

Scalable network-on-chip architecture for configurable neural networks

Dmitri Vainbrand*, Ran Ginosar

Technion—Israel Institute of Technology, Haifa, Israel

ARTICLE INFO

Article history:

Available online 13 August 2010

Keywords:

Networks on Chip
Reconfigurable neural networks
Hardware implementation
Interconnect architecture

ABSTRACT

Providing highly flexible connectivity is a major architectural challenge for hardware implementation of reconfigurable neural networks. We perform an analytical evaluation and comparison of different configurable interconnect architectures (mesh NoC, tree, shared bus and point-to-point) emulating variants of two neural network topologies (having full and random configurable connectivity). We derive analytical expressions and asymptotic limits for performance (in terms of bandwidth) and cost (in terms of area and power) of the interconnect architectures considering three communication methods (unicast, multicast and broadcast). It is shown that multicast mesh NoC provides the highest performance/cost ratio and consequently it is the most suitable interconnect architecture for configurable neural network implementation. Routing table size requirements and their impact on scalability were analyzed. Modular hierarchical architecture based on multicast mesh NoC is proposed to allow large scale neural networks emulation. Simulation results successfully validate the analytical models and the asymptotic behavior of the network as a function of its size.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The inherent parallelism of multi-processor VLSI systems on chip (SoC) enables the efficient emulation of biological neural networks (NN) and the construction of artificial neural networks for complex tasks such as pattern recognition. When the structure and connectivity are implemented rigidly in hardware, the emulated neural networks suffer from limited flexibility and functionality [6,7], requiring redesign if any connectivity or function needs to be changed. To overcome these limitations, we seek a SoC architecture that enables programmable and reconfigurable NN. Such architecture could serve as a generic medium for neuroscience and machine learning research, enabling emulation of arbitrary neural network topology and supporting dynamic connectivity changes as a result of training.

The high level architectural framework for this research is a multiprocessing chip (CMP) comprising a large number of homogeneous processors, specially reconfigured to optimize neuronal and synaptic functions implementation (Fig. 1). Each processor emulates multiple neurons. Connectivity within the processor is implemented in software.

A key issue in emulating reconfigurable neural networks is the complexity of dynamic and flexible neural connectivity. The purpose of this work is to find best suitable on-chip interconnect architecture for NN emulation. We investigate and compare four interconnect architectures (mesh NoC, tree, bus, point-to-point)

and three packet-based communication methods (unicast, multicast and broadcast) and study how they support configurable communications for spiking neural networks. We show that mesh NoC using multicast is the most suitable architecture for a wide range of neural network topologies. It is evident from the analysis that the NoC maximally preserves the inherent parallelism of neural networks, the mesh topology provides high operational frequency and preserves relative latency, and multicast closely emulates one-to-many neural communications. We analyze routing tables (RT) size requirements for each architecture. We discuss methods for reducing RT size and propose modular hierarchical architecture that facilitates scalability and allows large-scale neural network emulation.

Background and related work are presented in Section 2. Section 2.1 provides a short overview of spiking neural networks and their basic properties. Previous work in the field of neural networks implementation is presented in Section 2.2. Early implementations demonstrated a clear trade-off between flexible connectivity and high element count. Later implementations employed an arbitrated shared bus and an Address Event Representation (AER) protocol. More recent papers present spiking NN implemented on NoC, raising the need for comparative analysis that will answer the question of which architectural choices are preferred, and whether indeed packet switching is the appropriate interconnect solution for the emulation of large scale neural networks, thus motivating the present paper.

Theoretical cost and performance analysis is provided in Section 3. Two types of neural network models are investigated: Hopfield network defined by all-to-all connectivity of the

* Corresponding author. Tel.: +972 54 788 5953.

E-mail address: Dmitri.Vainbrand@intel.com (D. Vainbrand).

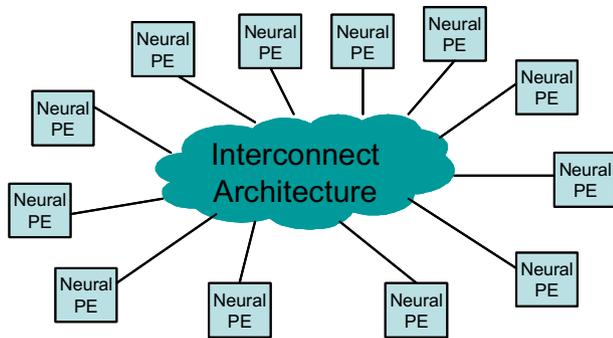


Fig. 1. Neural network on SoC.

neurons, and Randomly Connected NN, defined by local connectivity with a probability for connection that decreases exponentially with distance. In Section 3.1, the metrics for performance and cost evaluation are defined. Performance is measured by the zero-load effective bandwidth of the interconnect (BW_{eff}) which reflects the network capacity, the delivered spike rate which is the average rate in which the network can receive or deliver spikes from or to a single neuron ($f_{p,out}$), and the maximum spiking frequency of the neuron (offered spike rate, which depends on the biological model, $f_{spike,max}$). We also define a K -factor, representing the ratio between offered and delivered spike rates. Another figure of merit reflects spike latencies. The cost is presented by area and dissipated power. Finally, the performance to cost ratio, R , is defined as bandwidth divided by area and power. In Section 3.2, we compare the performance of unicast, multicast and broadcast mesh NoC emulating neural network topologies having full and random configurable connectivity. It is evident from the analysis that for both NN topologies multicast is the preferred communication method as it provides higher bandwidth and higher frequency. Intuitively, while in unicast NoC each spike produces multiple packets, one for each destination, and in broadcasting NoC the packet always traverses all links, multicast provides optimal network utilization, as only one packet per spike is generated and it is routed only to the relevant destinations. The tree NoC is analyzed in Section 3.3. We show that in order to support general configurable connectivity, a full fat-tree is required, resulting in longer links and lower operational frequency. Thus, although the tree has a shorter diameter (in number of hops), it provides lower average bandwidth and higher realization cost than MC mesh NoC. Shared bus and point-to-point architectures are analyzed and compared to MC mesh NoC in Sections 3.4 and 3.5, respectively. While limited parallelism of the shared bus yields much lower bandwidth at the same cost as the mesh NoC, point-to-point architecture provides slightly better bandwidth than MC mesh NoC but at a much higher realization cost. In Section 3.6, analytical results are discussed and asymptotic limits are summarized.

Performance simulations are summarized in Section 4. The simulations help to gain insight into the network behavior and to validate our analytical models. Traffic of the fully connected and the exponentially connected NNs, emulated on mesh NoC, were simulated using UC, MC and BC communications. Maximum input rate as a function of network size is derived. Maximum and average offered spike rates are derived and compared to maximum input rate. Good fit is achieved between empirical results and the analytical model. Simulations show that full network connectivity incurs higher average delay and lower achievable firing rate than exponential connectivity. In addition, multicast enables higher firing rate than unicast and broadcast, and scales better with network size.

The size of the routing tables is discussed in Section 5, as well as architectures for table size reduction. In Section 5.1, we analyze

routing tables and general storage requirements for each proposed architecture and communication methods. Allowing arbitrary connectivity requires $O(n^2)$ storage entries, regardless of the architecture, a major scaling limitation. For the MC mesh NoC architecture, which is best in terms of performance and cost (Section 3), distributed routing is preferred over source routing. In Section 5.2, we find that when only bounded connectivity is allowed, the routing table size is independent of the network size. Non-uniform storage allocation further decreases the overall storage size or, alternatively, extends the connectivity bounds. Finally, in Section 6 we propose modular hierarchical architecture. Modularity helps extend the architecture from a single chip to a multi-chip card and further to multi-card system, supporting NN of up to one million neurons. Conclusions and possible future research are presented in Section 7.

2. Background and related work

2.1. Spiking neural networks

Artificial neural networks have evolved over three different generations [31], with each generation raising a level of biological realism relative to its predecessor. The first generation McCulloch-Pitts threshold neuron [31] produces a ‘high’ output if the linear combination of the inputs exceeds a threshold. Neurons of the second generation employ a continuous activation function that outputs a firing rate, which is more biologically realistic and more computationally powerful [5]. The third generation spiking neural networks [32] are inspired by neurobiological studies indicating that neural information is not entirely encoded by firing rate but also by spike timing and by spatio-temporal codependency between spikes of different neurons. Spiking neurons communicate by short pulses. Spiking Neural Networks (SNN) are the subject of this paper.

Various models of spiking neurons exist [33]. They differ from each other in the level of biological realism, but their common principle is the leaking integration of charge produced by incoming spike. They fire a short spike when the integrated charge exceeds a threshold.

Modeling of SNN connectivity structure also varies. In the fully connected Hopfield network [4] every neuron sends its output to all other neurons. It is useful for our discussion since it represents an upper bound on connectivity. The alternative RNDC NN (randomly connected neural network) model [1] is inspired by neurobiological data [34] suggesting that connectivity is rather stochastic and the probability of connection between two neurons decreases exponentially with the distance between them, while the number of connections for each neuron increases with distance. RNDC connectivity is a simple representative of various NN models suggesting decrease of connection probability with distance.

We note that emulation of both types of NNs on a multiple processor architecture results in uniform traffic among the processors. Traffic is uniform if all the links of the interconnect are equally utilized. More formally, traffic is uniform if the rate of traffic messages crossing any crosscut that divides the processors into two roughly equal groups is the same regardless of how the crosscut is drawn. Uniformity of traffic can easily be explained by observing uniformity in average spike firing rates of all neurons.

2.2. Neural network implementation

Parallel processors and special purpose hardware are most suitable for fast emulation of computationally intensive neural networks, for both real time applications and for very large tasks

such as long-term learning simulations, which (thanks to the high performance of modern hardware) can be executed much faster than real time. Previous highly flexible and configurable implementations were limited to emulating a small number of neurons [8], while systems of many neurons offered limited flexibility [6,7], and flexible systems of many neurons required very large implementations [9], due to the quadratic complexity of a fully connected design. Analog implementations of spiking neural networks appear to execute faster, and require less power than some digital ones [35], but they are usually much less flexible, less programmable and employ fixed interconnect. Hence, they are less suitable for general purpose neural network chips. Systolic array implementations of NN, such as SYNAPSE [36] imply a fixed interconnect topology and are limited in the models that they can implement. Systolic arrays are most suitable to conventional static neural networks (compute-bounded problem), but inadequate for the simulation of spiking neural networks (interconnect-bounded problem).

Hierarchical and simplified communication methods have been used to mitigate the complexity of full connectivity. Shared bus neural network implementations typically provide efficient event driven communications, whereby only the address of the spiking neuron is broadcast to all bus elements. Non-arbitrated shared bus, which detects and ignores collisions, was described in [10]. Other works employed arbitrated buses using the Address Event Representation (AER) asynchronous protocol [11–13]. That representation is useful for either point-to-point connections [11,12] or broadcasting over a shared bus [13]. The former case requires expensive communication network while parallelism is quite limited in the latter case.

Network on Chip (NoC) architecture have been proposed by Jantsch et al. [39,14–16] to solve interconnect problems in SoC. NoC are particularly attractive for spiking neural networks, as they facilitate parallelism, reconfigurability, independence of the network topology, and network expandability. Multicast NoCs [40] closely emulate one-to-many neural communications. A small 2D torus network with four processing elements (neurons) per routing node is described in [17]. The architecture supports deadlock free XY routing, and uses wormhole packet switched communication. The packet combines the outputs of four logical neurons and is sent to the router of the next layer of the layered neural network router, thus enabling only layered structure and not allowing arbitrary communications. That NoC architecture is less applicable to spiking neural networks. Scaling is also limited.

An FPGA-based mesh NoC using XY unicast routing for clustered neural network has been proposed in [19]. As shown in this paper, unicast communications may be a limiting factor in neural network implementations.

Spinnaker, a large scale multi-chip spiking neural-network system was reported in [18]. A hierarchy of seven chips, 20 processors per chip and 1000 spiking neurons per processor is contemplated. Small AER packets are exchanged using multicast NoC. As the ability to emulate arbitrary neural network connectivity depends strongly on the size of routing tables, the paper addresses optimization techniques to reduce table sizes.

While [18] employs a mesh, [17] uses 2D torus and [19] describes a more complex hierarchy. While [17,19] rely on unicast, [18] chooses multicast. While [17] uses wormhole routing, [18,19] employs short AER packets. Packets in [18] consist of source address and [19] uses destination address. This variety of approaches raises the question of which architectural choices are preferred over others, and whether indeed NoC is the appropriate solution for the emulation of large scale neural networks. This paper attempts to address these questions, by an analytical evaluation and comparison of different interconnect architectures emulating different neural network topologies. Previous cost/per-

formance characterizations employed either analytical [22,23] or empirical [24,25] approaches. We adopt the method of [20], for its generality and applicability to NoC and non-NoC architectures, as explained in the following section.

Storage size is a big obstacle to scalable NoC design. It is suggested in [27] that there is a trade off between the efficiency of routing scheme and its storage requirements. It is shown that general routing schemes, which guarantee a stretch factor of k (reflecting that the average path is k times longer than optimal), require storing a total of $O(n^{(1+1/k)} \log n)$ bits of routing information in the network, tag the vertices with $O(\log n)$ -bit names and use $O(\log n)$ -bit packet headers. For optimal routing path ($k = 1$) the storage requirement is $O(n^2 \log n)$. There exist different approaches toward the reduction of routing table size. Interval routing is presented in [28] and is extended for regular meshes and tori in [29]. Interval routing reduces RT sizes in large networks by grouping the set of destination addresses that share the same output port into intervals of consecutive integers. In turn-table routing [30], used for irregular mesh, the router performs a default operation (“route XY” or “don’t turn”) unless it finds the destination address in its routing table. Both approaches decrease routing tables within the routers but do not deal with $O(n^2 \log n)$ destination entries within the sources. In this paper we evaluate the scalability of different interconnect architectures by analyzing their total storage requirements (in routers, sources and destinations) and compare them to each other and to the lower bound in [27]. In addition, a scalable modular NoC architecture is proposed

3. Network architecture

This work investigates the emulation of neural networks on many processors inside a single SoC. Several neurons can be allocated to each processor. Spikes transmitted between neurons that are assigned to the same processor are transferred internally within the processor. Spikes between neurons assigned to different processors must be transferred over the interconnect. In the following, Section 3.1 defines cost and performance metrics. In Sections 3.2–3.5, we consider four alternative implementations for how the processors are interconnected, and evaluate how the interconnect architecture impacts the cost and performance of configurable neural networks. In Section 3.6, we summarize the results. We compare the following interconnect architectures:

NoC_Mesh: a mesh network-on-chip.

NoC_Tree: a tree network-on-chip.

AER_BUS: a broadcasting AER protocol on a shared bus.

P2P: a point to point reconfigurable connection matrix.

Note that other common NoC topologies such as hexagon mesh [18] and torus [17] are closely related to the mesh. In addition, the wrap-around property of the torus does not reflect biological systems, as the cortex itself may be modeled as a flat surface.

We consider three different communication methods (“casts”): unicast (UC), multicast (MC) and broadcast (BC). Two types of neural network models are investigated: Hopfield network [4], which requires full connectivity of the neurons, and Randomly Connected NN (RNDC NN), defined by random exponential connectivity. The former represents maximal connectivity bounds, whereas the latter, adopted from [1], represents more realistic scenarios.

3.1. Cost and performance metrics

We use a metric similar to [20]. Cost is associated with interconnection area and power dissipation (another cost item, not discussed in this paper, relates to the size of memory needed for

routing tables). Performance is evaluated by effective bandwidth, throughput and maximal spiking frequency of the neurons. The maximal theoretical bandwidth is:

$$BW(nn, arch, cast) = \frac{\sum_{i \in \{\text{links}\}} w(i)f(i)}{TotalDist(nn, arch, cast)} \quad (1)$$

were $w(i)$ is the width of link i , $f(i)$ is its switching frequency, and $TotalDist(nn, arch, cast)$ is the average total distance traversed by each neural spike (going to all its destinations), measured in the number of hops; $arch \in \{\text{NoC_Mesh, NoC_Tree, AER_BUS, p2p}\}$, $cast \in \{\text{UC, MC, BC}\}$, and $nn \in \{\text{Hopfield, RNDC}\}$. While the maximum BW indicates the possible level of parallelism for a given architecture, communication method and neural network, it does not take into account inter-packet interactions and variable latency. If the network is allowed to operate at maximum BW, it stalls to a halt due to congestion. It is shown empirically in Section 4 that if the network operates below the congestion threshold, there are no congestion effects at all, and we can assume fixed small router delays. This is modeled by an empirical architecture-specific utilization factor U_{arch} , defining the effective bandwidth:

$$BW_{eff}(nn, arch, cast) = BW \times U_{arch} \quad (2)$$

For topologies with a constant number of wires per link \bar{w} and constant frequency, (1) becomes:

$$BW_{eff} = \frac{\bar{w} \cdot TL_{arch}}{TotalDist} f_{arch} U_{arch} \quad (3)$$

where TL_{arch} is the total number of links in the architecture. The area cost of the architecture is calculated as a total wire area consumed by the interconnect in all metal layers. The NoC circuit area is considered to be significantly smaller than the area of the interconnect wires and it scales much better than wires, thus it does not affect the asymptotic results:

$$A_{arch} = W_p \sum_{i \in \{\text{Arch links}\}} w(i)l(i) \quad (4)$$

where $l(i)$ is the length of link i and W_p is the wire pitch for a given technology. We disregard router delays, since they do not scale with network size and thus the link delay is also the link cycle time:

$$T_{cycle} = R_0 C_0 \bar{l}^2 \quad (5)$$

where R_0 and C_0 are the wire resistance and capacitance per unit length, respectively, and \bar{l} is the average link length. Thus the maximum link frequency is:

$$f_{arch} = \frac{1}{T_{cycle}} = \frac{1}{R_0 C_0 \bar{l}^2} \quad (6)$$

Power dissipation is estimated as dynamic power dissipated on the link and gate capacitances:

$$P_{arch} = \sum_{i \in \{\text{links}\}} C(i) f_{arch}(i) V_{DD}^2 U_{arch} \quad (7)$$

The maximum spiking frequency is determined by the biologically inspired *neuron refractory period* $T_{refractory}$, a “blinking” time following a spike during which the neuron cannot fire again:

$$f_{spike,max} = \frac{1}{T_{refractory}} \quad (8)$$

In biological cortical neural networks, the refractory period and the average synaptic (axonal) delay are typically in the same range (2–10 ms), which in our case equals the packet end-to-end delay:

$$T_{refractory} \cong T_{Ax} \cong T_{cycle} \cdot \overline{Dist} \quad (9)$$

where \overline{Dist} is the average distance (in number of hops) between two connected neurons.

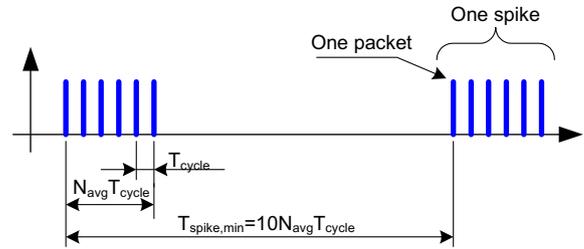


Fig. 2. Issue delay. UC maximal firing rate.

In the special case of unicast NoC implementations, each spike results in a succession of packets, one per destination. The delay between issuing the first and the last packets of the same spike is $N_{avg} T_{cycle}$, where N_{avg} is the average number of spike destinations for a given neural network. We arbitrarily choose a minimal time between successive spikes fired by the same neuron of $10N_{avg} T_{cycle}$ (the inter-spike interval should be longer than the time of sending all packets of one spike: Fig. 2) and the maximal firing frequency for unicast NoCs combines both issue and refractory delays:

$$f_{spike,max}^{UC} = \frac{1}{\min(T_{cycle} \cdot 10N_{avg}, T_{cycle} \overline{Dist})} \approx \frac{1}{10N_{avg} T_{cycle}} \quad (10)$$

For multicast and broadcast, only one packet is issued per each spike regardless of its number of destinations. The maximal firing frequency for multicast and broadcast NoC is thus implied by (9):

$$f_{spike,max}^{MC/BC} = \frac{1}{T_{cycle} \overline{Dist}} \quad (11)$$

The above definition of maximal firing frequency reflects the basic property of refractory time in neural networks in the presence of the geometrical and electrical delays of the implementation. Note that the maximal firing frequency does not necessarily match the maximal NoC bandwidth. Moreover, as shown below, some of the studied implementations (especially broadcasting NoC and BUS) are jammed when all neurons fire constantly at their maximal frequency.

BW_{eff} of (3) is the average rate at which an entire network absorbs new messages. The average rate at which a single processor can feed spikes into the network is obtained by dividing into the number of processors n_p :

$$f_{p,out} = \frac{BW_{eff}}{n_p} \quad (12)$$

The fact that the maximal firing rate $f_{spike,max}$ may be different than $f_{p,out}$ is expressed by a factor K , indicating the degree to which a given implementation enables neurons to operate at their maximal rate and providing a figure of merit for comparing implementations. In other words K is the ratio of delivered and offered spiking rates:

$$K = \frac{f_{p,out}}{f_{spike,max}} \quad (13)$$

If $K > 1$, multiple logical neurons can fit efficiently into a single physical processor. If $K < 1$, $f_{spike,max}$ is unachievable. Finally, the cost–performance ratio R is:

$$R = \frac{BW_{eff,arch}}{A_{arch} \cdot P_{arch}} \quad (14)$$

As noted above, NN information is encoded both spatially (who sent the spike) and temporally (when the spike occurred). Thus, the neural network emulation has to preserve both spatial and temporal relations among neurons. Timing of spike arrivals at neurons can be preserved if the spike propagation latencies in the biological NN are imitated in the emulation. To reflect this property, we define

Relative Latency Preservation, based on the ratio of the biological to architectural latencies l_{NN}/l_{arch} and considering the normalized variance of that ratio.

The average of l_{NN}/l_{arch} ratio is given by:

$$\mu \left[\frac{l_{NN}}{l_{arch}} \right] = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1}^n \left[\frac{l_{NN}(i,j)}{l_{arch}(i,j)} \right] \quad (15)$$

The variance is:

$$\text{Var} \left[\frac{l_{NN}}{l_{arch}} \right] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \left[\frac{l_{NN}(i,j)}{l_{arch}(i,j)} - \mu \left[\frac{l_{NN}}{l_{arch}} \right] \right]^2 \quad (16)$$

And the relative latency preservation ratio is:

$$RLP = \frac{\text{Var} \left[\frac{l_{NN}}{l_{arch}} \right]}{\mu \left[\frac{l_{NN}}{l_{arch}} \right]} = \frac{\frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \left[\frac{l_{NN}(i,j)}{l_{arch}(i,j)} - \mu \left[\frac{l_{NN}}{l_{arch}} \right] \right]^2}{\mu \left[\frac{l_{NN}}{l_{arch}} \right]} \quad (17)$$

$l_{NN}(i,j)$ and $l_{arch}(i,j)$ are the latencies between neuron i to neuron j in the original and emulated networks, respectively. Low levels of RLP indicate well preserved relative latency, meaning that the emulation process does not distort inter-neuronal delays but only scales them (usually downwards). $RLP = 0$ indicates maximum preservation.

3.2. Mesh NoC

A mesh NoC comprises n processors and n routers arranged in a $\sqrt{n} \times \sqrt{n}$ mesh (Fig. 3).

The total number of links in the mesh is:

$$TL_{Mesh} = 2\bar{w}\sqrt{n}(\sqrt{n} - 1) \quad (18)$$

The average distance between two nodes in the mesh is $\overline{Dist}_{Mesh} = \frac{2}{3}\sqrt{n}$ [38]

Considering the preservation of relative latency, we note that in the biological NN the spike propagation latency is proportional to the distance between neurons, $l_{NN}(i,j) \propto D(i,j)$, and thus if the physical mapping preserves the initial layout and the latency in the interconnect architecture is proportional to the distance, then the relative latency is well preserved. Both conditions are met in a mesh NoC: A cortical 2D surface is mapped to a 2D mesh, the mapping distortion is minimal and the distance is mapped to the number of hops: $D(i,j) \propto N_{hops}(i,j)$. If the network operates below the congestion threshold, the latency is $l_{arch}(i,j) \cong N_{hops}(i,j) \cdot T_{cycle}$, thus minimizing RLP_{mesh} , and ideally $RLP_{mesh} = 0$.

3.2.1. Emulating Hopfield NN on a mesh NoC

Consider a Hopfield (fully connected) NN emulated on a unicast mesh NoC. For simplicity of the analysis, assume that each processor emulates a single neuron. For each spike, a neuron sends a packet to all the other neurons ($n - 1$ packets). The total number

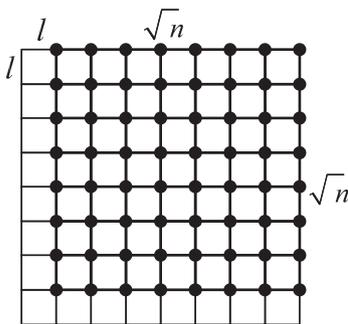


Fig. 3. Mesh NoC.

of hops traversed by one spike is the sum of distances between this neuron and all other neurons:

$$TotalDist_{UC,Mesh}^{Hopfield} = (n-1) \cdot \overline{Dist}_{Mesh} = \frac{2}{3}(n-1) \cdot \sqrt{n} \quad (19)$$

Substituting (18) and (19) into (3) yields:

$$BW_{eff,Mesh,UC}^{Hopfield} = \frac{3\bar{w}}{\sqrt{n}+1} f_{NoC} U_{NoC} \quad (20)$$

and the average frequency of feeding new spikes from a single processor is:

$$f_{p,out,Mesh,UC}^{Hopfield} = \frac{BW_{eff,Mesh,UC}^{Hopfield}}{n} = \frac{3\bar{w}}{n(\sqrt{n}+1)} f_{NoC} U_{NoC} \quad (21)$$

Following (10) we conclude:

$$f_{spike,max}^{UC} \cong \frac{f_{NoC}}{10n} \quad (22)$$

Comparing the results of (20) and (22) we can derive:

$$K = \frac{f_{p,out,UC}^{Hopfield}}{f_{spike,max}^{UC}} \cong \frac{30\bar{w}U_{NoC}}{\sqrt{n}} = O(1/\sqrt{n}) \quad (23)$$

This result implies that the UC mesh NoC does not offer sufficient bandwidth to emulate a Hopfield NN at the maximal firing frequency. Only about $nK = O(\sqrt{n})$ neurons may fire close to their maximal rate and the remaining other neurons will fire at a negligibly low rate. Alternatively, all neurons could fire at a $1/\sqrt{n}$ fraction of their maximal rate.

Turning now to multicast and broadcast NoCs, they are essentially the same for Hopfield NN, as each spike is transmitted to $n - 1$ other neurons. The number of hops traversed per spike is the number of edges in the mesh spanning tree,

$$TotalDist_{MC/BC}^{Hopfield} \cong n \quad (24)$$

Thus:

$$BW_{eff,MC/BC}^{Hopfield} = 2\bar{w}f_{NoC}U_{MC/BC} \left(1 - \frac{1}{\sqrt{n}}\right) = O(1) \quad (25)$$

Intuitively, one packet, sent to all neurons, utilizes the entire network. Thus, the network can handle only one spike at a time. The average NoC input frequency is:

$$f_{p,out,MC/BC}^{Hopfield} = \frac{BW_{eff,MC/BC}^{Hopfield}}{n} \cong \frac{2\bar{w}f_{NoC}U_{NoC}}{n} \quad (26)$$

The fact that the source neuron issues one packet per spike enables a much tighter bound for spiking frequency, following (11):

$$f_{spike,max} = \frac{1}{T_{refractory}} \cong \frac{3f_{NoC}}{2\sqrt{n}} \quad (27)$$

Comparing (27) and (26) yields:

$$K_{Mesh,MC/BC}^{Hopfield} \cong \frac{4\bar{w}U_{NoC}}{3\sqrt{n}} = O(1/\sqrt{n}) \quad (28)$$

The remaining parameters depend only on the interconnect topology, and are the same for all communication methods:

Table 1
Hopfield network emulated on a mesh NoC.

Metric	UC	MC	BC
BW	$O(1/\sqrt{n})$	$O(1)$	$O(1)$
Area		$O(n)$	
Power		$O(n)$	
Spiking frequency	$O(1/n)$	$O(1/\sqrt{n})$	$O(1/\sqrt{n})$
K		$O(1/\sqrt{n})$	

$$\begin{aligned}
A_{\text{mesh}} &= 2W_p l \bar{w} \sqrt{n} (\sqrt{n} - 1) \\
C_{\text{mesh}} &= C_0 2l \bar{w} \sqrt{n} (\sqrt{n} - 1) \\
f_{\text{mesh}} &= \frac{1}{R_0 C_0 l^2} \\
P_{\text{mesh}} &= P_0 2\bar{w} U_{\text{NoC}} \sqrt{n} (\sqrt{n} - 1) \\
P_0 &\triangleq V_{dd}^2 / R_0 l
\end{aligned} \tag{29}$$

The asymptotic metrics for Hopfield NN emulated on a mesh NoC are summarized in Table 1

3.2.2. Emulating RNDC NN on a mesh NoC

For RNDC model, the probability of having a connection from neuron a to neuron b is defined similarly to [1]:

$$p(a, b) = \frac{C}{2\pi\lambda^2} e^{-D(a,b)/\lambda} \tag{30}$$

where $D(a, b)$ is the Euclidean distance between a and b , λ is a spatial connectivity constant, and $C = N_{\text{links}} = \|\mathbf{p}(\cdot)\|$ is the average number of connections per neuron. The mean distance between two connected neurons is:

$$\overline{\text{Dist}} = \frac{1}{2\pi\lambda^2} \int \int_{x,y} \sqrt{x^2 + y^2} e^{-\sqrt{x^2 + y^2}/\lambda} dx dy = 2\lambda \tag{31}$$

When emulating RNDC neural network on a unicast mesh NoC, a neuron sends individual packets to all of its C destinations. The average total number of hops for each spike packet is:

$$\text{TotalDist} = \overline{\text{Dist}} \cdot N_{\text{links}} = 2\lambda C \tag{32}$$

Thus, the bandwidth (3) of RNDC emulated on unicast mesh NoC is:

$$BW_{\text{eff,UC}}^{\text{RNDC}} = \frac{\bar{w} \sqrt{n} (\sqrt{n} - 1)}{\lambda C} f_{\text{NoC}} U_{\text{NoC}} \tag{33}$$

Simulations in [1] provide an example of small neural microcircuits (~ 1000 neurons) that reach optimal performance with:

$$\lambda \cong \sqrt[3]{nC} \cong \sqrt[3]{n} \tag{34}$$

Substituting these values for unicast mesh:

$$BW_{\text{eff,UC}}^{\text{RNDC}} \cong \bar{w} \cdot \sqrt[6]{n} \cdot f_{\text{NoC}} U_{\text{NoC}} \tag{35}$$

Using (31) and (34) in (10) we obtain:

$$f_{\text{spike,max}}^{\text{UC}} = \frac{1}{10C \cdot T_{\text{cycle}}} = \frac{f_{\text{NoC}}}{10\sqrt[3]{n}} \tag{36}$$

Also,

$$f_{\text{p,out}}^{\text{UC}} = \frac{BW_{\text{eff,UC}}^{\text{RNDC}}}{n} = \left(1 - \frac{1}{\sqrt[3]{n}}\right) \frac{\bar{w}}{\lambda C} f_{\text{NoC}} U_{\text{NoC}} = O\left(n^{-\frac{5}{6}}\right) \tag{37}$$

Similarly, the K ratio is:

$$K_{\text{UC}} = \frac{f_{\text{p,out}}^{\text{UC}}}{f_{\text{spike,max}}^{\text{UC}}} = \bar{w} U_{\text{NoC}} \left(1 - \frac{1}{\sqrt[3]{n}}\right) \frac{10C}{\lambda C} = O\left(\frac{1}{\lambda}\right) = O\left(\frac{1}{\sqrt[3]{n}}\right) \tag{38}$$

For RNDC NN emulated on a multicast NoC, the total number of hops is approximated by the length of the linear path, traversing a distance of 2λ to the first destination and then adding one hop per destination:

$$\text{TotalDist}_{\text{Mesh,MC}}^{\text{RNN}} = C + 2\lambda \tag{39}$$

Thus, the bandwidth (3) of RNDC implemented on a multicast mesh NoC is:

$$\begin{aligned}
BW_{\text{eff,mesh,MC}}^{\text{RNDC}} &\cong \frac{2\bar{w} \sqrt{n} (\sqrt{n} - 1)}{(C + 2\lambda)} f_{\text{NoC}} U_{\text{NoC}} \\
&= \frac{2\bar{w} \sqrt{n} (\sqrt{n} - 1)}{(\sqrt{n} + 2\sqrt[3]{n})} f_{\text{NoC}} U_{\text{NoC}} = O(\sqrt{n})
\end{aligned} \tag{40}$$

Using (31) and (34) in (11) we obtain:

$$f_{\text{spike,min}}^{\text{MC}} = \frac{f_{\text{NoC}}}{2\lambda} = \frac{f_{\text{NoC}}}{2\sqrt[3]{n}} \tag{41}$$

Following (40):

$$f_{\text{p,out}}^{\text{MC}} = \frac{2\bar{w} (\sqrt{n} - 1)}{\sqrt{n} (C + 2\lambda)} f_{\text{NoC}} U_{\text{NoC}} = O\left(\frac{1}{\sqrt{n}}\right) \tag{42}$$

Finally the K ratio is:

$$K_{\text{MC}} = \frac{f_{\text{p,out}}^{\text{MC}}}{f_{\text{spike,max}}^{\text{MC}}} = \frac{2\bar{w} (\sqrt{n} - 1) 2\lambda}{\sqrt{n} (C + 2\lambda)} U_{\text{NoC}} = O\left(\frac{\lambda}{C + \lambda}\right) = O\left(\frac{1}{\sqrt[6]{n}}\right) \tag{43}$$

Another practical case of random connectivity is the Locally Connected NN (LCNN) where:

$$\text{LCNN} = \{\text{RNDC} | \lambda \cong C \ll n\} \tag{44}$$

Thus, since the connectivity is practically bounded,

$$BW_{\text{eff,mesh,MC}}^{\text{LCNN}} = O\left(\frac{n}{C}\right)_{C, \lambda \ll n} \rightarrow O(n) \tag{45}$$

$$f_{\text{p,out}}^{\text{MC}} = \frac{2\bar{w} (\sqrt{n} - 1)}{\sqrt{n} (C + 2\lambda)} f_{\text{NoC}} U_{\text{NoC}} \xrightarrow{C, \lambda \ll n} O(1) \tag{46}$$

$$K_{\text{MC}} = \frac{\bar{w} (\sqrt{n} - 1) 4\lambda}{\sqrt{n} (C + 2\lambda)} U_{\text{NoC}} \cong \frac{4}{3} \bar{w} U_{\text{NoC}} = O(1) \tag{47}$$

Thus, the multicast mesh NoC offers sufficient bandwidth for emulating any size of locally connected NN. Both maximal firing frequency and the average NoC input frequency do not decrease when the network size grows. For instance, a NoC with one logical neuron in each processor is only 75% utilized even if all neurons fire constantly at their highest frequency.

With broadcasting NoC the spike is sent to all destinations regardless of connectivity pattern and the performance is independent of the neural network topology. Thus, all BC performance parameters are similar to those calculated for the Hopfield NN model (Eqs. (25)–(28)). The remaining cost factors depend on neither the NN topology nor the communication method. They are the same as in (29).

Setting U_{NoC} constant for all communication methods (UC, MC, BC) we achieve same operating frequency and power consumption but different levels of throughput: MC provides the highest throughput using the same power. Table 2 summarizes asymptotic results for the RNDC model implemented on a mesh NoC using different communication methods (UC, MC, BC).

3.3. Tree NoC

Consider NoC with a binary tree topology, having n physical neurons at the leaves and $n - 1$ routers as in Fig. 4. The results

Table 2
RNDC with $\lambda \cong \sqrt[3]{n}$, $C \cong \sqrt[3]{n}$ on a mesh NoC.

Parameter	UC	MC	BC
BW	$O\left(\frac{1}{n^{\frac{1}{3}}}\right)$	$O(\sqrt{n})$	$O(1)$
Area		$O(n)$	
Power		$O(n)$	
Spiking frequency	$O(1/\sqrt[3]{n})$	$O(1/\sqrt[3]{n})$	$O(1/\sqrt{n})$
K	$O(1/\sqrt[3]{n})$	$O\left(n^{-\frac{1}{6}}\right)$	$O(1/\sqrt{n})$

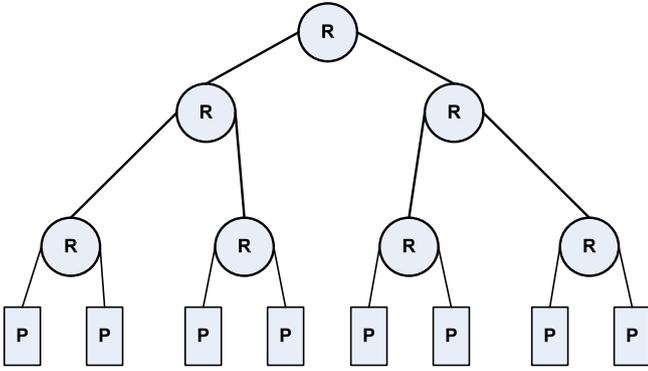


Fig. 4. Regular binary tree with eight leaves.

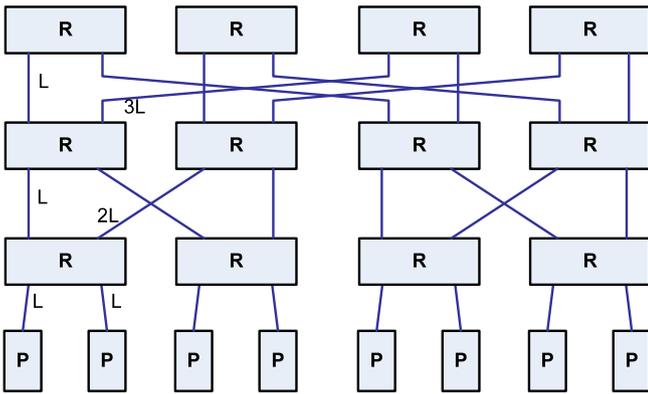


Fig. 5. Fat tree topology example.

can also be generalized to trees of higher degrees. The diameter of the binary tree is $2\log_2 n$. The total number of links is:

$$TL_{tree, NoC} = 2(n-1)\bar{w} \quad (48)$$

In a Hopfield (full connectivity) neural network with unicast communications, half of all traffic passes through the root, resulting in a serious congestion. This issue has been addressed by fat-trees (FT) [2], in which link bandwidth increases when going upward to the root, maintaining more uniform traffic. We employ the approximation of FT of [3] (Fig. 5), enabling nodes with small constant degree at the cost of more routers ($n\log(n)/2$) and more complex connectivity ($n\log(n)$ links). Such a FT introduces multiple alternative paths, providing additional bandwidth for short messages.

Based on the mesh NoC analysis of Section 3.2, we conclude that Hopfield NN is better emulated using either broadcast or multicast rather than unicast communications. Broadcast and multicast perform identically emulating Hopfield NN, and result in uniform traffic on the regular tree. For RNDC NN topology, recalling the exponential connection probability of (30), we investigate the question whether a FT is needed. The spikes sent from a neuron which is connected to other neurons at distance D traverse only a sub-tree of $k = \log(D)$ levels, or $k+1$ levels when connected to other neurons at distance $2D$ (a less likely case). The ratio of bandwidth required in levels $k+1$ and k is reflected by the ratio of the probabilities:

$$\frac{BW(k+1)}{BW(k)} = \frac{p(2D)}{p(D)} = \frac{e^{-2D/\lambda}}{e^{-D/\lambda}} = e^{-D/\lambda} = e^{-2^k/\lambda} \quad (49)$$

In a regular binary tree, the link capacity reduces by half at every level, $e^{-2^k/\lambda} = 0.5$, leading to:

$$k = \log_2(\lambda) - 0.52 \quad (50)$$

Thus, for connectivity with average distance 2λ , a sub-fat tree of height $k \sim \log_2(\lambda)$ provides the required communications, and the bandwidth of a normal binary tree suffices at the higher levels of the tree. In conclusion, the desired architecture combines a fat-tree at the bottom with a normal binary tree at the higher levels, depending on λ . To satisfy any value of λ , a full fat tree is required. Next we compare FT NoC performance to mesh NoC. The number of links in the FT:

$$TL_{FT, NoC} = n\log_2(n) \quad (51)$$

The average link length of the FT is computed as follows. Assuming layout as in Fig. 5 and counting link length as the sum of horizontal and vertical distances, the total distances in the FT is:

$$\sum_1^{n\log n} l_i = l \left(n \left(\log(n) - \frac{1}{2} \right) + \frac{n^2}{4} \right) = O(n^2) \quad (52)$$

And the average length $\bar{l}_{FT, NoC}$ is:

$$\bar{l}_{FT, NoC} = \frac{1}{n\log n} \sum_1^{n\log n} l_i \cong l \left(\frac{n}{4\log(n)} + 1 \right) \quad (53)$$

Recall that the average switching frequency is $f_{arch} \cong \frac{1}{R_0 C_0 l^2}$. For FT NoC this yields

$$f_{FT, NoC} \cong \frac{16(\log n)^2}{R_0 C_0 l^2 n^2} \quad (54)$$

Fat Tree maximal switching frequency decreases quadratically with network size and it is $\sim n^2$ times smaller than the mesh switching frequency

$$f_{FT, NoC} \approx \frac{16(\log n)^2}{n^2} f_{Mesh, NoC} \quad (55)$$

The ratio of FT to mesh total distances (in hops) traversed by one spike is expressed by the ratio of diameters:

$$TotalDist_{FT, NoC} \cong \frac{\log(n)}{\sqrt{n}} TotalDist_{Mesh, NoC} \quad (56)$$

The total length (in number of hops) of the FT is:

$$TL_{FT, NoC} = n\log_2(n) = \log_2(n) \cdot TL_{Mesh, NoC} \quad (57)$$

Thus, the FT NoC bandwidth is:

$$\begin{aligned} BW_{eff, FT, NoC} &= \frac{TL_{FT, NoC} f_{FT, NoC}}{TotalDist_{FT, NoC}^{NN}} U_{NoC} \\ &\cong \frac{16(\log_2 n)^3 \sqrt{n} \cdot TL_{Mesh, NoC} f_{Mesh, NoC}}{n^2 \log_2(n) TotalDist_{Mesh, NoC}^{NN}} U_{NoC} \\ &= \frac{16(\log n)^2}{n\sqrt{n}} BW_{eff, Mesh, NoC} \frac{BW_{eff, FT, NoC}}{BW_{eff, Mesh, NoC}} \\ &= O\left(\frac{(\log n)^2}{n\sqrt{n}}\right) \end{aligned} \quad (58)$$

The area of FT is given by

$$A_{FT, NoC} = TL_{FT, NoC} \cdot \bar{l}_{FT, NoC} \cdot \bar{w} \cdot W_p = \bar{w} \cdot W_p \cdot l \left[\frac{n^2}{4} + n\log n \right] \quad (59)$$

And the power dissipation (using P_0 as in (29)):

$$P_{FT, NoC} = C_T f_{FT, NoC} V_{dd}^2 U_{NoC} \cong 4(\log n)^2 \cdot P_0 U_{NoC} \quad (60)$$

The relative latency preservation of the tree is higher (worse) than that of the mesh. Inter-neuronal distances (as well as latencies), which are $O(\sqrt{n})$ in the biological NN as well as in the mesh, are converted to $O(\log n)$ on the tree. Thus, the average ratio is estimated at

$$\mu \left[\frac{l_{NN}}{l_{tree}} \right] = O \left(\frac{\sqrt{n}}{\log n} \right) \quad (61)$$

The distances in the tree are approximated by the average tree distance $\log_2 n$ and the original inter-neuron distances are approximated by distances in mesh. According to [38], the second moment of distances in 2D mesh, is:

$$E[l_{NN}^2] = \frac{5}{9}n - \frac{7}{9} + \frac{2}{9n} \quad (62)$$

Using (62) we derive the variance of latency relations:

$$\begin{aligned} \text{Var} \left[\frac{l_{NN}}{l_{tree}} \right] &= E \left[\left(\frac{l_{NN}}{l_{tree}} \right)^2 \right] - \left(\mu \left[\frac{l_{NN}}{l_{tree}} \right] \right)^2 \\ &= \frac{1}{(\log_2 n)^2} E[l_{NN}^2] - (\mu[l_{NN}])^2 \\ &= \frac{1}{(\log_2 n)^2} \left[\left(\frac{5}{9}n - \frac{7}{9} + \frac{2}{9n} \right) - \frac{4}{9}n \right] = O \left(\frac{n}{(\log_2 n)^2} \right) \end{aligned} \quad (63)$$

Thus, the relative latency preservation in NoC tree is:

$$RLP_{tree} \cong O \left(\frac{\sqrt{n}}{\log_2 n} \right) \quad (64)$$

In summary, a fat tree is required for implementing RNDC on a tree NoC, resulting in longer communication paths and consequently in lower frequency compared to a mesh NoC. In addition, the latency is not well preserved in the tree topology. Distances are distorted when mapped onto the tree, resulting in high variance of the latency transformations.

3.4. AER shared bus

The AER shared bus naturally employs only broadcast communications. A neuron transmits address events (namely, a packet containing only its address) on the bus once it gains bus control. Each receiving neuron compares the source address with the addresses of the neurons to which it is connected. Following [20] the total length of a bus is $L_{BUS} = \frac{l(n-4)}{2}$ (Fig. 6).

Each spike occupies the entire bus regardless of the topology of the emulated NN. Following (3), the bus effective bandwidth is:

$$BW_{AER_BUS}^{NN} = \bar{w} \cdot f_{BUS} \cdot U_{BUS} \quad (65)$$

The bus operating frequency can be related to the mesh NoC frequency following (6) and (29):

$$f_{BUS} = \frac{1}{R_0 C_0 L_{BUS}^2} = \frac{4}{(n-4)^2} f_{NoC} \quad (66)$$

Likewise, the bus BW can also be expressed in terms of the BW of the MC mesh NoC (the preferred mesh NoC communication method):

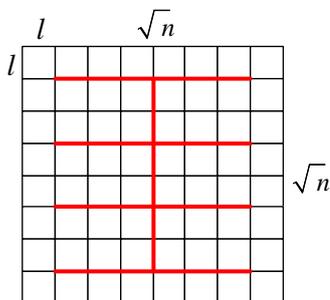


Fig. 6. AER shared bus topology.

$$BW_{AER_BUS}^{Hopfield} \cong BW_{Mesh,MC}^{Hopfield} \cdot \frac{4}{(n-4)^2} \frac{U_{BUS}}{U_{NoC}} \quad (67)$$

Evidently, AER bus utilization is lower than NoC utilization, and is decreasing as the network size grows. Even when disregarding this and assuming $U_{BUS} \approx U_{NoC}$, it is evident that NoC effective bandwidth is about n^2 times higher than the AER bus bandwidth. Moreover, the latter is $n^{2.5}$ lower than the bandwidth of mesh NoC used for RNDC emulation (40):

$$BW_{AER_BUS}^{RNDC} \cong BW_{Mesh,MC}^{RNDC} \cdot \frac{4}{\sqrt{n}(n-4)^2} \frac{U_{BUS}}{U_{NoC}} \quad (68)$$

On the other hand, the area consumed by the AER bus is about twice smaller than the area for NoC Mesh. The power dissipated by AER bus follows (7):

$$P_{BUS} = C_{TfBUS} V_{DD}^2 U_{BUS} = P_0 \cdot \frac{2}{(n-4)} U_{BUS} \quad (69)$$

where $P_0 \triangleq V_{dd}^2 / R_0 d$. Comparing with (29), observe that both the AER bus and the mesh NoC dissipate roughly similar power, $O(1/n)$. In summary, the mesh NoC offers higher performance than the AER bus at the same cost. Considering latencies, the bus transfers spikes at a fixed latency l_{BUS} of one cycle time, $O(1)$. In a biological NN, similarly to a mesh, the latency is a function of the distance:

$$\mu \left[\frac{l_{NN}}{l_{BUS}} \right] = \frac{\frac{2}{3}\sqrt{n}}{T_{cycle}} = O(\sqrt{n}) \quad (70)$$

The variance of the latency ratio is:

$$\begin{aligned} \text{Var} \left[\frac{l_{NN}}{l_{BUS}} \right] &= E \left[\left(\frac{l_{NN}}{l_{BUS}} \right)^2 \right] - \left(\mu \left[\frac{l_{NN}}{l_{BUS}} \right] \right)^2 \\ &= \frac{1}{T_{cycle}^2} E[l_{NN}^2] - (\mu[l_{NN}])^2 \\ &= \frac{1}{T_{cycle}^2} \left[\left(\frac{5}{9}n - \frac{7}{9} + \frac{2}{9n} \right) - \frac{4}{9}n \right] = O(n) \end{aligned} \quad (71)$$

and the relative latency preservation is higher than that of the mesh and the tree:

$$RLP_{BUS} = \frac{O(n)}{O(\sqrt{n})} = O(\sqrt{n}) \quad (72)$$

3.5. Point-to-point architecture

We consider n neurons arranged in a regular mesh and fully connected with point-to-point (PTP) unidirectional XY routed links. The total length of all PTP connections can be calculated by multiplying the total number of links $TL_{p2p} = n(n-1)/2$ by the average length of the link, $\bar{l}_{p2p} = 2l\sqrt{n}/3$

$$L_{P2P} = \frac{1}{3} l \sqrt{n} \cdot n(n-1) \quad (73)$$

The average frequency for P2P is:

$$f_{P2P} = \frac{1}{R_0 C_0 L_{P2P}^2} = \frac{9}{4R_0 C_0 l^2 n} = \frac{9}{4n} f_{NoC} \quad (74)$$

With Hopfield NN (full connectivity) every spike traverses $n-1$ links. The effective bandwidth is:

$$BW_{eff,P2P}^{Hopfield} = \frac{\bar{w} \cdot TL_{P2P} \cdot f_{P2P} \cdot U_{P2P}}{n-1} = \frac{\bar{w} \cdot n \cdot f_{P2P} \cdot U_{P2P}}{2} \quad (75)$$

It is intuitively evident that the P2P architecture can carry out $n/2$ simultaneous transactions with average frequency of f_{P2P} . Similarly to the AER BUS architecture, P2P bandwidth is compared with the multicast mesh NoC implementation (25):

$$BW_{\text{eff},P2P}^{\text{Hopfield}} = \frac{\bar{w} \cdot n \cdot 9f_{\text{NoC}} \cdot U_{P2P}}{4n} = \frac{9}{4} \frac{U_{P2P}}{U_{\text{NoC}}} BW_{\text{Mesh,NoC}}^{\text{Hopfield}} \quad (76)$$

Note that although there are more wires in P2P than in other networks, they are longer and hence slower so that the total bandwidth remains similar to that of the mesh NoC.

Considering RNDC (random local connectivity) implementation on P2P, every spike traverses on average C links. Applying average connection length of $2\lambda l$ to (6) yields the average frequency of

$$f_{P2P}^{\text{RNDC}} = \frac{1}{4R_0 C_0 l^2 \lambda^2} = \frac{1}{4\lambda^2} f_{\text{NoC}} \quad (77)$$

The effective bandwidth for RNDC implementation on P2P is

$$\begin{aligned} BW_{\text{eff},P2P}^{\text{RNDC}} &= \frac{\bar{w} \cdot TL_{P2P} \cdot f_{P2P}^{\text{RNDC}} \cdot U_{P2P}}{C} = \frac{\bar{w} \cdot (n-1)n \cdot f_{P2P}^{\text{RNDC}} \cdot U_{P2P}}{2C} \\ &= \frac{\bar{w} \cdot (n-1)n \cdot f_{\text{NoC}} \cdot U_{P2P}}{8C\lambda^2} \end{aligned} \quad (78)$$

Comparing this to MC mesh NoC architecture (40):

$$BW_{\text{eff},P2P}^{\text{RNDC}} = \frac{\sqrt{n}(\sqrt{n}+1)(C+2\lambda)}{16C\lambda^2} \frac{U_{P2P}}{U_{\text{NoC}}} BW_{\text{eff,mesh,MC}}^{\text{RNDC}} \quad (79)$$

Using the assumption in (34),

$$\begin{aligned} BW_{\text{eff},P2P}^{\text{RNDC}} &= \frac{(\sqrt{n}+1)(\sqrt{n}+2\sqrt[3]{n})}{16n^{\frac{3}{2}}} \cdot \frac{U_{P2P}}{U_{\text{NoC}}} BW_{\text{eff,mesh,MC}}^{\text{RNDC}} \xrightarrow{n \gg 1} \frac{\sqrt[3]{n}}{16} \\ &\cdot \frac{U_{P2P}}{U_{\text{NoC}}} BW_{\text{eff,mesh,MC}}^{\text{RNDC}} \end{aligned} \quad (80)$$

Thus, while the effective bandwidth of Hopfield implementation is about the same for P2P and mesh NoC, RNDC bandwidth of P2P can be higher than the mesh NoC. The PTP area is:

$$A_{P2P} = L_{P2P} \bar{w} = \frac{l\bar{w}}{3} \sqrt{n} \cdot n(n-1) \quad (81)$$

And P2P power dissipation is:

$$P_{P2P} \cong f_{P2P} \cdot C_T \cdot V_{dd}^2 U_{P2P} = \frac{3}{4} P_0 \sqrt{n}(n-1) U_{P2P} \quad (82)$$

Comparison P2P and mesh NoC costs:

$$\frac{P_{P2P}}{P_{\text{NoC}}} \cong O(\sqrt{n}) \quad \text{and} \quad \frac{A_{P2P}}{A_{\text{NoC}}} \cong O(n\sqrt{n}) \quad (83)$$

These results imply that although point-to-point architecture can provide better performance for certain neural network topologies than mesh NoC, it comes at a price of higher power dissipation and significantly higher area. The combined cost $A \cdot P$ of P2P is n^2 times higher than the mesh NoC.

Table 3
Cost and performance comparison.

	NoC mesh MC	NoC fat tree	AER bus	P2P
Hopfield BW	$O(1)$	$O\left(\frac{\log^2 n}{n\sqrt{n}}\right)$	$O\left(\frac{1}{n^2}\right)$	$O(1)$
RNDC BW	$O\left(\frac{n}{(C+2\lambda)}\right)$	$O\left(\frac{\log^2 n}{\sqrt{n}(C+2\lambda)}\right)$	$O\left(\frac{1}{n^2}\right)$	$O\left(\frac{(n-1)n}{C\lambda^2}\right)$
Practical RNDC BW	$O(\sqrt{n})$	$O\left(\frac{\log^2 n}{n}\right)$	$O(n^{-2})$	$O(n^{\frac{3}{2}})$
RLP	$O(1)$	$O\left(\frac{\sqrt{n}}{\log_2 n}\right)$	$O(\sqrt{n})$	–
Area	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2\sqrt{n})$
Power	$O(n)$	$O(\log n^2)$	$O(n^{-1})$	$O(n\sqrt{n})$
perf/cost Hopfield	$O(n^{-2})$	$O(n^{-3\frac{1}{2}})$	$O(n^{-2})$	$O(n^{-4})$
perf/cost RNDC	$O(n^{-1\frac{1}{2}})$	$O(n^{-3})$	$O(n^{-2})$	$O(n^{-2\frac{3}{2}})$

3.6. Summary of cost and performance

In the previous sections we have analyzed cost and performance of various interconnect architectures implementing neural networks. In Section 2 it is shown that multicast is preferred for NN emulation on mesh NoC, as summarized in Tables 1 and 2. In Section 3.5, we have analyzed the cost and performance of NoC tree, AER shared bus and point-to-point connectivity and compared performance and cost to MC mesh NoC, as summarized in Tables 1 and 3.

It is evident from Table 3 that mesh NoC with MC communications is preferred for large-scale configurable VLSI implementation of neural networks. It offers the highest performance/cost ratio, provides a high bandwidth which, thanks to high level of parallelism, grows with the size of the network, and demonstrates best preservation of relative latencies. Only the maximally parallel P2P provides a higher bandwidth than the mesh NoC, but at an extremely high cost. The shared bus and fat tree implementations are less favorable.

One method of improving bandwidth of the bus and tree topologies is to use pipeline registers. However, when pipelined, the tree could be considered a mesh with additional n^2 registers, emulating the connectivity of the fat tree, and the bus would practically become a broadcasting mesh.

4. Performance simulations

The mesh NoC architecture was simulated using UC, MC and BC in order to validate the analytical model and to gain insight into network behavior. We employed the NS2 Network Simulator [21] to investigate our neural networks. The link frequency $f_{\text{NoC}} = 1$ GHz and the packet header is 10 bit. The network size n (n processors emulating one neuron each) was varied from 25 to 196 and higher in some cases. For each n , a MC and a UC mesh NoCs are simulated with both Hopfield and RNDC connectivity patterns, using Poisson firing rate. The realistic connectivity parameters of RNDC (34) were employed. For each simulated network, the firing rate was varied in search of a “knee point” (Fig. 7), in which contentions became significant and the average network delay started to grow exponentially. For lack of analytical identification of that knee point, we place it at the point where the average delay doubles relative to its initial value. This metric is selected as the maximal firing frequency. We then examine the dependence of the maximal firing frequency on network properties, validating our analytical model.

Comparing Hopfield NN (top row of Fig. 7) with RNDC (bottom row), we observe that the full connectivity of Hopfield networks comes at the price of higher average delay and lower achievable firing frequencies. Considering unicast (left-hand side of Fig. 7) versus multicast (right-hand column), it is evident that multicast enables higher firing rates. Notice, that while above the knee point the delays are intolerable, below it the delays seem constant. This validates the model assumption that the network effectively operates with no congestion. Notice further that if each processor emulates k neurons, then the maximum firing rate would decrease proportionately by k .

The maximum firing rates of a Hopfield NN achieved for the simulated values on n is shown in Fig. 8a. The firing rate achieved using UC is lower than MC and BC, and it decreases faster with n . The simulations indeed validate that the MC firing rate behaves approximately as $O(1/n)$ and UC frequency behaves as $O(1/n\sqrt{n})$.

Similar results for RNDC NN are shown in Fig. 8b. MC achieves higher spiking rate than both UC and BC, and the rate scales better with network size. The simulations indicate that MC spiking rate ranges between $O(n^{-0.5})$ and $O(n^{-0.8})$ depending on the specific connectivity pattern. BC spiking rate is $O(1/n)$, the same as for Hopfield NN, and UC scales slightly better than BC.

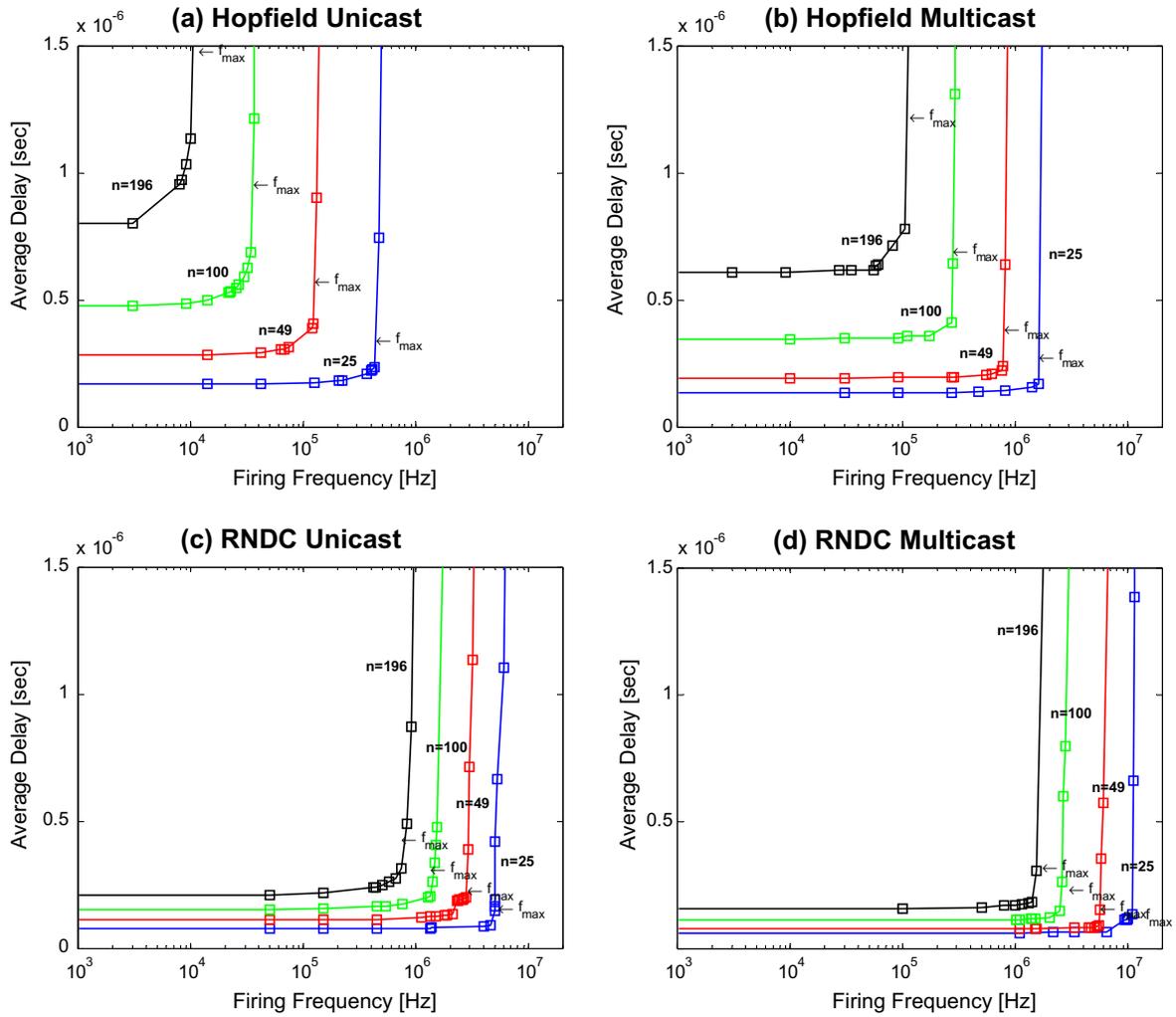


Fig. 7. Average end-to-end delay vs. firing frequency on a mesh NoC: (a) Hopfield NN, unicast, (b) Hopfield NN, multicast/broadcast, (c) RNDC NN, unicast, and (d) RNDC NN, multicast/broadcast.

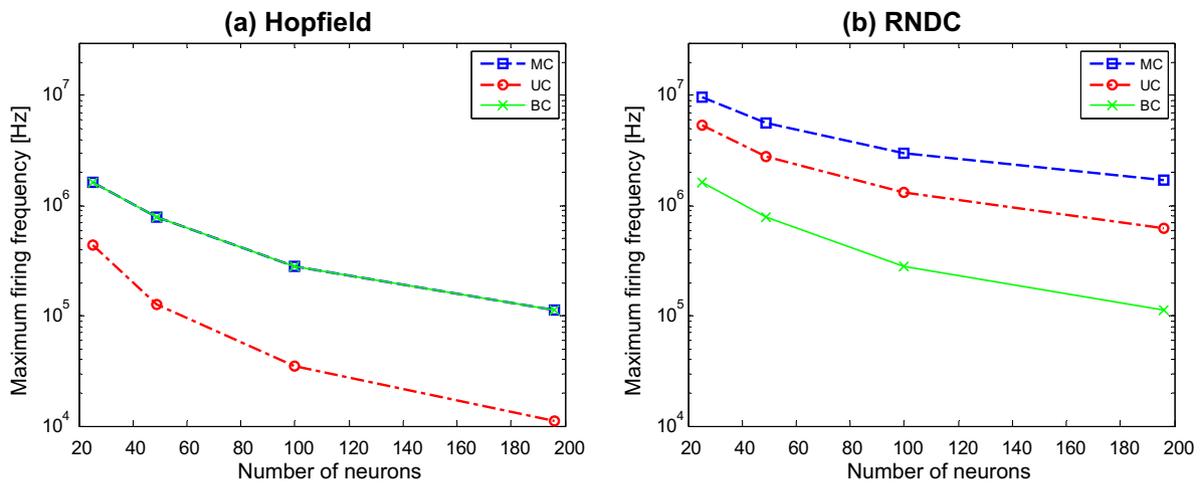


Fig. 8. Maximal firing frequency vs. network size on a mesh NoC, comparing UC, MC and BC. (a) Hopfield NN. (b) RNDC.

The delivered vs. maximal offered spike ratio K is derived from simulations. Supporting the results in Section 3.2 the maximal offered frequency, which indicated biologically inspired maximal fir-

ing rate of the neuron, in most of the cases is higher than maximal delivered frequency, which indicates NoC capacity. Example of the phenomenon is shown in Fig. 9: for each NN size, the maximal of-

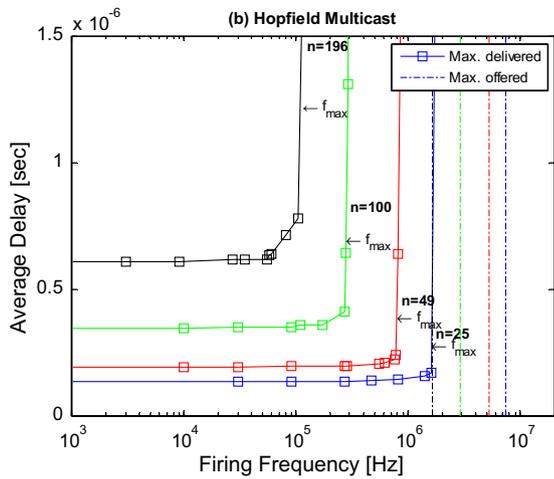


Fig. 9. Maximum offered firing rates (dashed lines) exceed maximum delivered rates.

ferred spike frequency (marked by dashed lines) is higher than the respective knee point (the maximal delivered spike frequency).

Although the maximal offered spike frequency may be a good theoretical metric, it does not provide a realistic estimation of the actual average spiking frequency in a real ANN. In fact, if all neurons constantly fire in their maximum rate, the network becomes saturated and cannot encode any information or perform any computational task. Thus, the average neuron firing rate never reaches the maximum, to allow effective information encoding. We arbitrarily choose a network with average neuron firing rate of one tenth of the maximal rate and about two decades of rate variance. The average offered frequency vs. network size is shown in Fig. 10, and compared with the maximal delivered rate in Fig. 11. For small network size, the ratio is higher than one, implying that NoC provides enough capacity to emulate more than one logical neuron in the neural processor. For example, emulating RNDC NN with 50 neural processors on MC mesh NoC, one can allocate four neurons to each processor. As the network size grows, the delivered vs. offered ratio decreases for all simulated cases, which agrees with the analytical model. The best ratio (declining at the slowest rate), is achieved with RNDC model emulated on MC mesh NoC. This result also fits the theoretical analysis. The network size yielding a delivered/offered ratio of unity indicates the point beyond which the offered frequency cannot be reached even with one neuron per processor. With the broadcasting scheme, the ratio of one is met with NN size of one hundred. Other curves are extrapolated to find

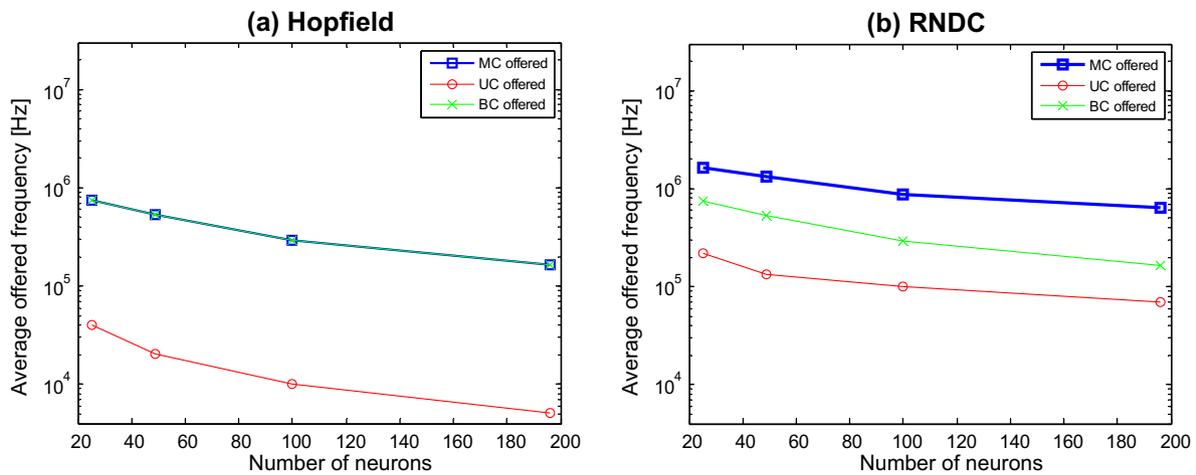


Fig. 10. Average offered spiking rate of (a) Hopfield and (b) RNDC networks.

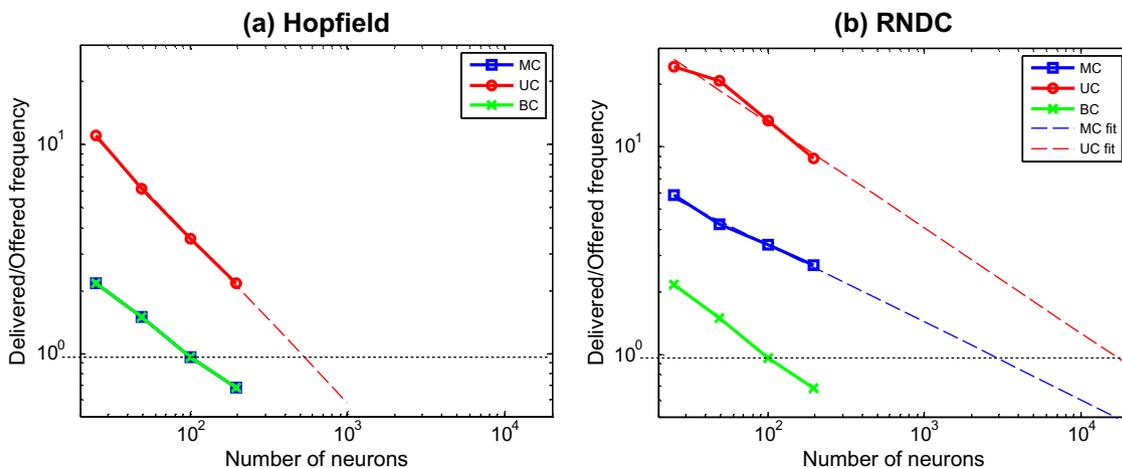


Fig. 11. Delivered-to-offered rate ratio for (a) Hopfield and (b) RNDC networks.

this point: 500 for UC Hopfield, 20,000 for UC RNDC and 3000 for MC RNDC. Note that while the ratio for UC appears better than for MC, it is misleading because the offered rate of UC is actually very low (Fig. 10).

This paper focuses on finding the maximal number of logical neurons that one physical processor can implement, assuming the network is maximally utilized. Another interesting problem is the trade-off in allocating area and power to processors versus the network when the target number of logical neurons in the NN is pre-defined. Should we implement only a single neuron in each processor and a large network with many processors, or should we implement a very large number of neurons in each processor and a small network? The more neurons in a processor, the less bandwidth is required from the interconnect but at the cost of reduced parallelism. The optimum point in this trade-off is largely a function of the processor architecture.

5. Routing tables for NN architectures

5.1. Routing table and packet header size analysis

Neural network interconnect architecture should accommodate high element count. To achieve that, it should be scalable, modular and extendable. Naturally, the size of routing tables and the ensuing power and area grow with the network size. Moreover, the time required to access each table, which affects performance, depends on its size and, thus, on the network size. For NoC architectures, the cost of the routing tables can influence architectural decisions such as the routing method (source vs. distributed routing). Consequently, routing table size is considered an important metric in comparing interconnect architectures. In this section we discuss storage requirements of the proposed architectures and routing methods, comparing them to each other and to theoretical bounds [27].

For Hopfield network emulation, routing tables are not needed, since the network is expected to communicate the spike to all destinations, and it can be achieved with simple routing algorithms. However, in the general case, for each source neuron N_i there is a group of K_i destination neurons so that $0 \leq K_i \leq n$, where n is a total number of processors. With RNDC model, the probability of any neuron i being connected to any other neuron j is proportional to $C \exp(-D(i,j)/\lambda)$, where D is the Euclidian distance between the two neurons. This connection probability is always non-zero and any spatial connectivity is possible. Designing for the worst case (WC) requires the support of arbitrary connectivity. In practice, however, only a more limited connectivity should be supported. In the following we first estimate sizes of the routing table and the packet header, as functions of the routing schemes. We then analyze possibilities for improving these general estimates.

Routing of the shared bus and P2P architectures is simple. In the case of bus architecture, each event (spike) is distributed to all destinations, and all processors “fish out” the events that are addressed to them. The routing table is located at the destination side. It stores the information on all the sources connected to that destination. The routing table size is thus $O(n)$ for each destination, and $O(n^2)$ for the entire system. The packet header size is $O(\log n)$.

With P2P architecture, all neurons are connected directly to each other. The routing table is located at the source and describes the set of destinations to which the events should be sent. Thus, its size is $O(n)$ for each source and $O(n^2)$ for the entire system. The packet header is $O(1)$ since there is no need to describe either the source or the destination address.

For NoC, the routing and addressing methods affect the storage requirements of the routing tables. In the most general case, a NN implemented on a UC mesh NoC requires routing table of $O(n)$ at

each source processor and $O(n^2)$ entries for the entire system. In this case one bit per entry is sufficient to encode a connection, and n^2 entries imply n^2 bits. The packet header can be $O(\log n)$.

Two routing methods are considered for MC mesh NoC, source and distributed routing. When using source routing, the tables are located in the source processors and the packet header describes the routing path. Similarly to UC mesh NoC, in the general case there are n entries in each source and n^2 bits in total. The packet header is $O(n)$ bits.

When distributed routing is employed with source addressing [18], the routing tables are located in the routers, and the packet header contains the source address. In the worst case there are n entries in each router. Each entry is four bits long when a typical five-port router is used. As a result, there are $O(n^2)$ bits for all routing tables, while the header is only $O(\log n)$ bits long.

Broadcasting on mesh NoC is very similar to shared AER bus. All destinations receive all packets and need to screen out the relevant ones. The source neuron sends its address in the packet header and each destination keeps a table listing all connected sources. In general, the required storage is n^2 bits and the packet header is $\log(n)$ bits long.

The RT size and packet header size analysis is summarized in Table 4.

To conclude, regardless of architecture, communication and addressing, the general case calls for $O(n^2)$ memory storage for all routing tables. When employing MC mesh NoC (as suggested in previous sections), distributed routing offers a shorter packet header of only $O(\log n)$ bits. The next section discusses routing table and packet header sizes in practical implementations.

5.2. Fixed RT size and bounded NN connectivity

The practical meaning of the foregoing analysis is that a choice of memory depth and packet header size in a specific implementation predefines the possible connectivity allowed by the implementation. Scaling the network up may require redesign of the implementation. Alternatively, the scaled-up network will not be able to implement arbitrary connectivity. In particular, choosing a source routing scheme and allocating m entries for the routing table in each source allows the implementation of only a sub-group of neural networks where the maximum number of destinations per each neuron is bounded by m . Similarly, choosing a broadcasting method with m -size source table in each destination allows implementing another sub-group of neural networks where the maximal number of sources for each neuron is also bounded by m .

In order to analyze the implication of limiting the memory size for a mesh with MC distributed routing, we consider turn-table routing [30] configured for distributed multicast. The packet is tagged by the source address and the router directs the packet straight through (default routing) unless it finds the source address in its routing table, in which case the packet needs to either sink, turn or split in that router. Assuming that each source is connected on average to only m destinations, then on average $2m$ entries per router are needed, since each packet per destination typically

Table 4
Routing table and packet header size (asymptotic $O(\cdot)$).

Arch	RT size			Total storage	Header size
	Source	Dest	Router		
AER bus	0	n	0	n^2	$\log n$
P2P	n	0	0	n^2	1
NoC UC	n	0	0	n^2	$\log n$
NoC BC	0	n	0	n^2	$\log n$
NoC MC SR	n	0	0	n^2	n
NoC MC DR	0	0	n	n^2	$\log n$

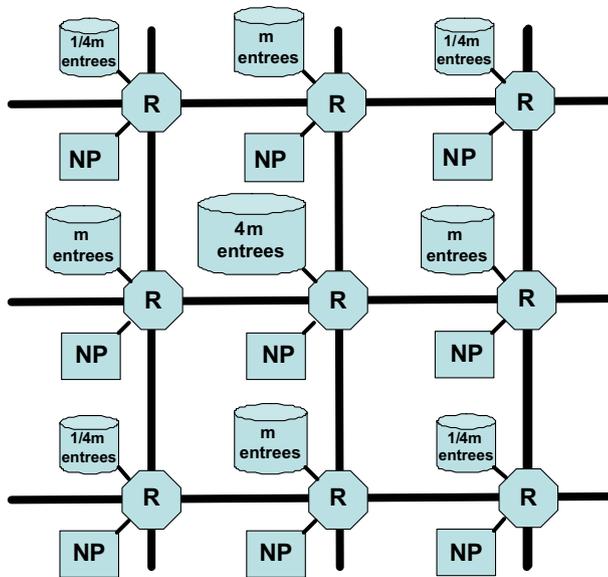


Fig. 12. Non-uniform table size example.

needs one entry to affect a turn and a second entry to mark the sink, totaling $2m \times n$ storage. Such an implementation constrains the actual NN topologies that can be implemented to networks with only up to $2m$ sources per destination.

In realistic neural networks with probabilistic connectivity rule that yields uniform traffic, such as the RNDC model, the connectivity is localized and the probability of connection decreases exponentially with the distance from the source. Targeting only this subset of topologies, we achieve smaller routing tables by allocating routing table storage non-uniformly. For example, in Fig. 12 the routers are divided into groups of nine routers each and within each group the storage can be allocated as follows: four routers with $m/4$ entries, four routers with m entries, and one router with $4m$ entries, totaling $9m$ entries. Overall storage size remains m routing table entries per router, but now one can emulate neural network topologies where the maximal number of sources for some neurons can be up to four times higher than m . Routing table utilization can be further enhanced and maximal connectivity bound can be further extended by combining a number of individual tables into a single shared memory, e.g. all nine tables of Fig. 12 combined into one memory of $9m$ entries.

6. Modular hierarchical NoC architecture

In the previous section it was shown that fixing the memory size of the routing tables in a mesh NoC improves scalability at the expense of bounding the connectivity. In this section we limit the discussion to networks with bounded exponential connectivity and propose a modular, hierarchical architecture that enables emulating large scale NNs. The architecture is based on MC mesh NoC with distributed turn-table routing, one flit packets containing only the source address, and non-uniform storage for the routing tables.

Consider a network of up to 1 million neurons, enough to emulate a large neural microcircuit (~ 100 cortical hypercolumns [37]). The source address, and thus the packet size, is 20 bits. Empirical data from neurological research suggests that the average number of connections per neuron in the cortex ranges between 1000 and 3000 [37]. We limit the connectivity of the emulated network by fixing the routing table size to 1000 connections per neuron on average. The maximal number of connections, following the example in Fig. 12 is 4000.

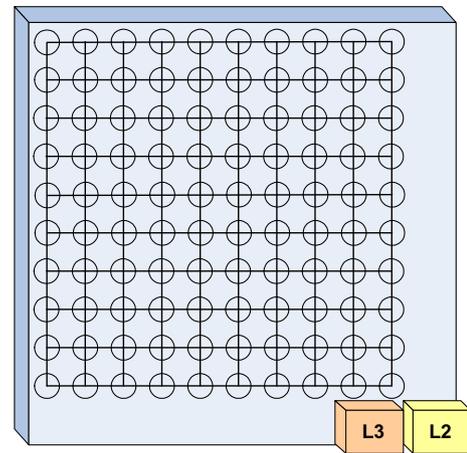


Fig. 13. Hierarchical architecture: basic SoC.

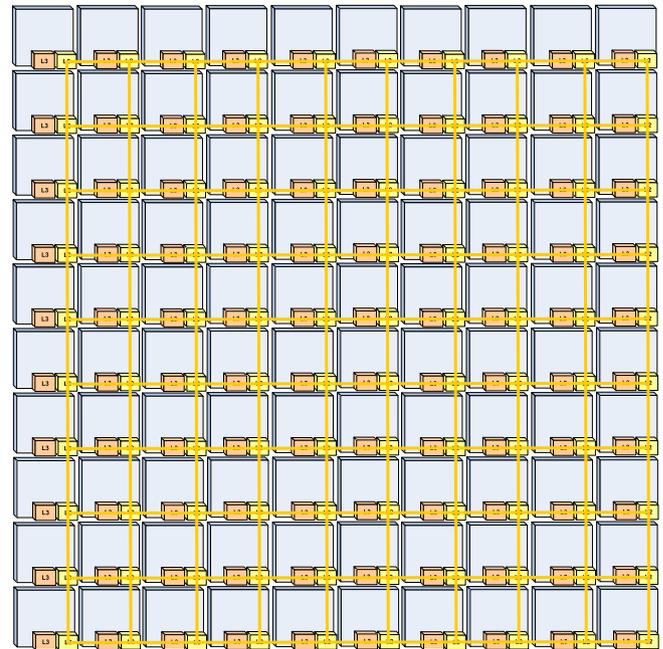


Fig. 14. Hierarchical architecture. 2nd level: multiple SoC on a single board.

The one million neurons network can be constructed as follows. A single SoC of 100 processors is designed. One hundred such chips can be placed on a single board, and 10 such boards are perceived in a single rack, totaling 100,000 processors in this modular hierarchical system. Technology scaling, while retaining this architecture, may help emulate a growing number of neurons per SoC. For instance, 10 neurons per processors will facilitate a one million neurons in the emulation.

The architecture is modular and scalable thanks to pre-design of routing that enables one million neurons in such a structure, as demonstrated in Figs. 13–15. Obviously, each SoC implements a mesh NoC including the necessary routers. To access neurons on other chips, each SoC also contain a SoC-to-SoC 2nd level router (at the board level). To access neurons on other boards, each SoC also contains a Board-to-Board 3rd level router, but only one such router is activated on each board. Thus, a single SoC architecture enables interconnect at all three levels.

Uniform traffic implies $10\times$ bandwidth capacity on each 2nd level link, since each 2nd level link substitutes for ten regular links in

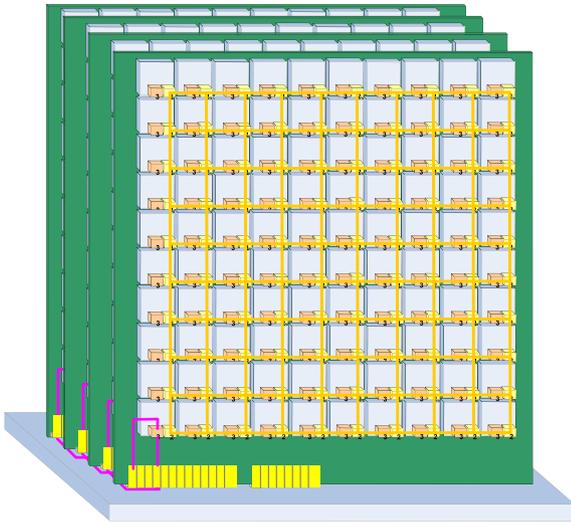


Fig. 15. Hierarchical Architecture. 3rd level: multiple cards on a single rack.

a flat topology. In addition, the 2nd level router requires $40 \times$ routing storage, since it provides a turn table for traffic from the forty 1st level nodes around the periphery of the chip. The storage and bandwidth requirements of the 3rd level router are about three times higher than those of the 2nd level, but a practical implementation shares the same large storage between the 2nd and the 3rd level routers. Bandwidth increase in 2nd and 3rd level routers may constitute a bottleneck. The required increase in throughput is achieved by special higher speed links. The total storage consumed by the routing tables in one chip is $100 \text{ routers} \times 1000 \text{ entries per router} \times (20 \text{ address bits} + 4 \text{ turn bits}) = 2.4 \text{ Mbit}$, plus 1 Mbit for the 2nd and 3rd level routers.

7. Conclusion

This paper presents a theoretical comparative analysis of interconnect architectures for general purpose configurable emulation of spiking neural networks. We show that a mesh NoC is preferred over the other analyzed architectures (fat tree, point to point and shared bus). Further, it is shown that multicast communications outperform unicast and broadcast. Simulations successfully validate the analytical models and the asymptotic behavior of the network as a function of its size. The results may be extended to the more general case of address event transactions. It is shown that regardless of the routing method, routing table size grows at $O(n^2)$ and limits network scalability. Bounding the size of the tables facilitates scalability. We propose a multi-level system with modular hierarchical NOC architecture for large-scale neural network emulation.

To conclude, the properties of spiking networks, such as planar-like structure, fault and drop tolerance and pulse-information encoding makes simple multicast mesh network-on-chip suitable for massively parallel communication required by these networks. Moreover, since NoC is designed to be extendable, the same architecture can be used to emulate large scale spiking NN.

Future research may address the following questions: the architecture of the neural processor, its effect on the NoC, the trade-off in allocating area and power to processors versus network, and the architecture of multicast NoC routers optimized for emulating spiking neural networks. In addition, simulations may be extended to cover networks of many thousands of neurons.

Acknowledgements

We are grateful for contributions to this work by Isaac Keslassy. This work was partly supported by the European Research Council Starting Grant No. 210389.

References

- [1] R. Legenstein, W. Maass, Edge of chaos and prediction of computational performance for neural microcircuit models, *Neural Networks* 20 (3) (2007) 323–334.
- [2] C. Leiserson, Fat-Trees: universal networks for hardware-efficient supercomputing, *IEEE Transactions on Computers* C-34 (10) (1985) 892–901.
- [3] S.R. Ohring, M. Ibel, S.K. Das, M.J. Kumar, On generalized fat-tree, in: *Proceedings of the International Conference on Supercomputing*, 1995, pp. 276–285.
- [4] R. Rojas, *Neural Networks: A Systematic Introduction*, Springer-Verlag, Berlin, 1996, pp. 338–371 (Chapter 13).
- [5] W. Maass, Networks of spiking neurons: the third generation of neural network models, *Neural Networks* 10 (9) (1997).
- [6] T. Morie, M. Nagata, A. Iwata, Design of a pixel-parallel feature extraction VLSI system for biologically-inspired object recognition method, in: *Proceedings of the International Symposium on Nonlinear Theory and its Application*, 2001, pp. 371–374.
- [7] J. Schreier et al., Cellular pulse coupled neural network with adaptive weights for image segmentation and its VLSI implementation, in: *Proceedings of the International Symposium on Electronic Imaging: Science and Technology*, vol. 5298, 2004, pp. 290–296.
- [8] G. Indiveri, E. Chicca, R. Douglas, A VLSI reconfigurable network of integrate-and-fire neurons with spike-based learning synapses, in: *Proceedings of the European Symposium on Artificial Neural Networks*, 2004, pp. 405–410.
- [9] C. Mayr et al., large-scale spiking neural networks on VLSI, *International Journal of Applied Science Engineering and Technology* 4 (1) (2007).
- [10] A. Mortara, E. Vittoz, P. Vanier, A communication scheme for analog VLSI perceptive systems, *IEEE JSSC* 30 (6) (1995) 660–669.
- [11] K.A. Boahen, Point-to-point connectivity between neuromorphic chips using address events, *IEEE Transactions Circuits and Systems* 47 (5) (2000).
- [12] S.C. Liu et al., Orientation-selective VLSI spiking neurons, *Neural Networks* 14 (2001) 629–643.
- [13] J. Lazzaro, J. Wawrzyniec, A multi-sender asynchronous extension to the AER protocol, in: *Proceedings of the Conference on Advanced Research in VLSI*, 1995, pp. 158–169.
- [14] W.J. Dally, B. Towles, Route packets not wires: on-chip interconnection networks, in: *Proceedings of the 38th Design Automation Conference*, 2001, pp. 684–689.
- [15] A. Jantsch, *Networks on Chip*, Workshop at the European Solid State Circuits Conference, 2001.
- [16] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, *IEEE Computer* 35 (2002) 70–78.
- [17] T. Theocharides, G. Link, N. Vijaykrishnan, M.J. Irwin, A generic reconfigurable neural network architecture as a network on chip, in: *Proceedings of the International SOC Conference*, 2004, p. 191.
- [18] L.A. Plana, S.B. Furber, et al., A GALs infrastructure for a massively parallel multiprocessor, *IEEE Design & Test of Computers* 24 (5) (2007) 454–463.
- [19] R. Emery, A. Yakovlev, G. Chester, Connection-centric network for spiking neural networks, in: *Proceedings of the NOCS*, 2009, pp. 144–152.
- [20] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, Cost considerations in Network on Chip, *Integration* 38 (1) (2004) 19–42.
- [21] S. McCanne, S. Floyd, ns Network Simulator. <<http://www.isi.edu/nsnam/ns/>>.
- [22] N. Eislery, L.S. Peh, High-level power analysis for on-chip networks, in: *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2004, pp. 104–115.
- [23] U.Y. Ogras, R. Marculescu, Analytical router modeling for Networks-on-Chip performance analysis, in: *Proceedings of the DATE*, 2007, pp. 1096–1101.
- [24] F. Gilbert, S. Medardoni, D. Bertozzi, L. Benini, et al., Exploring high-dimensional topologies for NoC design through an integrated analysis and synthesis framework, in: *Proceedings of the NOCS*, 2008, pp. 107–116.
- [25] A. Banerjee, R. Mullins, S. Moore, A power and energy exploration of Network-on-Chip architectures, in: *Proceedings of the NOCS*, 2007, pp. 163–172.
- [26] D. Peleg, E. Upfal, A trade-off between space and efficiency for routing tables, *Journal of the Association for Computing Machinery* 36 (1989) 510–530.
- [27] J. Van Leeuwen, R.B. Tan, Interval routing, *The Computer Journal* 30 (1987) 298–307.
- [28] M.E. Gómez et al., A memory-effective routing strategy for regular interconnection networks, *IPDPS*, 2005.
- [29] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, Routing table minimization for irregular mesh NoCs, in: *Proceedings of the Design, Automation and Test in Europe Conference*, April 2007, pp. 942–947.
- [30] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* 5 (1943).
- [31] W. Maass, C.M. Bishop, *Pulsed Neural Networks*, MIT Press, Cambridge, MA, 1999 (Chapters 1, 2, and 4).
- [32] E.M. Izhikevich, Which model to use for cortical spiking neurons?, *IEEE Transactions on Neural Networks* 15 (5) (2004).

- [34] J.G. Milton, Dynamics of small neural populations, American Mathematical Society, Providence, RI, 1996 (Chapter 8).
- [35] D. Goldberg, G. Cauwenberghs, A. Andreou, Analog VLSI spiking neural network with address domain probabilistic synapses, in: Proceedings of the IEEE International Symposium Circuits and Systems (ISCAS2001), IEEE Press, 2001, pp. 241–244.
- [36] Ulrich Ramacher, SYNAPSE: a neurocomputer that synthesizes neural algorithms on a parallel systolic engine, *Journal of Parallel and Distributed Computing* 14 (3) (1992) 306–318.
- [37] C. Johansson, A. Lansner, Towards cortex sized artificial neural systems, *Neural Networks* 20 (2007) 48–61.
- [38] M.J. Pertel, Mesh Distance Formulae, Technical Report, California Institute of Technology (CaltechCSTR:1992.cs-tr-92-05).
- [39] A. Jantsch, H. Tenhunen, *Networks on Chip*, Kluwer Academic Publishers, Hingham, MA, 2003.
- [40] K. Goossens et al., *Guaranteeing the quality of services in networks on chip*, *Networks on Chip*, Kluwer Academic Publishers, Hingham, MA, 2003.



Ran Ginosar received his BSc from the Technion and his PhD from Princeton University. After conducting research at AT & T Bell Laboratories, he joined the Technion where he is now an Associate Professor at the Electrical Engineering and Computer Science departments, also heading the VLSI Systems Research Center. Professor Ginosar has been a visiting Associate Professor with the University of Utah and co-initiated the Asynchronous Architecture Research Project at Intel (Oregon), where he worked during a two-year sabbatical on Intel's asynchronous test chip. He has co-founded a number of VLSI companies. He has published numerous papers and patents on VLSI. His research interests include VLSI architecture, asynchronous logic and synchronization.



Dmitri Vainbrand received his BSc from the Technion and he is at the final stages of conducting his masters at the Technion. He is working at Intel Israel since 2003 in the field of signal integrity and signaling analysis methodologies and high-speed interconnect design. His research interests include hardware architecture for neural networks, configurable interconnect architectures and networks on chip.