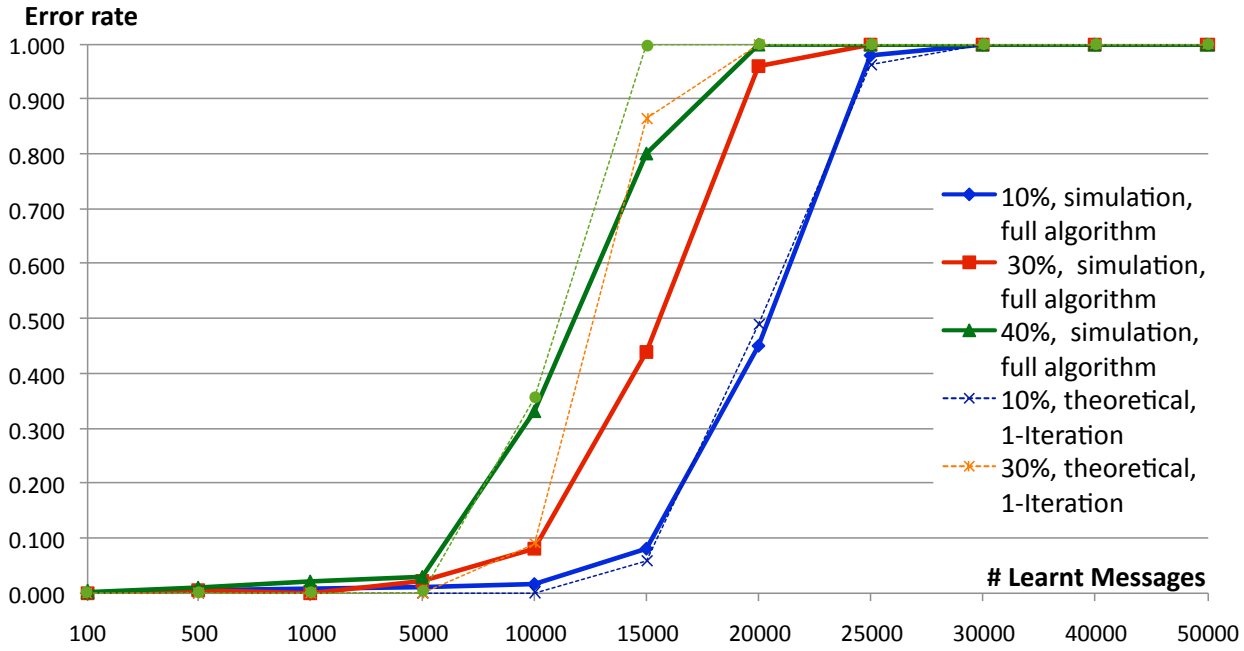


Figure 8: Error rate vs number of learnt messages for NC(l=128,c=16, Mesh NoC): full algorithm SystemC-TLM simulations and 1-iteration theoretical values

As predicted by theory, the system is able to learn and recover large numbers of partially known messages. We note for instance that with an important input error rate of 40%, 5000 messages can be learnt and decoded with an error rate lower than 2%. For all configurations a threshold with a sharp transition from 0 to 1 appears, it corresponds to a kind of saturation, which is the limit of the correction power of the network. In these transition phases, the dispersion of simulation results is important, it actually strongly depends on the occurrence of unsolvable ambiguity situations and also on the number of simulations per point.

The average of four instances of each point cannot faithfully represent a probability, however we still observe remarkable results despite these limitations. The first important point is position of the thresholds that correspond to the maximum number of messages that can be faithfully learnt, namely before the saturation starts. Secondly the relative positions of the simulation curves, according to the theoretical values, are also very good, we clearly note a shift to the right for 30% and 40% case studies. The 10% simulation curve is very close to the theoretical one, this result was expected since most of the time a single iteration is necessary to solve the decoding process with low error rates. The points located under the theoretical curves are due to the limited number of experiments in regions close to 0 and 1 combined with the important dispersion observed for this case study.

Our architecture faithfully implements the NC algorithm, which has an intrinsic efficiency proved to be better than HNN with a factor of a least one order of magnitude. The proposed architecture is optimized and uses the minimum number of (connection) memory bits, which represent by far the most significant integration cost, moreover this architecture takes advantage of a NoC to acquire essential properties such as scalability and flexibility. Considering also the cost of CAM-memories, which are not adapted to large memory sizes, we conclude that this family of architectures becomes a very attractive alternative to design efficient associative memories.

8 Perspectives

The proposed NC is the first step towards more complex networks, it is actually designed for the retrieval of a single message from a large set of learnt messages associated with a given concept. A message corresponds to an instance of this concept that can be a word from a language, a note from melody, etc. In future bio-inspired associative memories, multiple concepts will be associated like a human memory, which are basically an association of concepts (e.g. location, word, faces, voices, ...). It means that future associative memories will be hierarchical to mimic connections that exist between macrocolumns in the neocortex. Fig. 9 represents the kind of hierarchical NC that could be designed. In these future structures, the complexity of interconnections will increase and the efficiency and scalability of NoCs will be of primary importance to implement connections between thousands of memories. If we consider 10 concepts and if each of them is implemented with a NC of type {16,128,120}, it means that 1370 IPs including 1200 connection memories must be interconnected with a NoC. Considering the complexity of such architectures and the length of resulting wires on a single plan, we believe that 3D NoC appear as the solution for future implementations. An interesting trade off between performance constraints and power/area costs can for instance be obtained by means of a combination of MESH (e.g. first level of NC) and 3D (higher levels) topologies.

In future work we will address some implementations alternatives. The first one is the content of the packets, instead of sending full lines or diagonals of bits from the connection memories, it actually makes sense to send only the ID of existing connections. The efficiency of this solution lies on a tradeoff that depends on the number of neurons and on the sparsity of memories (number of '1'). Another dimension that is worth to explore is the size of the flits, for instance a serial implementation with flits of size 1 is an extreme version that may be interesting if the response time constraint can be fully relaxed. Finally, the number of WTAP can be reduced since the number of errors will always be lower than the number of clusters (e.g. $0.75 \cdot c$), it means that a dynamic allocation of WTAP may be considered.

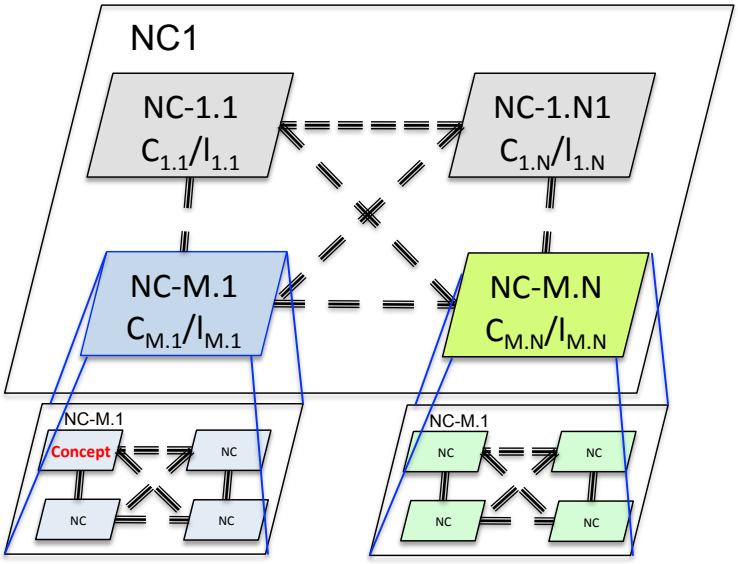


Figure 9: Hierachical NC for multiple concept association

9 Conclusion

Neural Coding is a promising solution for future efficient associative memories, it will be based on a huge number of memories and processing elements to be interconnected, which means that efficient NoC will be necessary to provide the required scalability. The memory cost is by far more important than the processing

one. In this paper we have detailed an architecture that meets the minimum memory bound, we have also presented an optimized implementation of the *winner-takes-all* processor. We have proposed a complete study of dedicated NoCs including performance, area and power results. Our implementation results show that a combination of different topologies (e.g. 3D+MESH) can offer the expected requirements to interconnect future hierarchical NC. The simulation results confirm expected theoretical values and validate the proposed architecture. This is a first solution that will be our reference design for future architectures, which will also include ongoing theoretical work on neural coding.

References

[1] H. Ackley, E. Hinton, and J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, pages 147–169, 1985.

[2] K. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circuits & Systems*, 47:416–434, 2000.

[3] Y. Dong, Y. Wang, Z. Lin, and T. Watanabe. High performance and low latency mapping for neuronal network into network on chip architecture. In *IEEE Int. Conf. on ASIC (ASICON)*, pages 891–894, 2009.

[4] V. Gripon and C. Berrou. Sparse neural networks with large learning diversity. *Neural Networks, IEEE Transactions on*, 22(7):1087 –1096, july 2011.

[5] M. Healy. Design and analysis of 3d-maps: A many-core 3d processor with stacked memory. In *Custom Integrated Circuits Conference (CICC)*, 2010.

[6] H. Jarollahi, N. Onizawa, V. Gripon, and W. Gross. Architecture and implementation of an associative memory using sparse clustered networks. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 2901 –2904, may 2012.

[7] K. Pagiamtzis and A. Sheikholeslami. Content-addressable memory (cam) circuits and architectures: a tutorial and survey. *Solid-State Circuits, IEEE Journal of*, 41(3):712 – 727, march 2006.

[8] A. Y. R. Emery and G. Chester. Connection-centric network for spiking neural networks. In *IEEE/ACM NOCS conference*, pages 144–152, 2009.

[9] A. Sheibanyrad. 3d integration for noc-based soc architectures. *Integrated Circuits and Systems. Springer.*, 2011.

[10] A. Storkey. Increasing the capacity of a hopfield network without sacrificing functionality. In *ICANN97: Lecture Notes in Computer Science 1327*, pages 451–456. Springer-Verlag, 1997.

[11] T. Theocharides, G. Link, N. Vijaykrishman, J. Irwin, and V. Srikantam. A generic reconfigurable neural network on chip. In *SoC Conference*, pages 191–194, 2004.

[12] D. Vainbrand and R. Ginosar. Network-on-chip architectures for neural networks. In *ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, pages 135–144, Washington, DC, USA, 2010.