Abstract—The design of a resource allocation mechanism is becoming a major challenge with many-core SoC architectures. We define a hardware controller in which a grid of processing elements (PEs) will support a set of neuro-cognitive processes in order to drive a robot in different tasks. We propose an original artificial neural network (NN) named DMAD-SOM for "Distributed Multiplicative Activity Dependent Self Organizing Map" inspired by Neural Fields (NF) equations that have shown self-organizing behaviors and can be suitable for this purpose. This model is shortly described here. It can be used to take allocation decisions locally, taking in account the state of the whole system through the emergent behavior of the network. This paper describes the distributed DMAD-SOM model and focuses mainly on its implementation onto FPGA.

I. INTRODUCTION AND STATE OF THE ART

This work takes place in the French ANR SATURN project [1]. The aim of this project is to design the hardware architecture of a self-adaptive and intelligent controller in the context of mobile robotics. This controller is composed of four main layers described in figure 1.

![Layered view of the system architecture](image)

Fig. 1. The layered view of the system architecture: the proposed architecture adapts itself to its environment in a data driven way. The first layer leads the primitive signals to the preprocessing layer. This second layer processes compute saliency and prepares the needed informations to the third layer. The adaptation layer then learns the data structure and leads the replication mechanism of the fourth layer. The fourth layer is the computing layer composed by an array of processing elements.

This paper describes the definition and the implementation of the adaptation layer which is mainly an intelligent distributed resource allocation mechanism. The design of an efficient resource allocation mechanism for highly parallel SoC architectures remains an important research topic. A centralized solution lets the system be aware of each processing element to take global decisions, but is also an obstacle for the robustness, the scalability and the efficiency of the whole system. NNs have been shown to have a good behavior applied to the scheduling problem and can easily be dispatched in the architecture in order to take local decisions.

II. DMAD-SOM MODEL DESCRIPTION

The first explored solution was based on Kohonen SOM (KSOM) and required a centralized election of a winner neuron. A second solution aims to distribute this election on the whole system [2]. The functional restrictions of this solution lead us to investigate others NN models as NF [3]. This new model can be situated somewhere between the KSOM and the NF and is able to take local clustering decisions in a complete distributed way. It’s behavior emerges from local decisions taken relatively to the activities of the neurons and their neighbors. It is then named "Distributed Multiplicative Activity Dependent Self Organizing Map" (DMAD-SOM). It is composed of two layers. The first one drives the outputs of the preprocessing layer, normalized in [0, 1], up to the second layer. This second layer is composed of the learning neurons which are connected together thanks to lateral connections (their four nearest neighbors). The activity \( U(t) \) of each neuron follows a classical activation function relatively to its action potential \( P(t) \), computed thanks to two terms (eq. 1).

\[
P(t) = X(t) + Y(t) \tag{1}
\]

Where \( X(t) \) represents the activation due to the input layer synapses (eq,2) and \( Y(t) \) the lateral excitation from the nearest neurons (eq,3).

\[
\Delta X(t) = -X(t) + \prod_i e^{-(w_i(t) - x_i(t))^2} \tag{2}
\]

\[
\Delta Y(t) = -Y(t) + \sum_j (W_j(t)U_j(t)) \tag{3}
\]

\[
\Delta w_i(t) = l \times U(t)(x_i - w_i) \tag{4}
\]

The learning rule is applied only on the input synapses, with \( l \) a learning rate, \( U(t) \) the current activity of the neuron.

III. HARDWARE IMPLEMENTATION

Building a distributed hardware self-organizing map is not without difficulties. In this section we will expose some elegant solutions to this different constraints. Then we will explain more in details the proposed implementation.
A. Challenges and constraints

The first constraint in hardware architectures is the communications bottleneck. A link between a neuron and all the others will dramatically limit the number of neurons. As explained in the introduction, the DMAD-SOM model is completely distributed and gives satisfying results considering a lateral connection with only its four nearest neighbors.

The second constraint in hardware neural networks is the use of non-linear functions. Like in many cases, this functions will be pre-processed and stored in look-up tables (LUTs).

Finally, the hardware consumption can be prohibitive for a full parallel implementation. As the samples come at the sensor rate, it is reasonable to suppose that the working frequency of our architecture will be greater than the one of the samples. According to this hypothesis, the inputs have been serialized to reuse a part of the hardware resources.

B. The proposed architecture

Assuming this last hypothesis, the neurons have one input signal on which the values of the stimulus vector are inputted serially. They also have four input signals from the neighbors which come in parallel, but for a matter of space consumption each neighbor activity is read sequentially. Finally a neuron has one output, the activity, which is valid when all the inputs are processed. The neuron, depicted in fig. 2 is composed of three branches, the input, the lateral and the output branch.

The input branch computes the formula described in eq. 2. The weights are stored in a shift register with a loopback responsible for the learning. Finally, the product is computed by an iterative multiplier.

The lateral branch computes the formula described in eq. 3. The synchronization of the weights and the lateral input selection is done by a simple 2-bit counter. The \( y(t) \) term is then computed by a classical multiplier-accumulator.

Finally the output branch is composed of an adder, an activation block and a register. The activation block is a simple saturation. The activity is sent back to the learning block and to the neighbors.

Thanks to the DMAD-SOM emergent behavior, the weights of the neurons can be randomly initialized. In order to compare the hardware results with the golden version, the network’s internal values (as weights and learning rate) can be initialized from the outside.

In order not to increase the number of signals in the whole architecture, a synchronization path is created thanks to a set of multiplexers. In this new datapath, the data input from the previous neuron (from one of the lateral inputs), pass through the registers to initialize, and output to the next neuron to the activity port. The lateral input to use depends on the position of the neuron in the architecture and is decided off-line.

The architecture then becomes a long shift register during the initialization where the data flow from neuron to neuron. This synchronization path can be used for reading back and making a snapshot that can be reloaded later.

IV. RESULTS

We explore the impact of some parameters of the DMAD-SOM described earlier on the hardware resources consumption. This exploration was done on a new Stratix V 5SGXEA7N2F45C2 which is composed of 469k ALUTs and 938k registers.

![Fig. 2. Internal view of a neuron](image)

We vary the number of neurons for two sets of parameters and observed the resources utilization. One can see in fig. 3 that in both cases, the ALUTs and registers increase linearly, which is due to the regularity of the architecture.

For the first set of parameters (8-bit bus width, 16-bit internal bus and four inputs), up to 1000 neurons can be implemented. For the second set (16-bit bus width, 16-bit internal bus, and still four inputs), a maximum of 300 neurons can be implemented.

A future study is planned to measure the error of the different hardware implementations regarding to the software implementation.

REFERENCES

