A hardware configurable self-organizing map for real-time color quantization

Mehdi Abadi, Slavisa Jovanovic, Khaled Ben Khalifa, Serge Weber, Mohammed Hedi Bedoui

A Hardware Configurable Self-Organizing Map for Real-Time Color Quantization

Mehdi Abadi*†, Slavisa Jovanovic*, Khaled Ben Khalifa†, Serge Weber*, and Mohammed Hédi Bedoui†
*UMR 7198, Institut Jean Lamour, Université de Lorraine, Nancy, France
Email: firstName.name@univ-lorraine.fr
†LR12ES06, Laboratoire de Technologie et Imagerie Médicale, Université de Monastir, Monastir, Tunisia
Email: khaled.benkhalifa@issatso.rnu.tn

***Abstract*—Real-time color quantization requires high performances and high configurability. Self-organizing maps (SOMs) are very suited as color quantizers. However, widely used software SOM quantizers are flexible with limited performances, whereas the hardware counterparts lack in flexibility. In this work, we propose a flexible and adaptable real-time hardware implementation of a SOM map applied to color quantization. The proposed architecture allows to find iteratively, for each processed image, the adequate SOM structure providing optimal quality and performances. It is validated on a 16×16 map using 128×128 RGB images. The proposed reconfiguration allows to improve dynamically the PSNR of processed images by changing the structure of the SOM and its time is estimated to 38 clock cycles.**

I. Introduction

Since its introduction by Kohonen 1982, Self-Organizing Maps (SOMs) drew a considerable attention and were a subject of many research work [1]. SOM is a competitive neural network with unsupervised learning. Its ability to interpret high-dimensional input data made it inevitable for classification and visualization of even noisy or incomplete data. The SOMs are used in many applications such as signal processing, supervision, noise reduction and image processing. Depending on the application needs, several implementation approaches have been proposed for the SOM: software, hardware or mixed. Each application has its own specificities thus requiring different SOM parameters: input layer size (the size of input vectors), output layer size (the number of neurons in the SOM map), timing constrains, memory requirements, etc. Therefore, the reuse of a SOM map tailored to one application in different working conditions of another one is difficult, sometimes impossible without considerable design efforts. The software SOMs bring more flexibility and are customizable, but suffer from limited performances which constrain their use to the soft real-time applications, where the timing constraints are easily achievable. For the tight real-time constraints, the hardware SOM implementations are inevitable.

The high performances brought by hardware SOMs are often at the detriment of flexibility. However, some hardware SOM implementations require not only the real-time performances but also more flexibility during their operation. For instance, real-time color coding requires high throughput capabilities and dynamic adaptability to different image formats and quantization parameters. The hardware SOM implementations proposed in the literature are often configured in the design phase and cannot be adapted during the runtime. In this work, we propose a flexible and adaptable real-time hardware implementation of a SOM map applied to color quantization.

This paper is organized as follows: the state-of-the-art works and the theoretical background are presented and discussed respectively in Sections II and III. Section IV presents the proposed architecture and its application to image quantization. Section V shows the obtained results whereas some conclusions and perspectives are drawn in section VI.

II. Related work

In [5], L. Chen et *al.* proposed a software implementation of improved SOM algorithm named MFD-SOM. To improve color feature extraction, the authors introduced dynamic adjusting of learning rate and neighbourhood size in function of input data, allowing non-principal components to be more competitive and present in the color codebook at the end of the learning phase. The timing performances of the presented approach were evaluated outside the real-time processing framework. In [4], T. Kuremoto et *al.* presented a flexible and adaptable software implementation of SOM algorithm combining parameterless and structure growing features. The proposed association allows additional learning, optimal neighborhood preservation and automatic tuning of parameters with SOM structure growing during the learning phase. Thus, the increased success rate and reduced training and processing times for voice instruction learning system are obtained. To improve timing performances of software implementation of SOMs, A. De et *al.* in [6] proposed a SOM algorithm adaptation for multi-GPU platforms. The presented implementation speeds up the timing performances compared to the existing software implementations of SOMs, but remains still inadequate for hard real-time applications having tighter timing constraints.

However, for real-time applications, hardware solutions based on the use of Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs) are preferred. In [7], Hikawa and Maeda proposed an FPGA-based hardware implementation of a massively parallel 16×16 SOM map with improved neighborhood function implementation. The proposed architecture provides improved quantization performances and real time performances. On the other hand, due to the massively point-to-point communication links, the proposed SOM implementation is application specific and with low or no flexibility because none of the SOM parameters can be changed dynamically during the normal SOM operation. In

[8], J. Lachmair et *al.* presented a scalable FPGA-based hardware implementation of SOM. The proposed architecture is reconfigurable and adaptable to different applications, whereas all modifications must be done offline, during the design phase. It was validated and implemented on a multi-FPGA support, on the example of a large SOM composed of 6050 neurons using mixed communication approach (point-to-point and bus). The obtained performances are among the best state-of-the-arts performances in terms of MCUPS (million connection updates per second) and MCPS (million connection per second) and may guarantee real-time processing constraints.

In the last decade, the communication approach that revealed as the most suitable for large System-On-Chips (SoCs), by resolving the main drawbacks of the traditional shared bus, is the NoC communication approach [3]. The same trend is also followed in large neural network implementations. For instance in [10], to improve the flexibility and modularity, Carrillo et *al.* proposed a hardware implementation of spiking neural network using NoC for communication purposes. Moreover, in [11], M. Abadi et *al.* proposed a scalable and flexible NoC-based hardware implementation of SOM. By separating computing from communication, the rigid SOM structure gained more flexibility and the possibility to reuse and adapt dynamically the same support to a large variety of applications. In this work, the SOM-NoC architecture was improved by allowing online parameter changes and adapted for real-time color quantization.

III. Background

A. Self-Organizing Map (SOM)

A *SOM* is usually described with a two-dimensional distribution of $L \times K$ neurons, where each neuron has its own weight vector $\overrightarrow{m}$ of dimension D, D being the dimension of the input vector $\overrightarrow{X}$. The SOM operation requires two phases: learning and recall. During the learning phase, the SOM generates its output cluster by changing the weights of its neurons. In the recall phase, this output cluster is used for decision purposes.

The learning and recall phases start in the same manner, by presenting an input vector $\overrightarrow{X}$ to the SOM's input. Then, each neuron calculates the distance between its weights $\overrightarrow{m}_{l,k}$ and this input vector $\overrightarrow{X}$. In general, the calculated distance is the Euclidean one as presented:

$$\left\| \overrightarrow{X} - \overrightarrow{m} \right\| = \sqrt{\sum_{\mathtt{k}=0}^{\mathtt{D}} (\xi_k - \mu_k)^2} \qquad (1)$$

Therefore, the winner neuron, which is the neuron having the weights $\overrightarrow{m}_{l,k}$ closest to the input vector $\overrightarrow{X}$, is identified:

$$c = \underset{l,k}{argmin} \left\| \overrightarrow{X} - \overrightarrow{m}_{l,k} \right\| \qquad (2)$$

where c is the index of the winner neuron.

During the learning phase, after identifying the winning neuron for a given input vector, the winner's weights and those of its neighbors are updated according to the used neighboring function $h_{c,l,k}(t)$, as follows:

$$\overrightarrow{m}(t) = \overrightarrow{m}(t-1) + h_{c,l,k}(t) \left[\overrightarrow{X}(t) - \overrightarrow{m}(t-1) \right] \qquad (3)$$

where the neighboring function $\boldsymbol{h_{c,l,k}(t)}$ adjusts the degree of learning for each neuron, depending on its position (l, k) with respect to the winner's one c and the number of the learning iterations called epoch number. This function is very complex for hardware implementation and is therefore often simplified with a hardware-friendly shift function [7].

During the recall phase, each input vector presented to the SOM's input finds its winning neuron having weights closest to it. Hence, different input vectors can select the same neuron. Therefore, the recall phase is a phase of clustering where the neurons' weights are used as codewords. In color processing applications, the weight vectors represent the codebook of the quantized colors.

B. Network on Chip (NoC)

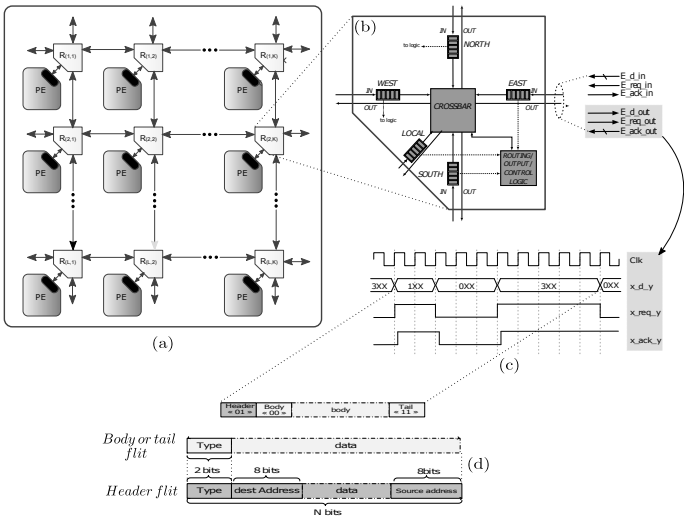

Figure 1. (a) Structure of a 2D Mesh NoC. (b) NoC router architecture. (c) Alternate Bit Protocol. (d) Message structure

Network on a chip (*NoC*) is presented as an alternative to shared bus-based communication allowing the connection of several processing modules on a single chip [2]. This approach enjoys an explicit parallelism, high bandwidth and a high degree of modularity, which makes it very suitable for distributed architectures [3]. Considering the advantages of NoC, it turns out that this is the most suited approach to Multi-Processor System on Chip (MPSoC) having high communication needs.

The structure of a 2D mesh NoC is shown in Figure 1(a). It is composed of processing elements (PEs) and routers. Each router is associated to a PE via a network interface (NI) whose primary function is to pack (before sending) and unpack (after receiving) data exchanged between PEs. Figure 1(d) illustrates the structure of packets circulating in the network using a wormhole switching technique. Each packet is composed of flits: header - opens the communication and "shows" the route to other flits; body flit - contains data to exchange and is often call payload; and tail flit - closes the established connection and frees the leaving router.

The synoptic scheme of a NoC router is presented in Figure 1(b). It is composed of a crossbar, input/output buffers and control/routing logic. The crossbar establishes multiple

links between inputs and outputs of the router according to the predefined routing algorithm and scheduling policy. The crossbar and the arbitration of packets in the router are handled with the controller, which manages also collisions between multiple packets asking the same output link. The input and/or output buffers temporarily accept flits before their transmission to either local PE or to one of the neighboring routers. Figure 1(b) shows also the interconnection signals of an isolated router: a bidirectional data bus, a transfer request and acknowledgment signals denoted with the infixes *_d_*, *_req_* and *_ack_* respectively. Moreover, the packet exchange between a router and its neighbors following the alternate bit protocol is illustrated in Figure 1(c). Sending a flit is accompanied with the request signal on the same port. Upon the reception of the acknowledge signal, the sending of the next flit can be proceeded. A flit is sent or received within a clock cycle. The type of flit is encoded with two most significant bits as presented figure 1(d).

IV. Proposed architecture

The proposed architecture of SOM color quantizer is shown in Figure 2. It consists of two layers: the communication layer, which represents a wormhole switching 2D NoC using the well known XY routing algorithm; and the processing layer, which regroups all processing elements (PEs), where each PE corresponds to one neuron of the SOM architecture. The synchronization between these two layers is ensured in a classical manner by a network interface (NI) located on each PE. The proposed architecture uses the Multiple-Instruction Multiple Data (MIMD) technique to achieve parallelism and does not posses global processing modules. Each neuron operates independently to process a portion of the overall SOM algorithm. The architecture of a neuron is shown in Figure 2. Four modules can be seen: 1) Vector Processing Element (VPE): the processing circuit whose main role is to calculate the Euclidean distance between the parent neuron's weights and the input vector and, to update the weights of the parent neuron. The VPE calculates the distance sequentially: for an N-element input vector, the distance calculation takes N+1 clock cycles. The size of input vectors to process by the VPE is configurable and is configured during the configuration process. 2) Local Winner Search (LWS): the comparator circuit performing the comparison between the local distance, calculated by the VPE module, and the distances sent through the NoC from its neighboring neurons. The LWS's operation is done within a clock cycle when all values to compare are received and put on the LWS's inputs. 3) Update Signal Generator (USG) : the circuit which launches the update process of the parent neuron. Depending on the position of its parent neuron with regards to the winner, it supplies the VPE module with the shift parameter needed to compute the neighboring function. 4) Local Configuration (LC): the circuit in charge to configure the parent neuron upon reception of the configuration data. The synchronization between different units of the architecture is ensured with the messages routed via the NoC. Therefore, a neuron receives via the NoC different types of messages: input data to process, processed data from other neurons of the SOM and configuration data.

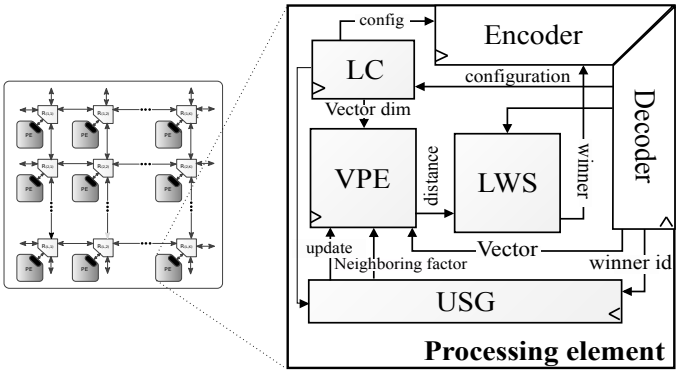

Figure 2. Architecture of the NoC-based SOM (left) and the synoptic scheme of a neuron with the corresponding network interface (right)

The processed data from all neurons' VPE modules are exchanged between neurons in a systolic manner starting from the top-left neuron (level 1) until the bottom-right one (level $L + K - 1$ for a $L \times K$ map). Two adjacent systolic levels have the Manhattan distance equal to 1 and data exchange between two neurons belonging to them involves the crossing of two NoC routers. A neuron belonging to the level i receives the comparison results of the adjacent neighbors belonging to the level $i - 1$, searches the local minimum distance with the LWS module and sends the obtained result (the new local minimal distance) to its adjacent neighbors belonging to the level $i + 1$. Thus, the minimum distance is obtained after the time needed to compare all calculated distances through $L + K - 1$ systolic levels. At the end of the competition phase for a given input vector, the ID of the winner neuron associated with the minimum distance will be identified in the bottom-right neuron located on the level $L+K-1$. This value will be broadcast during the adaptation phase, in the reverse direction, starting from the bottom-right neuron until the top-left one. In the proposed architecture, the broadcasting of the winner ID and the minimal distance is ensured by the neurons positioned at the rightmost column of the SOM.

The configuration data which allow to change dynamically the parameters of the SOM are received from the user (via an additional PE, not presented in Figure 2) through one of the available NoC links. The (re-)configuration of the SOM can be started at any time by user, but usually is launched at the end of the current data block processing (i.e. whole image). After reception of configuration data, each neuron via the LC module decodes the received configuration and sends the corresponding configuration signals to all modules of the neuron to accept this new configuration. During the (re-)configuration, if precised in the received configuration, the neuron configures the size of the input vector, the addresses of the neighboring neurons to which processed data will be sent and the overall size of the map. If the configuration data contain new SOM size, the LC identifies its parent neuron's position in this new structure, its direct neighbors and the new parameters to use during the update process. The size of input vectors to process by the VPE can also be configured during this phase; thus, between two configurations, the VPE processes all input data as vectors of known size, the one written during the last configuration. In the proposed approach, the size of the SOM is also configurable. This is achieved by fixing the maximal allowable size in function of the hardware platform used for implementation. Therefore, if a lower number of neurons is needed for a given application, these extra neurons will be

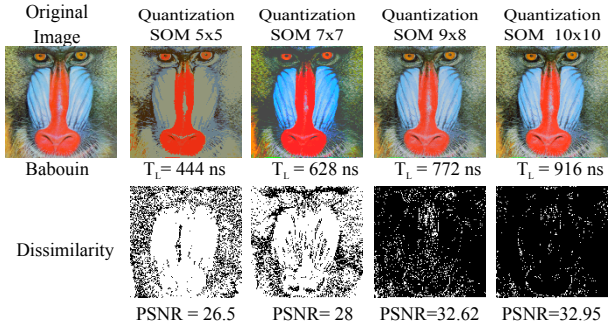

Figure 3. A snapshot of obtained results with the proposed architecture used as color quantizer for 4 different configuration: 5×5, 7×7, 9×8 and 10×10; T_L is the time needed to process one pixel on a given configuration

deactivated during the configuration process. Consequently, although the maximal allowable size of the SOM is fixed and hardware platform dependent, by activating and/or deactivating neurons in function of application needs, the size configurable SOM architecture is obtained.

V. Results

The proposed architecture was tested as color quantizer. The maximal allowable size of the architecture was fixed to 16×16. As inputs, 3-element vectors representing pixels of 128×128 RGB images were used. These input vectors as well as the configuration data were fed externally to all neurons of the SOM, one by one, via external links of the NoC (managed by the user). A snapshot of the obtained results for 4 different configurations is presented in Figure 3. To evaluate the quality of the quantization for each configuration, the Peak-Signal-to-Noise Ratio (PSNR) was calculated and is presented in dB below the gray scale images showing the dissimilarities between the original and reconstructed images. The proposed SOM architecture was first configured as a 5×5 SOM and its structure evolved through 7×7 and 9×8 until 10×10 map. Each configuration was used the time needed to process the entire image. Once the entire image processed, the user decides or not (in function of the obtained quality, evaluated also by the user) to send new configuration data. The proposed architecture allowed to change iteratively the structure of the SOM (and thus the number of used neurons) until a satisfactory quality (evaluated with PSNR) is obtained. Therefore, for each image an adequate structure of SOM with an optimal ratio quality/performance can be obtained. In the presented results in Figure 3, the optimal ratio is obtained with the 9×8 SOM. Real-time performances of the proposed 16×16 SOM-NoC architecture were evaluated and are presented in Table I. The

Table I. Performances of the proposed 16×16 SOM-NoC architecture used as color quantizer (time needed to process one input vector)

Step	Equation	Time [cycles]	Time [ns]
Structure changing	$T_g = T_{Transmission} + T_{(Re)config}$	38	152
Competition	$T_c = T_{DistanceComp} + T_{WinnerSearch}$	277	1108
Adaptation	$T_a = T_{Broadcasting} + T_{Update}$	123	492
Learning	$T_l = T_c + T_a$	400	1600
MCPS	$\frac{16 \times 16 \times 3}{T_c} \times 10^{-6}$	693	
MCUPS	$\frac{16 \times 16 \times 3}{T_l} \times 10^{-6}$	480	

Table II. The synthesis results for a neuron on a Xilinx Virtex-7 XC7VX485T FPGA board

Logic type	Neuron	Router	Available
Number of Slice Registers	8562	1005	607 200
Number of Slice LUTs	6313	1335	303 600
Number of DSP	1	0	2800

performances of each step during one learning iteration were evaluated in clock cycles and for a working frequency of 250 MHz (see implementation results). It can be seen that the structure changing time is depending on the reconfiguration time of all neurons and on the time needed to transmit all configuration data to all neurons (called $T_{Transmission}$).

The proposed architecture was described with VHDL and synthesized on a Xilinx Virtex-7 VC707 FPGA board with the use of Xilinx ISE Design Suite 14.7. The maximum working frequency obtained for this implementation is 250 MHz. The implementation results for neuron and router are shown in Table II.

VI. Conclusion

In the presented work, a hardware configurable NoC-based SOM architecture applied to real-time image quantization was presented. The proposed architecture allows to find iteratively, for each processed image, the adequate SOM structure providing optimal quality and performances. It was validated through simulation on a 16×16 map processing 128×128 RGB images. The time needed to (re-)configure the proposed architecture is estimated to 38 clock cycles.

References

[1] T. Kohonen, *"Self-Organizing Maps, Third Edition"*, Volume: 29, third edition edn. Springer, 2001.

[2] D. Wiklund and D. Liu, *"SoCbus: Switched network on chip for hard real time embedded systems"*. In: Parallel and Distributed Processing Symposium, International, IEEE, 2003.

[3] P.P. Pande et al *"Design of a Switch for Network on Chip Applications"*. In: Circuits and Systems, ISCAS '03. International Symposium on, IEEE, Volume: 5, pp 217-220, 2003.

[4] T. Kuremoto et al, *"Parameterless-Growing-SOM and Its Application to a Voice Instruction Learning System"*. Journal of Robotics, vol. 2010, 9 pages, 2010.

[5] L. Chen et al, *"An improved SOM algorithm and its application to color feature extraction"*. Journal of Neural Comput and Applic (2014), vol: 24, pp 17591770, 2013.

[6] A. De et al, *"A parallel adaptive segmentation method based on SOM and GPU with application to MRI image processing"*. Journal of Neurocomputing, vol. 198, pp 180-189, 2015.

[7] H. Hikawa and Y. Maeda, *"Improved learning performance of hardware self-organizing map using a novel neighborhood function"*. IEEE Transactions on Neural Networks and Learning Systems, vol. 26, pp 2861-2873, 2015.

[8] J. Lachmair et al, *"A reconfigurable neuroprocessor for self-organizing feature maps"*. Journal of Neurocomputing, vol. 112, pp 189-199, 2013.

[9] H. Tamukoh and M. Sekine, *"A Dynamically Reconfigurable Platform for Self-Organizing Neural Network Hardware"*. Springer-Verlag Berlin Heidelberg 2010, ICONIP 2010, Part II, pp 439-446, 2010.

[10] S. Carrillo et al , *"Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers"*. Journal of Neural Networks, vol. 33, pp 42-57, 2012.

[11] M. Abadi et al , *"A Scalable Flexible SOM NoC-based Hardware Architecture"*. Advances in Intelligent Systems and Computing, vol. 428, pp 165-175, 2016.