SCALP: Self-configurable 3-D Cellular Adaptive Platform

Fabien Vannel* Diego Barrientos*, Joachim Schmidt*, Christian Abegg*, Damien Buhlmann* and Andres Upegui*
* *InIT, HEPIA, University of Applied Sciences of Western Switzerland*
Geneva, Switzerland
Email: (fabien.vannel, diego.barrientos, joachim.schmidt, christian.abegg, damien.buhlmann, andres.upegui)@hesge.ch

***Abstract*—Parallel computation has appeared as the most promising technique to circumvent the limitations imposed by power consumption in order to continue increasing computation power, making thus manycore architectures a promising computer organization approach. Interconnecting and coordinating such high amount of computation nodes in an efficient manner is a hot research topic, several approaches to Network-on-chip architectures propose solutions for this. This paper presents a 3D multi-FPGA hardware platform permitting to prototype 3D NoC architectures with dynamic topologies. More precisely, we intend to use it to prototype self-adaptive and self-organizing hardware architectures in which the computation performed by a node and the interconnections between these nodes can be dynamically modified, being these modifications triggered by the platform itself. This paper presents the overall hardware organization and gives some hints about how to use it.**

***Index Terms*—Bio-inspired hardware, cellular computing, Network on Chip (NoC), Dynamic Partial Reconfiguration (DPR), self-organization, System on Chip (SoC)**

I. Introduction

During several years, improvements in computing performance were achieved mainly by process technology. Manufacturing processes allowed to increase the operating frequency of the system in each iteration, leading also to a higher power consumption per computing unit. In order to circumvent the limitations imposed by the high power consumption, the multicore approach [1] proposed the use of parallelism, which has proven to be a valuable strategy in portable systems requiring low-power consumption.

Modern trends are focused on a higher level of parallelization by using manycore architectures [2]. The rise of network capabilities in terms of high bandwidth and low latency has opened a new programming paradigm not only for High-Performance Computing (HPC), but also for systems with fewer computing power requirements. Latest developments in the field have already shown the advantage of heterogeneous computing architectures with CPUs (Central Processing Units), GPUs (Graphics Processing Units) and, lately, with FPGAs (Field-Programmable Gate Arrays) [3].

However, these systems are typically general-purpose architectures with high granularity and a predefined computer organization architecture. The flexibility and adaptation of the system relies on software running on the available computation cores.

A more versatile and flexible solution can be proposed by borrowing inspiration from biological systems. Some of the interesting features to mimic are local computation, preference for local interactions, fully distributed computation (absence of centralized control), core specialization, neural and synaptic plasticity, 3D physical structure or asynchronous execution, among others.

A 3D cellular flexible architecture result thus as a promising evolution for such type of systems. Local direct interconnections between physically adjacent nodes in a multilayer 3D stacked array, coupled with an efficient routing mechanism allowing to interconnect remote nodes, has the potential of satisfying the above mentioned features about local and remote interactions, synaptic plasticity and 3D connectivity. On top of that, each node containing a high-end dynamically reconfigurable FPGA device with its independent memory and clock but with the potential of running applications on hardware and/or software results on a powerful manycore computing system exhibiting all the above described features.

This paper focuses on presenting such platform and gives some ideas about how to exploit these hardware capabilities for building self-organizing and self-adaptive systems. In this regard, the paper is structured as follows: section II introduces existing hardware systems with similar features to the presented platform; section III describes the physical hardware capabilities, mainly concerning SoC and inter-board connectivity; section IV presents the overall architecture including system organization, inter-board routing, FPGA dynamic partial reconfiguration and interaction with input and output devices; section V briefly discuss how to endow the system with self organizing properties; and finally section VI concludes and discuss future perspectives of the presented work.

II. Cellular architectures

Distributed computing refers to a multiplicity of interconnected computing units performing computation in a concurrent manner. Several issues arise when programming such type of platform, including asynchronous execution, single point failures, and the handling of a distributed memory.

Network topology is a fundamental issue and a hot research topic in many fields. Neurosciences, cultural and social net-

The work presented in this paper has been founded by the Swiss National Science Foundation - SNSF

works, epidemiology, and, of course, computer architectures are only a few examples of a large list of research domains where network topology is a key element for dealing with and understanding the overall system dynamics. The topology of the network defines the interactions between different parts of a system. Fully connected networks, permitting to directly link every element to every other, are unrealistic networks from a certain level of system complexity (number of nodes). Real existing networks either found in biology or engineered by humans are defined by other types topologies. Grid, random, small-world and scale-free networks are different approaches to model these real topologies permitting to study, as a bottom-up approach, a global system behaviour following from its low level system internal connectivity.

The principle of locality in cellular computing [4] states that connections are more likely to get connected to "physically" close cells, and remote connections are less likely to be established and should contain lower amounts of information. These constraints suggest a close relationship with grid topologies, while still allowing a certain level of non-uniform randomness in its connectivity pattern.

An example of a physical cellular architecture machine is the BioWall [5], an electronic tissue built as a 2D array of FPGAs interconnected as a grid. It was mostly intended to prototype bio-inspired systems based on cellular automata. An evolution of such platform was Confetti [6], a 2D cellular architecture composed of two layers of FPGAs. The top layer was dynamically reconfigurable and was intended to host computation nodes, while the bottom layer was mainly in charge of interconnecting nodes. This architecture permitted more flexible topologies than the BioWall, thanks to the routing possibilities based on HERMES [7] proposed by the bottom layer.

A more recent work of a cellular multi-FPGA system is Bluehive [8], which intend to run spiking neural models in an efficient way by intensive parallel computation. Node interconnection is a critical issue, therefore they support 3D topology interconnection between FPGAs. However, the 3D interconnectivity is performed by means of a custom 2D motherboard routing every possible interconnection between two nodes: imposing, inevitably, an eventual limitation in terms of system scalability.

One of the most remarkable efforts for building scalable multi-device systems is the SpiNNaker project [9], which goal was to build a neuromorphic architecture able to simulate 1 billion neurons in real-time. The SpiNNaker system is built as a toroidal topology, where every computation node is connected to 6 different nodes forming a triangular lattice. Each device contains a custom chip composed of 18 ARM cores [10]. The main reason for considering a toroidal topology is because of practical issues related to 2D printed boards. However, the 6 connections may allow more complex topologies like the 3D grid topology described in this paper.

It is also worth mentioning multi-FPGA systems like the DN8000K10 from Dini Group [1], BEE7 from BEEcube [2] and Springbok [11] which are mainly intended to prototype very large ASIC designs not fitting in a single FPGA. These systems are not intended to deal with distributed computations, scalability issues, network topology, concurrency, etc. For that reason, we consider that they fall into a different category and we do not intend to compare to them.

The Self-configurable Cellular Adaptive Platform (SCALP), presented in this paper, proposes a step forward on cellular multi-FPGA architectures by proposing a real 3D system topology with direct hardware inter-node connectivity. This feature, compared to previous platforms, provides a more promising solution in terms of system scalability, given the absence of physical constraints when stacking new FPGA boards on any dimension of the system. Of course, some other limitations may appear such as power supply and task synchronization, among others; however, it will be part of a further study.

III. SCALP HARDWARE

With the increasing deployment of SoC solutions provided by major FPGA/CPU manufacturers, we decided to develop a cellular architecture based on them. Such a system has been chosen to provide reconfigurability and interconnectivity of its nodes (cells), as will be described in section IV. The basic building block of the system nodes is depicted in Fig. 1. Each module has a size of 8x8 cm and is implemented in a 10-layer PCB (Printed Circuit Board).

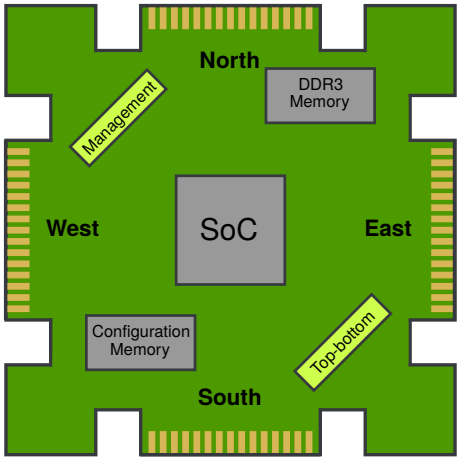


Fig. 1: Schematic representation of a SCALP node.

The main component in the board is a Xilinx Zynq SoC. The selected part includes a dual-core ARM Cortex-A9 processor operating at a frequency up to 866 MHz and an Artix-7 programmable logic with 74,000 cells. In addition, it includes 160 DSP (Digital Signal Processing) blocks and 4 high-speed serial interfaces capable to sustain data rates up to 6.25 Gb/s in each direction (North, East, South, and West). This SoC has been selected due to the presence of 4 embedded multi-gigabit

[1] http://www.dinigroup.com/web/DN8000k10.php
[2] http://www.beecube.com/bee7.html

transceivers while having a moderate cost, which will become relevant when scaling the size of the complete system.

The modules also include an external 2 Gb DDR3 (Double Data Rate Type 3) SDRAM (Synchronous Dynamic Random-Access Memory) device used to store any data required by the self-organizing algorithms or the system application. The SoC provides a dedicated DDR3 controller which can be used directly from the processor or from the programmable logic with a DMA (Direct Memory Access) controller.

A. Node interconnections

Each SCALP node has been designed to interface with its four neighbors in a two dimensional grid: labelled North, South, East and West (see Fig. 1 for details). These links are wired to the four high-speed serial transceivers available in the SoC. Besides the horizontal interconnections, the architecture also foresees serial links to the top and bottom modules, implemented in differential pairs allowing to reach data rates up to 928 Mb/s in both directions. Fig. 2 shows three planes of nodes and their links.

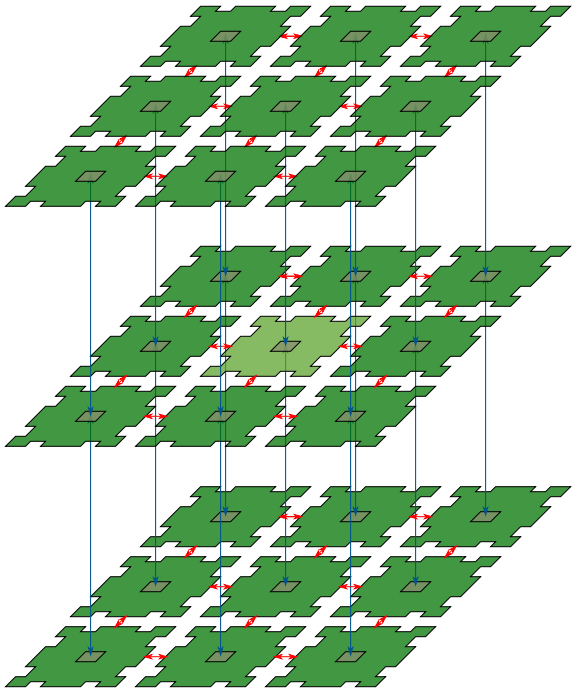


Fig. 2: 3D SCALP nodes interconnection. Red arrows represent bi-directional high-speed links capable to sustain data rates up to 6.25 Gb/s in each direction. Blue arrows are used for bi-directional differential links with data rates up to 928 Mb/s.

Connections in the horizontal plane are accomplished with edge connectors with a custom pin-out. Extra differential pairs routed to the SoC have been added as general purpose input-outputs. For hardware management and top-bottom links, two hermaphrodite connectors are placed at each side of the module, allowing to interface with the top and bottom nodes. The pin-out of these connectors has also been customized for the project, as described in the following section.

B. Hardware management and supervision

In order to keep system nodes as simple as possible and avoid dedicating large areas for management connectors, an extra board has been foreseen for that purpose. The management board has the same form-factor as the node boards and includes the following connectors: two mini-USB (one for USB On-The-Go and one for UART), one RJ45 (Ethernet), one JTAG and one for power supply. Fig. 3 shows an arrangement of a node board and a management board.

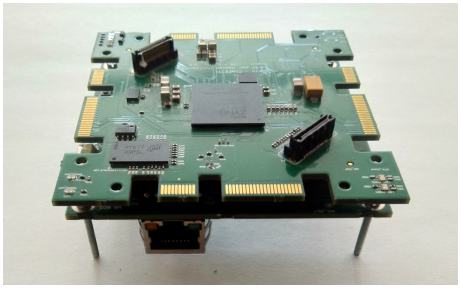


Fig. 3: Picture of SCALP node on top of a management board.

The JTAG interface is directly routed to one of the connectors to the upper node in the management board. Then, each node connects the JTAG bus to its SoC and to the node on top, if present. Otherwise, it closes the bus towards the lower node. This mechanism has been conceived to access any SoC in a stack (from bottom to top nodes) using a single JTAG connector. Each node includes a 256 Mb SPI Flash memory being able to store up to 9 complete bitstreams. This memory is accessed through the SoC and intended to be used at boot time.

The management board includes a RJ45 connector and Gigabit Ethernet transceiver, which is routed to the hardware management connector towards the upper module. In the nodes, an Ethernet switch links the upper and lower interfaces as well as the SoC of the node itself. Similarly to the JTAG chain, it allows to use a single physical connector to access all SoCs in a stack. The Ethernet interface of each SoC is a general purpose management and supervision gateway to the internal state of the SoC.

Another two diagnostic interfaces included in the design are the UART and USB On-The-Go (OTG). Both use separate mini-USB connectors and transceivers in the management board. In the nodes, the UART bus is linked to the debug interface of the CPU and the OTG to dedicated pins. Both are also linked to the upper and lower nodes.

The power supply chain starts with a 5 V power jack in the management board. Linear regulators either in the management board or in the nodes are used to generate the different supply levels needed for all circuitry. The system has been designed to allow up to 5 nodes, placed in a stack, powered from a single management board.

IV. SCALP Architecture

The hardware platform previously described has been conceived to develop and study different characteristics of bio-

inspired systems. In particular, we intend to investigate self-organizing techniques based on cortical plasticity. This attribute refers to the development of the brain according to the arrangement of its neurons (structural plasticity) and its connections (synaptic plasticity). By analogy, dynamic reconfiguration of computing nodes and flexible routing of Network on Chip (NoC) will provide the means to *hardware plasticity*.

The SCALP platform provides independent nodes (neurons) to be interconnected in a decentralized cellular architecture. The topology of the system and the processing element(s) of each node can be dynamically adapted to the application in order to optimize the global performance. Thus, upon a change in the system inputs, the arrangement and physical connections of the nodes will adapt consequently.

In order to achieve hardware plasticity, we have based our design in SoCs, external memories and high-speed node-to-node interconnectivity, as presented in section III. These choices make possible the implementation of dynamic reconfiguration techniques and NoCs. In addition, they provide other features, such as hardware/software co-design, which may facilitate the development of the system and/or the deployment of specific applications. Some of the building blocks of SCALP, represented in Fig. 4, can be implemented in Programmable Logic or in the Processing System. A detailed description of the system properties, usage of hardware resources and different techniques available in the SCALP platform to achieve hardware plasticity are included in the following sections.

A. Dynamic reconfiguration

The possibility to dynamically change the computing core of each node in the system permits to emulate the structural plasticity of the brain. In previous works, we have explored different ways to reconfigure the nodes. One option is to load the Flash memory of each node with different bitstreams implementing diverse computing cores [6]. Each SCALP node has a flash memory which can store up to 9 complete bitstreams. Therefore, the node can decide to swap dynamically the FPGA contents with a different configuration available in the flash memory of the node itself.

Another option is to constrain the computing core in a hardware area with a well-defined interface which could be replaced dynamically. FPGA manufacturers provide a low-level interface to reconfigure these areas. In particular, the SoC present in our cellular architecture is suited to perform Dynamic Partial Reconfiguration (DPR) [12]. A dual-core ARM processor can be used to load partial bitstreams stored in the DDR memory of the nodes.

Dynamic reconfiguration can be performed at different levels. In the previous examples, complete functional blocks have been considered. However, computing cores can be reconfigured at bit/gate level by changing the contents of look-up-tables (LUT) or multiplexers [13]. The reconfiguration of low-level blocks could allow to generate new computing cores locally at each node and avoid using a predefined set of global/partial bitstreams.

B. Network on Chip

Extrapolating the concept of NoC [14] to interconnect the nodes of our system, we aim at more efficient routing algorithms. The presence of high-speed serial links to interconnected SoCs, allow the use of NoC principles and algorithms (such as [7]). This concept take advantage of the topology of the system to achieve better performance than traditional bus-like solutions.

The three dimensional SCALP architecture opens the door to prototyping 3D NoCs [15]. The adding of the vertical links in our architecture allows a better congestion control in the network as well as lower latencies for communications. As the performance of horizontal and vertical links is different by design, the introduction of a Quality of Service (QoS) metric yields to more efficient routing. This metric can also be used to categorize the priority of the routed messages, leading the introduction of multiple virtual 3D NoCs, as shown in Fig. 4.

Different layers of the system require dedicated dynamic topologies for node-to-node connections. The physical layer is shared between multiple NoCs, which act as dynamic routing elements with different virtual topologies. One topology/NoC may be used to find the most appropriate node to process the input data, for example. This NoC may have a higher priority for its messages, as another NoC managing the load balance of the system may need this information to be exchanged rapidly.

In addition, processing cores (dynamically reconfigurable as discussed in the previous section) may need to communicate with each other to perform complex computations in a parallel but synchronous manner. Input data or inter-node data in the application layer may be exchanged between different nodes as well. Each of these communications require a different virtual topology, which will be dynamically adapted according to physical location of the nodes, availability, etc. Virtual NoCs will provide a hardware abstraction to the interconnectivity of the nodes which emulates the property of synaptic plasticity.

C. System Inputs-Outputs

Either the high-speed links or the standard differential lines used to connect system nodes (see Fig. 1 for details) can be used as system Inputs or Outputs (I/Os). The granularity of the system allows having multiple distributed I/Os which can be managed by any node or group of nodes due to the intrinsic decentralized architecture.

Examples of I/Os may range from RGB LED matrices (as in [6]), touch pixels or screens to high resolution cameras or motor controllers depending on the application. Depending on the specific I/O requirements, dedicated hardware may be developed to provide a suitable physical connection, adapt the link protocol or data pre/post-processing, if necessary. One or several of these hardware adapters can be used to connect I/Os to the different nodes of the system, allowing to demonstrate the plasticity of the complete system.

D. Hardware/software co-design

The SoC available in each node can be used to hardware/software co-design in any of the system elements: NoCs,

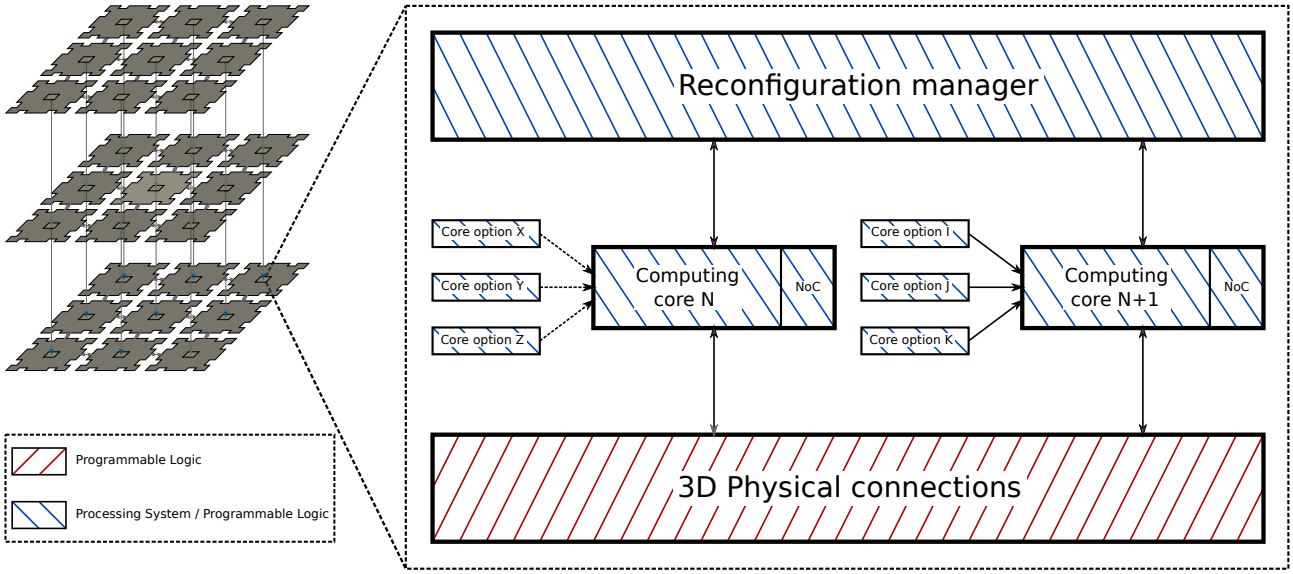


Fig. 4: SCALP building blocks. The 3D physical topology is enhanced with reconfigurable capabilities (structural plasticity) and the use of several virtual topologies (synaptic plasticity).

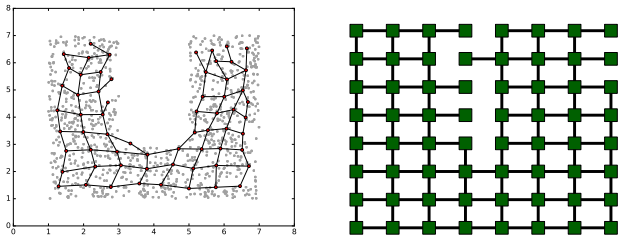
processing core, I/O processing, etc. The smart use of the SoC resources can give better performance numbers when adapting to specific applications than architectures with fixed pre-configured resources. Additionally, it provides a more flexible interface for a quicker deployment of new applications.

V. Self-Organizing Hardware

One of the fundamental questions to be addressed when it comes to talk about self-adapting or self-organizing hardware is how to decide when to modify the hardware and what to modify on it. In other words, how to define the conditions to determine the configuration to be deployed on a node by taking the decision in a completely autonomous and decentralized manner, while exhibiting a coherent behavior at global system level.

Self-Organizing Maps (SOM) [16] suggest some answers to this question. Input data, represented as vectors (bitmaps, arrays of sensors, sequences of temporal data, etc.), are presented as inputs to the multi-core system. Each core can be associated to a prototype vector that will represent an ensemble of input vectors thanks to the vector quantization properties offered by the SOM. The quantization results in a density representation of the different vectors presented as inputs to the system. This principle can be applied to a multicore system assuming that a computing core is represented by a prototype vector and, due to the quantization, the architecture will be able to shape and adapt its functionality to the nature of the input data.

Some work in this direction has already been done in [17], where a SOM algorithm implemented on an FPGA architecture is used for load balancing of a multi-task application based on input video streaming. In this work a core processor can decide which configuration to adopt according to the represented prototype vector.



(a) Input 2D vectors (grey dots) and Prototype 2D vectors (graph)

(b) Cellular NoC architecture

Fig. 5: Example of resulting learned topology using PCSOM.

A step further is proposed in [18]. The Pruning Cellular Self-Organizing Maps (PCSOM) is a SOM algorithm specifically conceived for fitting in the SCALP platform, by taking into account its constraints and exploiting its flexibility. PCSOM is specific to cellular architectures by avoiding global values used to adapt neurons weights (as in SOM), but limiting the influence of a neuron only to its neighbors (i.e. the ones connected to it). Moreover, it yields to prune useless connections between neurons preventing the network to converge to optimal vector quantization. The pruning of the network will thus permit to drive the connectivity of the NoC. In [18], it is described an initial topology in the form of a 2D connected grid, followed by a learned NoC connectivity driven by the distribution of input data. Fig. 5a illustrates an example in the form of input vectors and vector prototypes mapped into a 2D Cellular NoC (Fig. 5b) which has been re-routed after pruning useless connections.

Future work on Self-organizing cellular systems will focus on exploring how to handle the creation of new connections. In biological neural networks, sprouting of new synapses is one of the main mechanisms of neural plasticity allowing to

shape the brain to an optimal topology. In a similar fashion, sprouting and pruning cellular networks will permit a many-core computer topology to converge to optimal architectures, and should permit to adapt such architecture in a dynamic way in the presence of changing environments during the machine lifetime.

VI. Conclusions and future work

In this paper we have presented the SCALP platform permitting to prototype self-adapting cellular systems in the form of dynamic Networks on Chip. An enhanced platform reconfigurability enabled by dynamic partial reconfiguration coupled with a dynamic routing scheme supporting physical 3D architectures endow the SCALP platform with an unprecedented architectural flexibility.

High-performance distributed applications may also benefit from this architecture thanks to its capability for supporting heterogeneous computing machines composed of a multiplicity of FPGA custom nodes plus ARM Cortex-A9 dual-cores. High-speed serial links along with a lightweight custom communication protocol will ensure that inter-node communication will not reduce system performance by a considerable amount.

Future work will be focused on building and validating a large amount of computing nodes and on proposing and deploying the distributed algorithms that will permit to exhibit self-organizing and self-adapting properties present in mammals' brains based on neural plasticity mechanisms such as sprouting (synapse creation) and pruning (synapse removal).

References

[1] P. Gepner and M. F. Kowalik, "Multi-core processors: New way to achieve high system performance," in *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, Sept 2006, pp. 9–13.

[2] J. Diaz, C. Muñoz-Caro, and A. Niño, "A survey of parallel programming models and tools in the multi and many-core era," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1369–1386, Aug 2012.

[8] S. W. Moore, P. J. Fox, S. J. Marsh, A. T. Markettos, and A. Mujumdar, "Bluehive-a field-programable custom computing machine for extreme-scale real-time neural network simulation," in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. IEEE, 2012, pp. 133–140.

[3] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Y. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, "A cloud-scale acceleration architecture," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–13.

[4] M. Sipper, "The emergence of cellular computing," *Computer*, vol. 32, no. 7, pp. 18–26, Jul 1999.

[5] G. Tempesti, D. Mange, A. Stauffer, and C. Teuscher, "The biowall: An electronic tissue for prototyping bio-inspired systems," in *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*. IEEE, 2002, pp. 221–230.

[6] P. A. Mudry, F. Vannel, G. Tempesti, and D. Mange, "Confetti : A reconfigurable hardware platform for prototyping cellular architectures," in *2007 IEEE International Parallel and Distributed Processing Symposium*, March 2007, pp. 1–8.

[7] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69 – 93, 2004.

[9] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[10] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[11] S. Hauck, G. Borriello, and C. Ebeling, "Springbok: A rapid-prototyping system for board-level designs," in *ACM/SIGDA 2nd International Workshop on Field-Programmable Gate Arrays*, 1994.

[12] M. A. Kadi, P. Rudolph, D. Gohringer, and M. Hubner, "Dynamic and partial reconfiguration of zynq 7000 under linux," in *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec 2013, pp. 1–5.

[13] A. Upegui and E. Sanchez, "Evolving hardware with self-reconfigurable connectivity in xilinx fpgas," in *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, June 2006, pp. 153–162.

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

[15] V. F. Pavlidis and E. G. Friedman, "3-d topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081–1090, Oct 2007.

[16] T. Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, 1989.

[17] L. Rodriguez, L. Fiack, and B. Miramond, "A neural model for hardware plasticity in artificial vision systems," in *Design and Architectures for Signal and Image Processing (DASIP), 2013 Conference on*. IEEE, 2013, pp. 30–37.

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