SCALPsim, a tool for modeling asynchronous Self-Organizing 3-D NoC architectures

Diego Barrientos, Claudio Sousa, Andres Upegui, Bernard Girau

▶ **To cite this version:**

Diego Barrientos, Claudio Sousa, Andres Upegui, Bernard Girau. SCALPsim, a tool for modeling asynchronous Self-Organizing 3-D NoC architectures. ICECS 2020, 27th IEEE International Conference on Electronics Circuits and Systems, Nov 2020, Glasgow/Virtual, United Kingdom. hal-02984429

HAL Id: hal-02984429
https://hal.univ-lorraine.fr/hal-02984429

Submitted on 30 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SCALPsim, a tool for modeling asynchronous Self-Organizing 3-D NoC architectures.

Diego Barrientos, Claudio Sousa and Andres Upegui
InIT, HEPIA
University of Applied Sciences of Western Switzerland
Geneva, Switzerland
Email: firstname.lastname@hesge.ch

Bernard Girau
Université de Lorraine, LORIA, UMR 7503
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, France
bernard.girau@loria.fr

***Abstract*—Manycore architectures are mainly composed of a very large amount of computing nodes interconnected with a multiplicity of links usually forming a NoC-like mesh architecture. High-speed links permit to obtain a higher throughput but are much more expensive than normal links, making the interconnection of the system a cost/performance trade-off. Simulating such architectures is very important in order to characterise the optimal network topology for a given problem. In this work we introduce SCALPsim: a simulation framework permitting to evaluate routing algorithms and network properties in 1-D, 2-D and 3-D regular mesh topologies simultaneously using links of different characteristics in terms of latency and throughput. These features are particularly interesting in large scale systems with processing elements grouped into clusters, where communication properties differ largely inside and between clusters. This paper presents the framework and an application based on Cellular Self-Organizing Maps - CSOM.**

***Index Terms*—Cellular Self-Organising Maps, parallel computation, multi-FPGA**

I. Introduction

During the last decades, parallel computing has taken the lead of general-purpose architectures, either in High-Performance Computing or in Commercial Off-The-Shelf equipments. In order to exploit this parallelism, algorithms must take into account the use of several resources concurrently. An efficient use of parallel architectures relies on efficient compilers, but also on smart and flexible architecture features that should be able to cope with unexpected situations. Self-organisation can propose a promising solution to this problem by adapting the computation resources in order to better respond to the computation requirements.

When the placement of many computing resources is performed inside a single integrated circuit, the use of Network-On-Chip (NoC) architectures becomes a straightforward solution [1]. The advantage of this approach is the separation of multiple computing resources, possibly asynchronous, with a simple architecture and a low usage of resources. Although the computing resources may not run synchronously, the NoC is usually kept synchronous, as the utilisation of resources and complexity of the modules are lower.

Nevertheless, several NoC architectures have used asynchronous elements in order to optimise the management of clock circuitry in very large circuits [2]. In several of these works, bio-inspiration plays an important role providing fault-tolerance or self-organising features on top of the asynchronous model [3], [4]. Modeling the behaviour of such NoCs is a key aspect for studying and characterizing the implementations of applications. This has motivated the development of several NoC simulators and frameworks ([5], [6], [7]).

In order to prototype asynchronous NoC architectures with heterogeneous connections, we have developed a multi-FPGA Self-configurable 3-D Cellular Adaptive Platform (SCALP) [4](section II). The main features for simulating SCALP behaviour are: (1) asynchronous parallelism between nodes, heterogeneous physical layer simulation, cycle accuracy, arbitrary topologies, open-source license and easy evolvability. Since none of the existing frameworks provide all these features we have developed SCALPsim[1], an asynchronous simulator for 3-D NoC with heterogeneous links and arbitratry topology intended to model cellular self-organizing architectures.

Section II presents the SCALP hardware platform, section III presents a description of the framework and main characteristics of SCALPsim. Section IV presents the application of SCALPsim to a complex topology of several asynchronous elements from a cellular SOM. At the end, section V discusses about the conclusions and future work.

II. The SCALP platform

A. Hardware

SCALP is a prototyping platform for reconfigurable 3-D NoCs and self-adaptive architectures. Nodes are arranged in a 3-D grid with dedicated links between them. The core of each node is a Xilinx Zynq SoC (System-on-Chip) with a dual-core ARM Cortex-A9 processor and an programmable logic with 74,000 cells. A 2 Gb DDR3 SDRAM is also available for each node. A picture of a SCALP module and a setup with four modules are presented in Fig. 1.

The platform provides dedicated high-speed serial links operating at data rates up to 6.25 Gbps and Low-Voltage Differential Signaling (LVDS) differential pairs connecting the neighbour SoCs in the grid. A total of 48 differential data pairs per board have been included, 8 for each direction (North,

This work is funded by the Swiss National Science Foundation - SNSF and the French National Research Agency - ANR under the grant SOMA.

[1]SCALPsim is available at: https://gitlab.com/scalp_hw/scalpsim

South, East, West, Top, Bottom), together with the associated clock lines. In addition, in the X-Y plane (North, South, East, West), high-speed serial links make it possible to reach higher data throughputs. High-speed serial links are bidirectional, whereas the eight LVDS pairs are unidirectional.

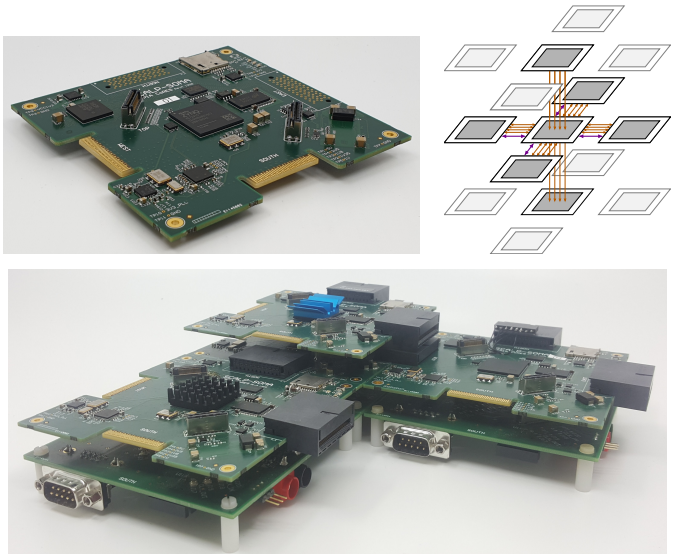

Fig. 1. A SCALP board and an array of 4 SCALP boards. Each SCALP module has 4 bidirectional high-speed serial links (in purple) towards other modules in the same plane and 24 LVDS links in all directions (in orange).

SCALP contains heterogeneous links between computing nodes. They differ in terms of latency and accepted throughput. The configurability of the platform permits to dedicate specific links to certain routing resources in order to use them to deploy multiple computing nodes in each physical SoC.

B. Self-Organising features

The decentralised architecture of SCALP results from a particular interest for self-organising approaches. Cellular Self-Organised Maps (CSOM) [8] are a particular case of SOM where interactions between neurons are local. Weight updates are performed through a cellular propagation among neighbouring nodes only according to the network topology. This topology can be imposed (or at least can be driven) by the physical system topology. A variant of CSOM permits to adapt the topology according to the needs of the task to solve by pruning and sprouting synapses [9].

Unlike SOM, CSOM performs a weight update rule which moves network weights towards neighbouring neurons weights instead of input patterns. Figure 2 illustrates this update: the green node represents a new input pattern, the orange node represent the BMU (best matching unit), which updates its weight towards the new pattern in the two cases, and the blue nodes represent other neurons in the SOM.

In CSOM, for each learning iteration, an input is evaluated and its BMU c is computed. First, the BMU updates its weight vector according to: $\mathbf{m}_c = \mathbf{m}_c + \alpha(\mathbf{x}(t) - \mathbf{m}_c)$ with $\alpha(t)$ being

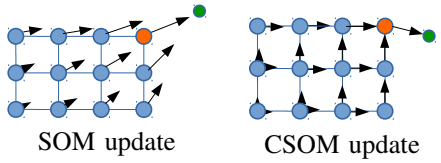

Fig. 2. Weight update rule on SOM and CSOM.

the learning rate. Then every other neuron $i \neq c$ in the map updates its weight vector as follows:

$$\mathbf{m}_i = \mathbf{m}_i + \alpha \frac{1}{\#Infl(i)} \sum_{j \in Infl(i)} (\mathbf{m}_j - \mathbf{m}_i) e^{(-\frac{\sqrt{d}}{\eta(t)} \frac{hops(c,i)}{||\mathbf{m}_i - \mathbf{m}_j||})} \quad (1)$$

where $hops(c, i)$ is the number of propagation hops to reach neuron i from the BMU through synaptic connections, $Infl(i)$ is the set of the influential neurons of neuron i, $\#Infl(i)$ is the number of influential neurons for neuron i, and d is the number of dimensions of the input vector. An influential neuron of neuron i is defined as a neuron closer to the BMU from which neuron i receives a propagating learning signal.

The cellular structure of the algorithm will permit to endow SCALP with Self-Organising capabilities.

III. SCALPsim

Aiming at having different users with different backgrounds, we have developed the simulation platform in Python in order to ease the tool customisation by the user. By using class inheritance, users can modify core functionalities of the tool implementing virtual methods ready for this purpose. A modular software architecture permits users to modify one or more of the following items:

- Routing algorithms.
- Distributed self-organization algorithms.
- High-level topologies or applications.

For high-level users, the main advantage of SCALPsim is to guide the development of the different algorithms or applications with a closer look to the final implementation. This way, hardware constraints such as the use of messages for communication, the absence of a global clock or shared memory, bandwidth limitations, etc. become explicit at an earlier development stage. The tool provides global metrics of the behaviour of the system.

An important requirement is the possibility of modeling a system of a configurable number of nodes operating in parallel and asynchronously. This model emulates the operation of multiple computing resources at slightly different clock rates. These resources can be implemented on a single or multiple configurable devices without a global memory available, therefore communicating by messages exchange.

Having computing and routing resources in either the same or different devices makes it necessary to use an heterogeneous physical layer, where links between computing resources are heterogeneous. SCALPsim permits to configure link properties

for each connection in any arbitrary 3D topology. Four particular features make SCALPsim an unique simulation platform for modelling distributed self-organising architectures:

1) Modular architecture: Building blocks in SCALPsim are generic modules that mimic the logical levels of the simulated system. Communication between different blocks is performed using dedicated FIFO elements, as shown in Fig. 3.

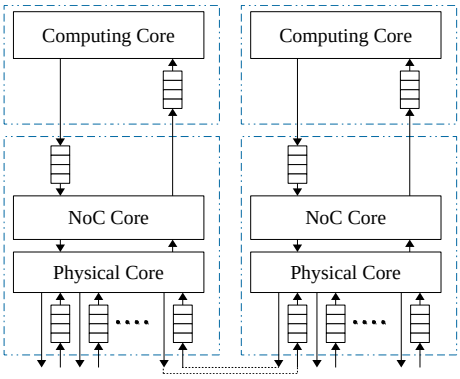

Fig. 3. SCALPsim multi-thread illustration of the execution of two nodes. Each blue box represents a thread executed concurrently.

Tthe *PhysicalCore* contains the ensemble of communication channels with the node's neighbors, and model their specific throuput and latency. The routing capabilities of each element in the network are implemented in the *NoCNode* module. Messages received at the *PhysicalCore* arrive at the *NoCNode* to be managed by the routing algorithms. The *ComputingCore* modules implement the application layer of the system.

2) Asynchronism: The asynchronism of the different computing elements of a multi-FPGA system is a strong constraint for the simulator architecture. These autonomously executing agents are implemented as threads. At the beginning of the simulation, a separate thread is created for each *NoCNode* and *ComputingCore* in the system, as depicted in Fig. 3. The main thread is also kept separate, therefore for a simulation with N nodes, the number of threads is $2N + 1$.

3) Time management: As the execution of the different elements (threads) of the system is asynchronous, a counter implemented in a local variable tracks the local time of each node. This simulated time is continuously growing during execution at a constant pace. Latency and throughput of the physical links are defined by the user in terms of number of cycles. This configuration is used in the *PhysicalNode* to extract messages from input queues. Messages include the time when they are injected into the queues in the receiver side.

4) Message types: Messages are the communication primitive we use to exchange data between nodes. They contain a header and a payload. The former includes information for routing purposes whereas the latter is defined at application-level. The header structure varies according to the type of message. Thus, for unicast operations, the header includes the type of message, destination node and length of payload.

There are two types of multicast operations available: broadcast and reduce. The broadcast operation sends information to all nodes in the network, therefore the header only includes the type of message and length of payload.

For the reduce operation, a message with a reduce request is sent from a node to all other nodes. On the response path (reversed from broadcast), a reduce operator (min, max, sum, prod, and, or) defined in the request is used to locally combine the responses of neighbouring nodes.

IV. CSOM for lossy image compression on SCALPsim

A. SOM and CSOM for lossy image compression

We consider now a well-known application of vector quantization (VQ): lossy image compression [10]. A picture to be compressed is split into smaller $k \times k$ pixels wide thumbnails. We then use these thumbnails as training samples of a VQ model. Once the training is finished, the compressed image is composed of the codebook, and the index of the BMU for each thumbnail extracted from the image. A final lossless entropy coding such as Huffman coding is used to further compress the file. Decompression is performed by reverting the final lossless compression, and recomposing the original image from the stored codebook and the indexes of the BMUs: each index is replaced by the corresponding codeword thumbnail. Figure 4 illustrates this compression/decompression process, without the entropy coding step. See [8] for applications to real images.

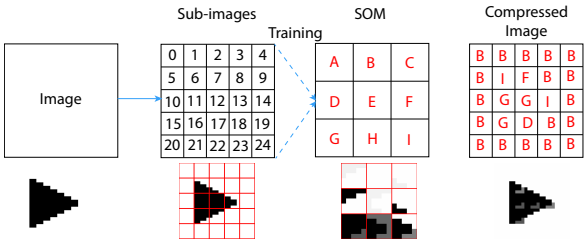

Fig. 4. Simplified scheme of the image compression /decompression process with only 25 sub-images and 9 neurons in a SOM used for VQ.

In the case of a SOM or CSOM used as VQ model, an additional step can be introduced just before entropy coding to further improve its efficiency. A differential coding can be applied to the stored BMU indexes, as in [10]. Each index is replaced by the difference between this index and the BMU index of the immediately neighbouring thumbnail in the direction which maximizes the image smoothness (see [10] for details). This process results in rather small differences thanks to SOM properties, and the final entropy coding performs better on such small values. As studied in [8], CSOM performs better and faster than SOM for highly dimensional sparse data such as considered for lossy image compression. CSOM particularly outperforms SOM for dynamic data.

B. CSOM on SCALPsim

We have simulated lossy image compression using CSOM on SCALPsim with the following method. For each learning epoch, input thumbnails are first shuffled, and for each learning iteration of this epoch (i.e. for each input in the shuffled list):

1) Reduce operation: the input node broadcasts the input to all nodes, each node computes the distance between this input and its own weight vector, the minimal distance (BMU) goes up the broadcast tree.
2) The input node sends an influential unicast message to the BMU with the input thumbnail as the influential vector (see II-B).
3) When a node (firstly the BMU) receives an influential message, it conditionally [2] updates its weights according to equation 1, and broadcasts an influential message to its immediate neighbours, using its updated weights as influential vector, and incrementing the number of hops.
4) When the input node receives a first influential message related to the input thumbnail currently processed, it starts the next learning iteration.

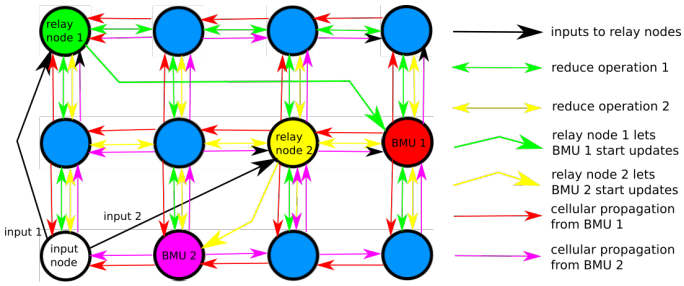

Fig. 5. SCALPsim simulation of CSOM (two inputs in parallel). Colored arrows depict the messages associated to the three first steps of a learning iteration (1. reduce, 2. unicast to BMU, 3. cellular propagation from BMU), these messages being associated to either input 1 or input 2.

In order to more intensely make use of the distributed NoC, it is possible to handle several input thumbnails simultaneously without altering the learning efficiency.

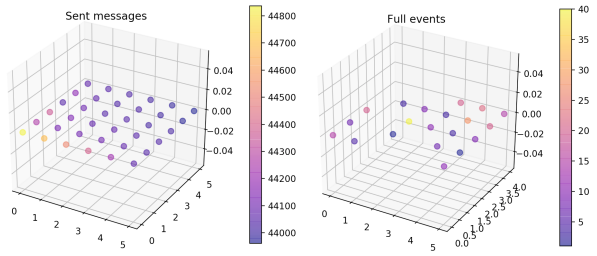

Fig. 6. SCALPsim analysis of a 6×6 CSOM (four inputs in parallel).

To avoid communication bottlenecks in this variant, the input node sends different input vectors to a group of randomly chosen nodes (relay nodes) that perform reduce operations to find the corresponding BMUs from which cellular propagations of influential messages are simultaneously initiated. This is illustrated on figure 5. Of course, if several relay nodes find the same BMU, cellular propagations will be handled sequentially by this common BMU (all messages are tagged by an input identifier, thus there is no possibility of confusion). In this specific case, the NoC usage is not improved.

[2] if this message is the first received for a given input or if this message has been received by a shorter path from the BMU than previous messages.

Analysing the efficiency of such an implementation needs to take into account asynchronism, latency, etc. It is computationally untractable. This is where SCALPsim provides us with useful analysis tools in order to find the best configurations, since the simulation takes into account all physical constraints. Figure 6 illustrates this with an example showing the number of sent messages by NoC nodes and the number of "full events" (a full event corresponds to a detected overflow of an output FIFO, i.e. all outgoing channels being busy when trying to send a message) in the case of a 6×6 CSOM using 4 simultaneous relay nodes with an image split into 16 thumbnails and 100 epochs. Note that the plot uses 3 axes but does not really use the 'z' axis since the 2D CSOM is configured on a 2D SCALP architecture. In the case of the "full events", only some nodes among the 6×6 CSOM appear: the other nodes do not detect any overflow.

V. Conclusions and Future Work

In this paper we presented SCALPsim, a simulation framework permitting to prototype and analyse the implementation of self-organising mechanisms on a distributed complex network of asynchronous elements interconnected with heterogeneous links. We presented a use case of SCALPsim deploying a CSOM algorithm for handling an image compression task by quantizing thumbnails which are learnt by the algorithm. On this deployment, SCALPsim permits to analyse network resource utilisation in order to identify potential bottlenecks.

Future work will be to implement the model on the SCALP platform, in order to validate the computation model.

References

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, pp. 1–es, 2006.

[3] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, and S. Cawley, "A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks," *International Journal of Reconfigurable Computing*, vol. 2009, 2009.

[4] F. Vannel, D. Barrientos, J. Schmidt, C. Abegg, D. Buhlmann, and A. Upegui, "SCALP: Self-configurable 3-d cellular adaptive platform," in *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, November 2018, pp. 1307–1312.

[5] A. B. Achballah and S. B. Saoud, "A survey of network-on-chip tools," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 9, 2013.

[6] M. Saleh Alalaki and M. Opoku Agyeman, "A study of recent contributions on simulation tools for network-on-chip (NoC)," *International Journal of Computer Systems*, vol. 4, no. 3, 2017.

[7] S. Khan, S. Anjum, U. A. Gulzari, and F. S. Torres, "Comparative analysis of network-on-chip simulation tools," *IET Computers & Digital Techniques*, vol. 12, no. 1, pp. 30–38, 2017.

[8] B. Girau and A. Upegui, "Cellular self-organising maps," in *Int. Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM'19)*, 2019.

[9] A. Upegui, B. Girau, N. Rougier, F. Vannel, and B. Miramond, "Pruning self-organizing maps for cellular hardware architectures," in *NASA/ESA Conf. on Adaptive Hardware and Systems (AHS 2018)*, August 2018.

[10] C. Amerijckx, J.-D. Legat, and M. Verleysen, "Image compression using self-organizing maps," *Systems Analysis Modelling Simulation*, vol. 43, no. 11, pp. 1529–1543, 2003.