Network-on-Chip Evaluation for a Novel Neural Architecture

DOI:
10.1145/3203217.3203268

Document Version
Accepted author manuscript

Network-on-Chip Evaluation for a Novel Neural Architecture

Markos Kynigos, Javier Navaridas, Luis A. Plana, Steve Furber
School of Computer Science
The University of Manchester

Manchester, United Kingdom

ABSTRACT

This paper provides a performance evaluation and trade-off analysis of a novel chip architecture for neuromorphic computing, especially focused on the memory subsystems and the Network-On-Chip (NoC). More precisely, we study the performance-related effect of the number of memory modules, as well as that of allowing direct core-to-core communication. Our simulation-based experimental work throws many interesting results on the above aspects and allows to ensure that congestion at the NoC-level is unlikely to degrade performance.

CCS CONCEPTS

Computer systems organization → Interconnection architectures

Computer systems organization → Neural networks

KEYWORDS

Biologically Inspired Architecture, Neuromorphic Computing, Networks on Chip, Performance Evaluation, Systems on Chip.

1 INTRODUCTION

In the era of high-performance computing, industry leaders are recognizing the slow-down of the 50-year long trend known as Moore's Law. The strategy of increasing computing power via adding more logic into individual chips has had to change significantly, as transistor miniaturization is approaching physical limits. Moreover, the Power Wall limits the amount of logic that can be active at the same time within silicon-based chips. A variety of solutions for this problem are currently prevalent; using custom ASIC systems or including FPGA technology are popular techniques and research for novel manufacturing materials that can exhibit better characteristics for computation (e.g. using MoS_2) [1] is a topic of interest. Nevertheless, currently the most widely adopted practice is to use more, simpler processors through parallelism.

However, high-performance computing (HPC) requires massive processor counts, and conventional parallelization is hard-pressed to deal with the problem, leading to the need for novel parallelism paradigms.

In addition, the significant increase in computing power exhibited over the past few decades has invigorated the field of machine learning, especially neural networks research. Neural networks are highly parallelized structures based on the workings of nervous systems, which show great potential for modelling biological constructs such as brains [2]. At the same time, neurobiology and human-brain research are progressing at a rapid pace. This has led to further understanding of the most massively parallel computational unit known to human kind, a mammal's brain. The domain's knowledge is being applied to the computing world to create new types of computers. These biologically inspired computers (i.e. neuromorphic computers) could provide the computing community with the paradigms required for continued progress, while at the same time contributing our understanding of the human brain.

Many projects that involve large-scale neuromorphic computing have been developed over the past few years, e.g., SpiNNaker [3], the Blue Brain project [6] or Neurogrid [10]. While these systems can perform simulations with millions of neurons, they still only represent a small proportion of the human brain (about 2 orders of magnitude less), so new architectures and modelling systems are needed.

Our goal within the SpiNNaker2 project is to produce a machine capable of modelling at least ten times the number of neurons as these systems. This would allow for models scaling up to 10% of the human brain in terms of neuron count, thereby enabling further research into more sophisticated aspects of neural models. For this reason, our design aims at providing improved scalability, at increasing the communication bandwidth per card, at extending the support for software applications (e.g. support more neuron models and features, more synaptic plasticity models etc.) and at providing low-power configurations for robotics. This can be achieved through higher orders of parallelism throughout the system and especially within our System-on-Chip (SoC). We are putting together over a hundred low-power cores to provide these improvements. However, we are still in the earlier design phases and core counts and the final architecture is not decided

yet. Indeed, this paper's objective is to perform a design space exploration of the communications and memory to ascertain the trade-offs of their different aspects.

More specifically, we aim to understand the effects of the number of memories on the performance of the memory and interconnection subsystems. We also investigate the trade-offs incurred when allowing processing cores to communicate directly with each other, rather than allowing them to communicate only indirectly, through memory.

2 RELATED WORK

Modelling the entire human brain requires levels of computation that have not been achieved yet. Estimations suggest that a rate of 10^{18} operations per second is required for modelling a human brain [4][5], or the performance of an exa-scale machine.

Additionally, conventional HPC systems are not optimized for the type of communication that occurs in biological systems, i.e. broadcasting many small pieces of information to many recipients, but rather for point-to-point transmission of large blocks of data. This leaves HPC machinery at a disadvantage within the context of neural modelling. Therefore, novel architectures that can handle the intricacies of the neural problems are imperative.

2.1 Neuromorphic Computing

A variety of approaches exist for modelling biological neural networks through *software* (i.e. neural models), sometimes run on top of HPC platforms. For instance, the EPFL Blue Brain project combines traditional neuroscience with HPC to model biological neural systems [6] using an IBM Blue Gene supercomputer. Another example is the Izhikevich point-neuron model, developed at the Neuroscience Research Institute in San Diego, which consists of 100 billion neurons [7]. However, as stated above, neural simulations do not fit very well to current computing systems, leading to many other projects that aim to develop new *hardware* approaches.

The Stanford Neurogrid project's neuromorphic hardware can emulate 1 million neurons in real time, approximately the capability of 200 BlueGene racks [8], while being able to scale the modelling capacity to 64 million neurons. The University of Manchester's SpiNNaker machine is a massively parallel, special purpose computer designed for low-power neural network simulation [9]. SpiNNaker's main innovation is its multicast interconnection network which perfectly meets the communication needs of the neural application.

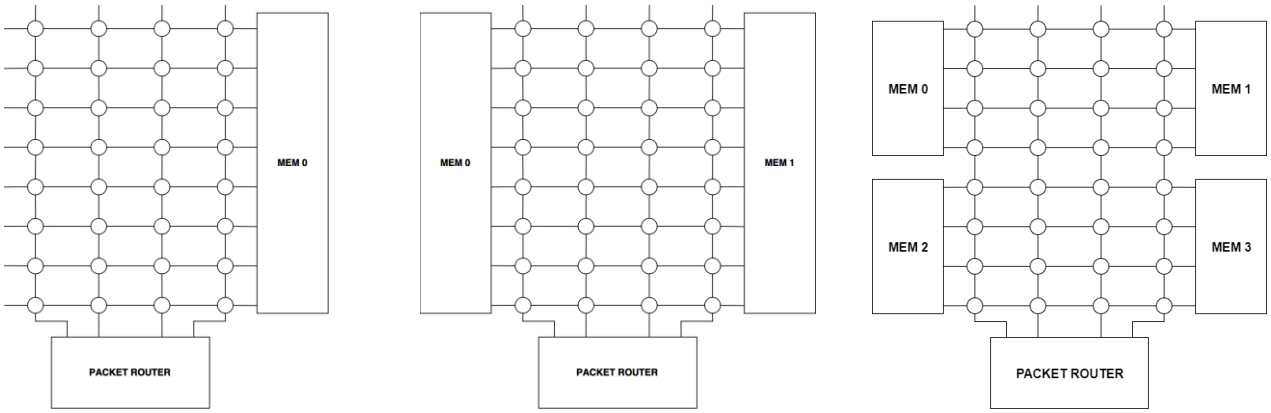
2.2 Many-Core SoCs

In principle, the mesh-like architecture we propose (detailed in next section) follows a similar approach to most many-core chips available in the market, such as the Tilera Tile64 [10], the Intel Xeon-Phi Knights Landing family of processors [11] or the Sunway's SW26010 [12]. All of them are devised as an array of processor cores with all the needed subsystems (including I/O) connected to the boundaries of the chip, while relying on mesh-like interconnects to perform inter-chip communication. However, there exist many differences between the above SoCs and our SpiNNaker2 SoC. The most important is related to data coherence and the way it affects the workloads the interconnect needs to support.

Both the Tile64 and the Xeon Phi, rely on traditional cache coherence architectures and so their NoCs have been optimized accordingly, by using very small control packets together with slightly larger data packets, most of which are originated in (and addressed to) the cores. The Sunway SW26010 attempts to avoid coherence altogether by isolating the dedicated cache in each core (scratch pad memories) and not allowing inter-processor communication (IPC). Indeed, one of the most significant design decisions for us is whether to allow direct IPC, as this would greatly assist neural simulations, or to stick to IPC being carried out through main memory only.

3 ARCHITECTURE DESCRIPTION

The central compute element of the SpiNNaker2 NoC is a Quad-Processing Element, or QPE, comprised of four processing



a) Single-memory design **b) Dual-memory design** **c) Quad-memory design**

Fig. 1 Proposed NoC Designs

modules and an AHB bus crossbar that includes a DMA controller and a NoC router. Each module contains an ARM Cortex M4F processor [13], an SRAM module of 128KB, single-precision floating point hardware, a fixed-point exponential accelerator and a random number generator. QPEs will be connected to form a mesh NoC to which external modules will be connected. However, we are still investigating the optimal component arrangement as well as the features the NoC interface and infrastructure need to support. Currently, we are looking at three different designs for the SpiNNaker2 SoC, presented in Fig. 1 above. All three topologies comprise of a 4×8 QPE mesh, a Packet router connected to the south edge of the mesh and either 1, 2 or 4 memory controller modules to access external memories. The memory modules are connected to the eastern and western edges of the mesh.

The main purposes of our evaluation are two; to see whether the single-memory module architecture creates a bottleneck on the eastern edge of the mesh, as the links on the eastern edge would need to accommodate traffic from the whole row, and to assess the effect of allowing for inter-QPE communication. In the first case, we want to assess to what extent adding more modules would affect performance. In principle, we are assuming a localized use of the memory modules in which QPEs will access their closest memory, effectively splitting the SoC into halves or quadrants (i.e. for Fig. 1c, all north-western QPEs communicate with memory module 0, north eastern ones with memory module 1 etc.).

4 EXPERIMENTAL ANALYSIS

In order to understand these performance aspects and explore the design space of the interconnection and memory subsystem, we modelled all our different architectures in INSEE [14], an open-source, time-accurate network simulator.

4.1 Experimental Setup & Methodology

Although the architecture of the NoC is not defined yet, we considered traditional crossbar-based routers using the bubble strategy for VC management [15], with a bubble size of 2. For simplicity we use oblivious dimension-order routing and Round-Robin arbitration. Also, given that no large bursts are expected we apply no congestion control to the traffic. This is based on practicalities rather than design limitations, since not applying congestion control will help us to see whether there is any performance degradation upon saturation, in which case we could explore such mechanisms. We modelled packets of fixed length (16 phits of 4 bytes). In our simulations, each QPE is abstracted into a single traffic generation instance.

The traffic patterns we use require special consideration due to the atypical communication needs of the machine. Traffic is injected by the nodes into the network using a Bernoulli process with a variable injection rate. Traffic distribution is regulated by two parameters that control the ratio of QPE-to-Memory (Q2M) and QPE-to-Router (Q2R), with the rest of the traffic representing QPE-to-QPE (Q2Q) traffic, which is uniformly distributed across the mesh. Packets addressed to the memories will be sent to the nearest memory using only horizontal links, while packets addressed to the Packet Router will travel vertically within the same column of the injecting QPE. Traffic is reactive, meaning that packets arriving to the memories trigger a response packet to the source. Packets addressed to the Packet Router trigger new packets to one QPE at random. To perform our analysis, we used the maximum injection load and set the Q2R traffic ratio to be 5% of the traffic. The rest of the traffic iterates over the Q2M ratio, in order to compare Q2Q and Q2M in terms of network throughput.

4.2 Effects of QPE-QPE Communication

The impact of Q2Q communication on the NoC's throughput for the 3 designs is shown in Fig. 2. Clearly, Q2Q communication can be advantageous in all cases. As the Q2M ratio decreases (and therefore the Q2Q ratio increases), we see a gradual performance increase which is most pronounced for the single-memory architecture. For more than half (two thirds in the single-memory case) of the traffic being sent among the QPEs, however, the throughput of the NoC starts decreasing. The dual- and quad-memory counterparts also exhibit significant gains as the Q2Q ratio increases; however, in both cases, using Q2Q traffic only will be counterproductive as its performance will be lower than doing all the communications through memory. Conversely, it is noteworthy that in this scenario, each Q2Q communication, which consists of 2 packet transmissions, would represent 4 packet transmissions in a Q2M-only implementation of inter-QPE communication (one QPE would write a value in memory while the other would read that value, each transaction requiring two packets to be transmitted). As such, the Q2M-only implementation would require twice the throughput to maintain the same application performance. Nevertheless, results with Q2M traffic only suggest that the mesh alone would yield a throughput of roughly 0.46 phits/node/cycle in the best case, that is without considering additional traffic scenarios where a QPE sends to more than

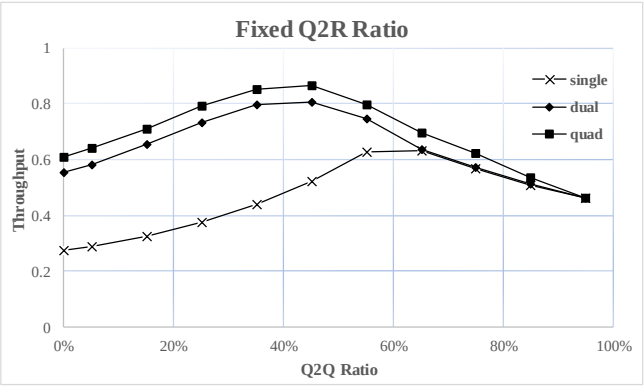


Fig. 2 Impact of Q2Q and Memory Traffic on Throughput

one memory. Therefore, Q2Q communication can be a favourable addition. With one memory, allowing Q2Q traffic always improves upon not allowing it; indeed, the curve's maximal point exhibits twice the throughput with respect to no Q2Q traffic (i.e. Q2M traffic only). For the quad- and dual-memory architectures, significant gains of up to 50% more throughput can also be achieved by allowing Q2Q traffic.

With respect to memory module usage, using a single memory is too restrictive as memory access seems to become a significant bottleneck. Using more memory modules is better in terms of throughput but moving from two memories to four yields relatively small improvements. With one memory, the throughput achieved using only the memory is a mere 0.27 phits/node/cycle. The dual- and quad-memory alternatives nearly double that by reaching 0.55 and 0.61, respectively. When considering the optimal cases in which QPE-to-QPE is enabled, this difference is not as substantial, 0.63 versus 0.8 (dual) and 0.86 (quad) phits/node/cycle respectively.

With more than one memory module in the periphery of the chip, the saturation problem presented by the single-memory alternative is alleviated, since twice as many links can be used with each link serving half the number of QPEs. Having 4 memory modules would not greatly improve the memory throughput but would allow to reduce the frequency of the memories by half, with the subsequent reduction in energy consumption. It also would allow for more graceful degradation in case of failures, as the dual-memory case would perform as the single-memory case in case of a memory fault, whereas the quad-memory would keep a more consistent behaviour. Obviously, with one memory, memory module faults would lead to the whole NoC potentially losing access to the synaptic information held in memory, causing the equivalent of a "stroke" for the neural network being modelled. Using more memory modules removes this single point of failure and allows for the introduction of more graceful failover mechanisms, where QPEs can be re-tasked to use a different memory module. This would allow them to have access to synaptic information and potentially to enable the algorithmic re-creation of lost data.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we evaluated the proposed designs for the NoC to be included in SpiNNaker2. More precisely, we have justified supporting QPE intercommunication and answered key questions about the number and configuration of memory controller modules the design should have. We have demonstrated that using more than one module is advantageous, as this should allow for higher memory throughput. Additionally, adding extra memory modules removes the risk of having a single point of failure for the chip and allows for introducing fail-over mechanisms. Finally, a failure in a single memory module would mean a complete loss of the synaptic information for the chip, something which is alleviated by introducing a form of redundancy.

This work is a first step in examining the characteristics of the SpiNNaker2 NoC. In the future, we plan to perform a more detailed evaluation and analysis of this design with respect to lower level hardware characteristics of the routers, such as buffer depth and the number of virtual channels. Additionally, we intend to examine whether adding inter-chip traffic onto the same NoC causes any significant performance issues. This is a key point of interest, as it would define whether extra NoC infrastructure is needed. We also intend to evaluate the merits of using multicast at the NoC level to better accommodate to the nature of neural traffic.

ACKNOWLEDGEMENTS

The design and construction of the SpiNNaker machine was supported by EPSRC under grants EP/D07908X/1 and EP/G015740/1, in collaboration with the universities of Southampton, Cambridge and Sheffield and with industry partners ARM Ltd, Silistix Ltd and Thales. Ongoing development of the software is supported by the EU ICT Flagship Human Brain Project (FP7-604102 & H2020-720270). Exploration of the capabilities of the machine is supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 320689. Dr. Navaridas is supported by the ExaNeSt project, which is funded by the EU's H2020 programme under grant agreement No 671553.

REFERENCES

[1] S. B. Desai et.al, "lengths, MoS2 transistors with 1-nanometer gate," Science AAAS, pp. 99-102, 7 October 2016.
[2] M. L. Forcada and R. C. Carrasco, "Finite-State Computation in Analog Neural Networks: Steps towards Biologically Plausible Models?", Lecture Notes in Computer Science, pp. 480-493, 2001.
[3] S. B. Furber, F. Galluppi, S. Temple and L. A. Plana, "The SpiNNaker Project," Proceedings of the IEEE, vol. 102, no. 5, pp. 652-665, 2014.
[4] H.-H. Suzana, "The human brain in numbers: a linearly scaled-up primate brain," Frontiers in Human Neuroscience, vol. 3, p. 31, 2009.
[5] S. B. Furber, "Brain-inspired computing," IET Computers & Digital Techniques, vol. 10, no. 6, pp. 299 - 305, 2016.
[6] H. Markram, ""The Blue Brain Project," Nature Rev. Neuroscience, vol.7, pp. 153-160, 2006.
[7] E. Izhikevich, "Simple model of spiking neurons," IEEE Transactions on Neural Networks, vol. 14, no. 6, pp. 1569 - 1572, 2003.
[8] B. V. Benjamin, P. Gao, et. al, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," Proceedings of the IEEE, vol. 102, no. 5, pp. 699 - 716, 2014.
[9] S. B. Furber et.al, "Overview of the SpiNNaker System Architecture," IEEE Transactions on Computers, pp. 2454 - 2467, 2012.
[10] D. Wentzlaff et.al, "On-Chip Interconnection Architecture of the Tile Processor," IEEE Micro, vol. 27, no. 5, pp. 15 - 31, 2007.
[11] J. Jeffers, J. Reinders and A. Sodani, Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition, Cambridge: Morgan Kaufmann, 2016.
[12] H. Fu et. al, "The Sunway TaihuLight supercomputer: system and applications," Science China. Information Sciences, vol. 59, pp. 1-16, 2016.
[13] ARM Limited, "Arm Cortex-M4 Processor, Technical Reference Manual," 2013. [Online].
[14] J. Navaridas, J. Miguel-Alonso, J. A. Pascual and F. J. Ridruejo, "Simulating and evaluating interconnection networks with INSEE," Simulation Modelling Practice and Theory, p. 494–515, 2011
[15] V. Puente et.al, "Adaptive bubble router: a design to improve performance in torus networks," in 1999 International Conference on Parallel Processing, Aizu-Wakamatsu City, Japan, Japan, 1999.