https://doi.org/10.48048/tis.2023.6539

A Novel NoC-Based Neural Networks Design for Implementation of A Biosensor Solution

Abdelkrim Ghazi[1,*], Mostefa Belarbi[2], Abdallah Chouarfia[1] and Mohamed Kadari[3]

[1]*University of Science and Technology Mohamed Boudiaf, Bir El Djir, Algeria*
[2]*LIM Research Laboratory, University of Tiaret, Tiaret, Algeria*
[3]*Ibn Khaldoun University, Tiaret, Algeria*

([*]Corresponding author's e-mail: abdelkrim.ghazi@univ-usto.dz)

Received: 25 December 2022, Revised: 5 February 2023, Accepted: 14 February 2023, Published: 17 March 2023

Abstract

Graphene-based nanosensors are used in many applications to monitor and survey physical or chemical phenomena. The use of Graphene improves the mechanical, electrochemical, optical, and magnetic properties of nanosensors due to its properties such as physical properties. But there are some characteristics and measurements that must be taken to manufacture such sensors. In this work, a Neural Network (NN) has been proposed that can predict the values of UV (ultraviolet) measurements, which are wavelength and absorption, based on XRD (X Ray Diffraction) measurements, namely diameter (in millimeters) and angle (theta in degrees). The proposed NN was developed on Network-on-Chip (NoC) to take advantage of the high level of parallelism and computing power provided by NoC. In addition, we used an adaptive XY routing algorithm due to its simplicity and allows exploiting multiple paths to route packets to their destination which reduces the number of lost packets due to the high injection rate without introducing any latency. Moreover, we proposed a mapping algorithm to extract maximum performance from the adaptive XY algorithm. The obtained results show the efficiency of the proposed architecture, as it allows packets to take different paths. Thus, the traffic is distributed across the network and significantly reduces packet loss. Moreover, the equal length of these paths allows avoidance of latency.

Keywords: Network on chip, Artificial intelligence, Neural networks, Nano-biosensors, XY routing algorithm

Introduction

Biosensors are currently used in many fields such as Food analysis, the study of Biomolecules and their interactions, drug development, crime detection, medical diagnosis, environmental field monitoring, industrial process control, manufacturing of pharmaceuticals, and replacement of organs. For instance, the use of biosensors in the medical field can lead to a remarkable improvement in patient care, early diagnosis of diseases, and pathogen detection. Gevaed *et al.* [1], designed and synthesized imidazole-functionalized graphene oxide for the detection of progesterone (P4), which is the most important progestogen and plays a key role in the stabilization and maintenance of gestation in mammals. Zvi *et al.* [2], designed a platform that enables the early diagnosis of gynecologic cancer at advanced stages where artificial intelligence techniques have been used to predict the presence of cancer indicators, based on the data collected through nanosensors. those data (DNA sequence) represent the input of the Artificial Neural Networks. The results being the presence and classification of each biomarker. Yet, the fabrication of biosensors requires knowledge and research in multiple disciplines, namely biology, chemistry, and engineering. Moreover, several constraints should be taken into account, such as reliability, stability, convenience, comfort, costs, miniaturization, biocompatibility, real-time, unobtrusive, etc.

Graphene-based materials have been considered superior to other nanomaterials for the development of sensitive biosensors [3]. Graphene-based sensor interfaces have many advantages, including an increased surface charge of the desired ligand due to the high surface-to-volume ratio, excellent conductivity, and a low bandgap that is beneficial for sensitive electrical and electrochemical readings as well as tunable optical properties for optical readings such as fluorescence and plasmonics. Graphene offers a high density of energy and power, as well as specific capacitance in accordance with its high-power structure-based

supercapacitors [4], due to the porous design of graphene, which increases accessibility to ion diffusion and provides high conduction. The use of ionic liquids helps to strengthen this characteristic. Sarmanova *et al.* [5], proposed a fluorescent nanosensor based on carbon dots for the measurement of pH and temperature. In their work, they used Neural Networks to simultaneously determine the value of pH and ambient temperature, achieving an accuracy of 0.0005 pH and 0.67 °C, respectively. This graphene-based nanosensor operates within a temperature range between 22 - 81 °C.

Sensors consist of 2 elements, namely the receptor and transducer. The receptor interacts with the target molecule, whereas the transducer converts the chemical information into measurable signals. In graphene-based biosensors, graphene is used as a transducer that converts the interaction between the receptor and the target molecules into a quantifiable signal [6]. The graphene-based sensors have a lot of promise in the fields of biomedical, food industry, etc., due to their advantages such as nano-size, edge effects, biocompatibility, good conductivity, and quantum confinement. Alongside their atomic thickness that characterizes the graphene layers, which allows direct contact with analytes resulting in high sensitivity compared to silicon. In addition, this also allows for conformal and intimate contact with organs [7]. Trung *et al.* [8], used reduced graphene oxide (rGO) fiber to develop a wearable temperature sensor with high responsivity, fast response time, and satisfactory recovery time. Furthermore, this sensor allows for real-time measurement of body temperature. Another example of the use of graphene-based sensors can be found in the work of Zhu *et al.* [9], where a bifunctional intelligent nanosensing platform based on graphene-like titanium carbide MXene (Ti2C MXene)/ Au-Ag nano shuttles (NSs) was used for both electrochemical and SERS intelligent analysis of ultra-trace carbendazim (CBZ) residues in tea and rice.

However, measurements such as UV, XRD, and dielectric are needed for the fabrication of accurate sensors, but the process of conducting the experiment and taking measurements is costly in terms of time and effort. The proposition of Artificial Intelligence (AI) can allow for the saving of time needed to do the laboratory measurements.

Neural Networks are considered an effective AI technology to solve various problems such as classification, prediction, and pattern recognition to name a few. This technology is characterized by its ability to simulate the biological brain in terms of its working mechanism and learning process, which is why it is named as such.

One of the biggest challenges when building Neural Networks is that they require a lot of parallel processing power. Embedded systems have experienced a significant evolution in terms of performance, leading designers to produce System-on-Chip (SoC) solutions. These systems integrate various elements such as processors, memories, I/O blocks, and communication media.

The evolution of embedded systems has suggested the idea of connecting SoCs in Networks-on-Chip (NoCs) to take advantage of parallelism and minimize runtime and power consumption.

NoCs are currently used in various fields of application due to the high level of parallelism that can be achieved, mainly because the links can work simultaneously on different data packets, which improves the system's performance. NoCs also provide more power efficiency at a lower cost. It is simpler in terms of hardware and routing function, helping to implement a complex system. NoCs reduce routing congestion and provide easy synchronization closure. Furthermore, it is much easier to exchange processing element (PE) blocks to create derivative chips and quickly respond to engineering change orders during the development phase [10].

In our work, we used data collected from the synthesized graphene oxide associated with ionic liquid (GO + IL) [11], using ultrasound irradiation and ionic liquid [12], as a reaction medium. This data was used to create a training dataset for NN. Additionally, we use NN to take advantage of the prospects it offers in order to predict the values of UV measurements of GO + IL based on a given XRD measurements which allows the reduction of time needed for fabrication of graphene-based biosensors.

This study aims to propose a NoC-based architecture to implement an artificial neural network, taking advantage of the high level of parallel computing power offered by NoC. The proposed NN can predict UV measurements with high accuracy, thus saving a considerable amount of time and effort needed for conducting experiments and measurements. Whereas, the proposed architecture uses an adaptive XY routing algorithm and a proposed mapping algorithm to exploit the maximum number of paths to route packets to their destination, resulting in an enhancement in the efficiency of the architecture. Furthermore, we used a router structure with buffers to store incoming packets in the case where packets arrive at a rate faster than what the router can handle. The main contribution of the present work is the combination of the adaptive XY routing algorithm and our proposed mapping algorithm, which aim to exploit as many possible paths. Moreover, we proposed a clustering strategy that considers the communication between different neurons in feed-forward NN where each neuron sends packets to all neurons in the next layer, so the neuron

doesn't send packets to neurons situated within it in the same layer or neurons in the previous layer. The obtained results show the efficiency and accuracy of the proposed approach.

The first part of this paper is devoted to presenting related work where we are going to show some of the challenges related to designing NoC architecture to implement NN and some of the research that has been done in recent years in this area. then we proposed NN which aims to predict the UV measurements from given XRD measurements. Then, we show the training phase of the NN. Finally, we present how we implement the NN into NoC architecture.

Related work

Neural Networks are used with NoC in various applications. For instance, Kun-Chih Jimmy Chen and Yuan-Hao Liao [13], used Artificial Neural Networks to predict the temperature of the NoC system. The aim is to control the temperature of the NoC to extract the best efficiency and performance; compared to the conventional temperature prediction that uses physical parameters, their approach reduces average error by 37 to 62 %, resulting in increased performance by 9 to 38 %.

Research [14-16], shows that NoCs are a powerful tool for implementing Neural Networks due to their high level of parallelism, flexibility, scalability, and low cost. However, despite all these advantages, there are still problems to overcome that are mainly caused by the high number of packets circulating in the network. Thus, it is crucial to implement methods and mechanisms to minimize their impact on the functioning of Neural Networks. Additionally, NoCs are vulnerable to transient or intermittent phenomena due to different factors such as electromigration, electromagnetic interference, cosmic radiation, etc. This can lead to a network system failure; therefore, it is critical to ensure that the NoC tolerates failures. During an outage, tasks can be transferred, relocated or substituted by implementing a self-organized mechanism [17], that works independently without needing external control.

A key element to address these challenges is a Network-on-Chip (NoC) topology that can manage the large volume of packets generated by Neural Networks. There are several topologies available, such as Tree topology [18], Butterfly Fat-tree [19], Octagon [20], Mesh topology, etc. The Mesh topology is the most widely used [21,22], due to its several advantages. It has better power consumption and latency thanks to its fixed wiring length. Additionally, it provides good scalability and is easy to implement with a well-understood structure [23]. Sandeep *et al.* [21], proposed the EMBRACE architecture, a 2-dimensional mesh topology that includes a set of routers (R) to route different packets to their final destination, a set of neurons (N), "Spike Generators" (SGs) to generate packets with different rates and inject them into the network, and "Spike Counters" (SCs) to count the number of packets received by a router. Neural configuration parameters such as synaptic weights and threshold, as well as Spiking Neural Networks (SNN) connection topology, are programmed in order to realize the EMBRACE architecture. In their architecture, the NoC router decodes the destination of the incoming packet and then transmits it either to the neuron (via the local port) or to one of the neighboring routers (via the North, South, East, or West output ports). Based on a routing table stored in the router's configuration memory, the router uses "round-robin" with 8 states (namely N (North), E (East), S (South), W (West), SG (Spike Generator)/SC (Spike Counter)/N (neuron) and 3 additional states used for housekeeping tasks) to poll and maintain its ports. Because the NoC router introduces a delay of a single clock cycle in routing the packet to a neighboring router, the maximum processing time is 9 clock cycles. The result shows that the EMBRACE architecture offers a good reception rate at higher injection rates. For instance, packet loss occurs at about a rate of 1 packet every 15 clock cycles for a 2-router path, and it occurs at about 1 packet every 30 clock cycles for a path of 3 routers. However, at a rate of 1 packet per clock cycle, almost all packets are lost.

To address the huge traffic generated by Artificial Neural Networks, Ouyang *et al.* in their work [24], proposed a multicast mechanism for a NoC-based Deep Neural Networks Accelerator (MMNNN). They used a tree-based multicast routing algorithm to address the degradation of performance due to the high number of one-to-many packet transfers in the unicast channels. Their proposition allows for high throughput and lower latency, while also minimizing the number of transmitted packets. However, the router area increased by 33 %.

Another parameter to take into account while designing a Neural Network is memory. As the Neural Network's size grows, the memory requirements for storing its configuration increase considerably, posing a serious challenge to its scalability. Architectural techniques for reducing the memory requirements of Neural Network topology are needed to ensure efficient and compact hardware behavior.

Modulization and simulation

In this section, we will present the process of preparing data fed to the Neural Networks. Then, we will demonstrate the training process using the MATLAB tool, which aims to obtain the configuration of the Neural Networks to solve the problem of predicting UV measurements (wavelength and absorption) of synthesized graphene oxide associated with ionic liquid (GO + IL) [11], based on XRD measurements (diameter and intensity). Finally, we will present our architecture to implement NN on NoC.

Data preparation

The data preparation is done in 2 stages, namely the stage of preliminary calculations of values and data data crossing. In the following, we will explain these steps in more detail.

Two measurements, XRD and UV, were used. For the XRD measurements, we used a diffractometer, which is a device for measuring the diffraction of radiation on a target. The results of the XRD measurement were the intensity values as a function of 2θ. We determined a frequency (rank value) for each intensity value using the following equation $f = D / 2\pi$, where D represents the diameter (the size of the crystallite which is calculated using Scherrer's formula). For the UV measurement, we used a spectrophotometer to determine the wavelengths of absorbed electromagnetic radiation. The results obtained were absorbance values as a function of wavelength (w), and we calculated their frequency as follows: $f = w / 2\pi$.

After extracting the measurements, the next step is to eliminate the noisy data which is the result of incorrect measurements such as obtaining negative values, and calculate the size of the crystallite using the Scherrer formula, which is essential information to gain a general understanding of the composition of the material studied. Then, we calculate the frequencies using the formula presented above.

In the data crossing step, we calculate the distance between the frequency values of the 2 measurements. In other words, we associate each XRD frequency with the UV frequency that is closest to it. This step will result in forming the data set used to train the NN. Thus, at the end of the data crossing phase, a data set is created that links these XRD and UV measurements to each other. The obtained data set contains the diameter and intensity (XRD measurements) along with their counterparts from the UV measurements, namely absorption and wavelength. The resulting data set contains 1,202 rows to be used as a training set for NN.

Table 1 Biases and synaptic weight of the NN.

Neuron	Biases	Inputs		Outputs	
		Theta (deg)	Diameter (nm)	wavelength	absorbance
Neuron 1	−8.563227836	6.392449127	6.540957	0.217391	0.511342
Neuron 2	7.259265641	−5.000084322	−7.26654	0.414984	−0.44807
Neuron 3	−5.049343477	8.470778475	2.216345	−0.36988	−0.01797
Neuron 4	−3.60425241	9.06009799	−0.175	0.117298	−0.82584
Neuron 5	0.452707173	−0.624864509	−8.75406	−0.54361	0.364355
Neuron 6	−3.141442749	5.536282053	−6.60776	0.042334	2.379774
Neuron 7	−2.134456214	−6.589638802	−4.86572	−0.00043	−0.35572
Neuron 8	−4.718465471	−8.2309162	2.783072	0.679358	0.38977
Neuron 9	6.489951214	1.417080919	−9.25321	−0.27669	0.169634
Neuron 10	8.589122267	8.645658922	−0.77402	−1.18812	−0.01355

Training of the neural networks

In this step, we use the data obtained in the previous step to train the Neural Network. We used MATLAB tools to train the Neural Network. We divide the data into 2 sets; the first set is used in the training process and contains 80 % of the data from the original set, and the second set is used in the validation process and contains the remaining 20 % of data from the original set. In the validation process, we compare the result obtained by the Neural Network with the real result to evaluate it, and we choose one with smallest error margin.

The Neural Network with the best result, shown in **Figure 1**, has 2 neurons in the input layer that correspond to XRD measurements (the size of crystallization and the angle). It has a single hidden layer, composed of 10 neurons, which uses a sigmoid activation function. The output layer has 2 neurons that correspond to the 2 UV measurement, namely the wavelength and absorption value. **Table 1** shows the synaptic weights and biases of the Neural Network, obtained after the training phase using MATLAB tools.

Proposed method

In the following, we are going to present our architecture to implement Neural Networks in a NoC design. We are going through all the elements that make up our architecture, starting from the packet structure until the routing algorithm.

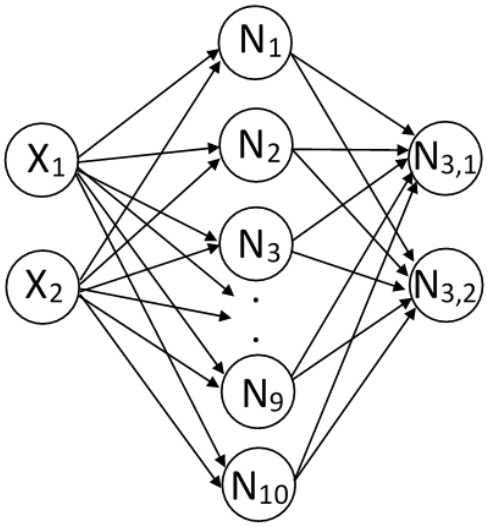


Figure 1 Proposed artificial neural networks layout.

Packet

A packet is composed of 48 bits. The most significant 4 bits are used to identify the final destination of the packet, with the first 2 bits representing the X coordinate and the other 2 bits representing the Y coordinate. The next 12 bits are not currently used, but they can be utilized in future works for error checking, such as using a parity bit, or for implementing a mechanism for detecting and resending lost packets. The remaining 32 bits represent the data being transmitted.

Router

The role of the router is to forward packets to their next destination. The proposed router [25,26] (as shown in **Figure 2**) consists of 4 blocks for each direction: North, South, East and West, plus an additional block (local port) which connects the router with a Processing Element (PE) (computing unit). Each block contains 2 sub-blocks. The first one deals with incoming packets and the second with outgoing packets. The first sub-block is the input block and is composed of 2 elements: An input port that receives the incoming packet and an input buffer that stores the packet until the router determines which output port it must be forwarded to. The second sub-block consists of an output port that allows the packets to reach their next destination and 3 buffers that store packets in case the next router is occupied.

Routing algorithm

The XY routing algorithm is a well-suited choice for mesh topologies. In this algorithm, the next destination of the packet is determined by comparing the X and Y coordinates of the router and those of the destination stored in the packet. The comparison begins with the X coordinate, followed by the Y coordinate, so that the packet will travel along the X-axis of the mesh before proceeding along the Y-axis. This algorithm is efficient in terms of memory since it eliminates the need for a routing table. However, since a packet always follows only one path, it can lead to congestion on these paths despite alternative paths being available, resulting in a bottleneck in system performance.

To distribute the traffic of packets across the network, we used an adaptive XY algorithm which is similar to the traditional XY algorithm. But in this algorithm, a packet can be forwarded through the Y axis

if the packet cannot be forwarded through the X axis (for instance, if the output port is occupied) and the Y coordinate of the packet is different from that of the router. In other words, the packet travels first along the X-axis, but if it encounters an occupied output port, it will be routed through the Y-axis temporarily unless it is on the same Y-axis as its destination. In this case, it has to wait until the output port is no longer busy. This reduces waiting time and allows traffic to be distributed over the entire NoC platform, resulting in improved performance.

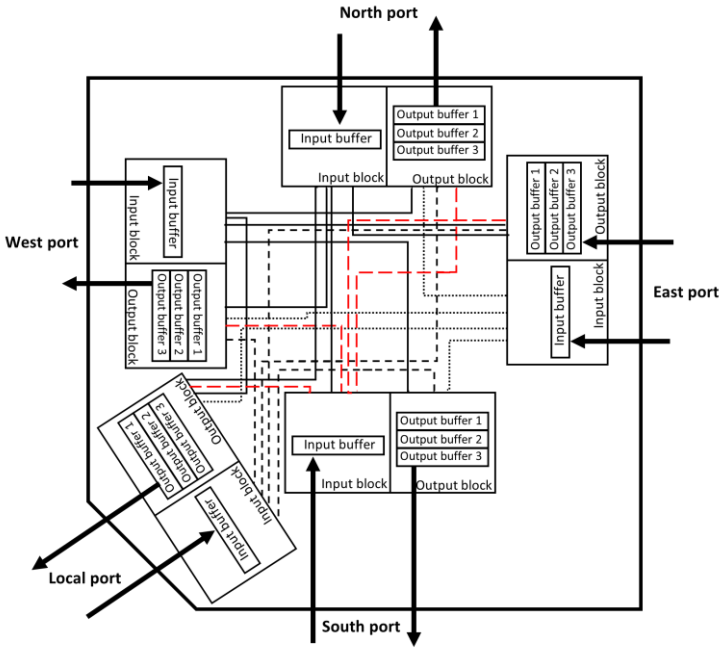


Figure 2 The structure of the proposed router.

Neuron clustering

The clustering of neurons enables the implementation of complex and larger neural networks on a small-size NoC platform. Instead of assigning the tasks of a single neuron to a single PE, by clustering, multiple neurons can be assigned to a single PE, which allows for the sharing of resources such as memory between neurons. Additionally, this will reduce the traffic of packets.

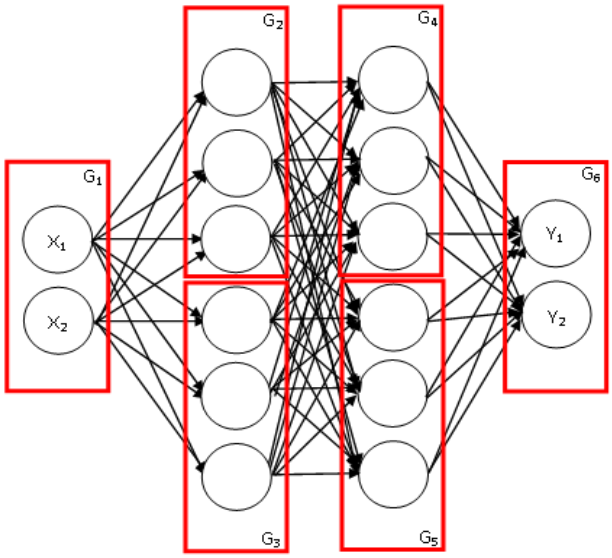


Figure 3 The strategy of clustering neurons. Neurons are divided into 6 groups each one is assigned to a single PE.

The maximum number of neurons for the PE depends primarily on the computing capacity of the PE. For instance, in **Figure 3** we have an NN with 2 neurons in the input and output layer plus 2 hidden layers with 6 neurons each. For the sake of simplicity in this example, we assumed that the PE could handle a maximum of 3 neurons. After clustering, we obtain 6 groups of neurons; each group will be assigned to a single PE. It should be noted that for the mapping algorithm it is mandatory that each PE is assigned only neurons from the same layer to increase performance. In other words, we are not allowed to assign neurons from 2 different layers to the same PE.

Computing unit

The computing unit (Processing Element) plays the role of multiple neurons in the Neural Networks, so it receives data from neurons in the previous layer and sums it up, then transmits the result to all neurons in the next layer. Since a computing unit plays the role of multiple neurons instead of just 1 neuron, it allows the architecture to be easily scaled to encompass a larger NN.

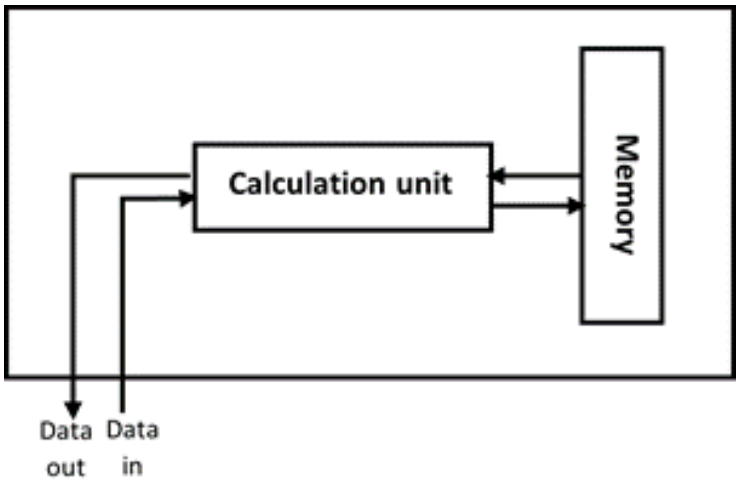


Figure 4 The structure of computing unit.

Each computing unit is attached to a local port of a router in the mesh topology. The structure of the computing unit is presented in **Figure 4**, as we can see it is composed of 4 components:

Input port: The input port is used to retrieve packets from the switch.

Memory: It stores the structure of the Neural Networks, i.e., the coordinates of all the neurons in the next layer, along with the synaptic weights; so basically, they store the configuration of our NN.

The calculation unit: Its role is to calculate the output of the neuron by using the synaptic weights stored in memory to produce packets which contain the result alongside its destination address.

The output port: Its role is to inject the packet built by the computing unit into the NoC by transmitting it to the connected switch.

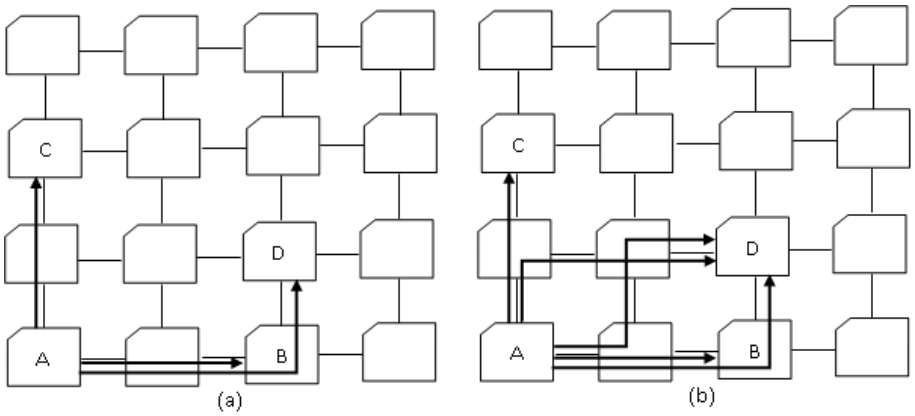


Figure 5 Possible paths for packet using (a) traditional XY routing algorithm and (b) adaptive XY routing algorithm.

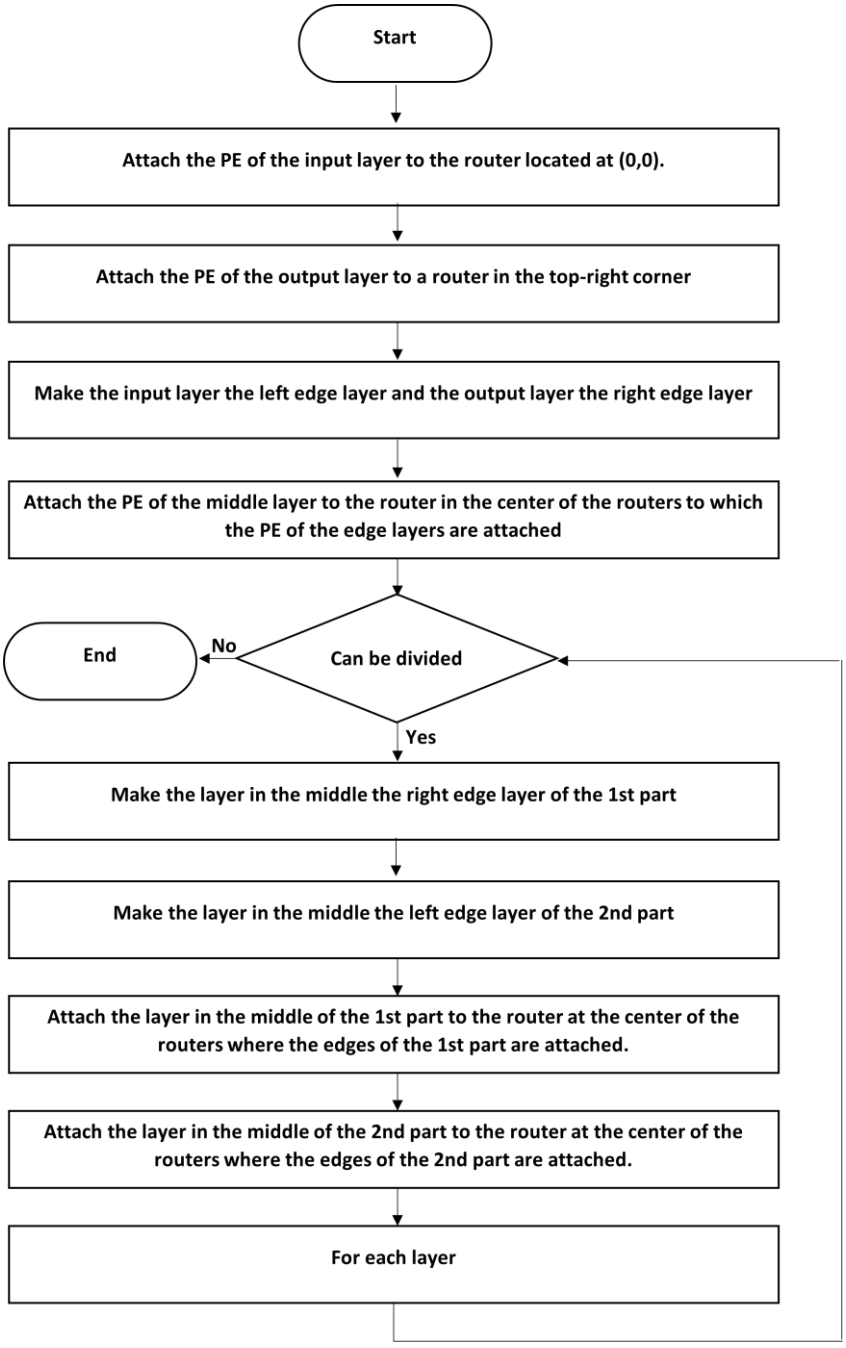


Figure 6 The flowchart of the proposed mapping algorithm.

Neuron mapping

The adaptive XY routing algorithm allows distributing the traffic over the network to avoid congested points in the network, unlike the traditional XY routing algorithm where all packets are routed through only one path which could result in loss of packets if the packets are sent at a rate higher than the router can handle. However, to extract the maximum performance from the adaptive XY routing algorithm, it is necessary to place the PE cores at a distance from each other on both axes. As we can see from **Figure 5**, as the destination and source PE cores are further apart and not on the same X and Y axes, the number of paths increases. For instance, there is only one possible path between A to B and A to C in the traditional and adaptive XY algorithms. But between A and D there is 1 path using the traditional XY routing algorithm and 3 possible paths using the adaptive XY routing algorithm. To extract the best performance from the adaptive XY routing algorithm, we proposed a mapping algorithm (**Figure 6**) that can be applied to an NoC with a mesh topology.

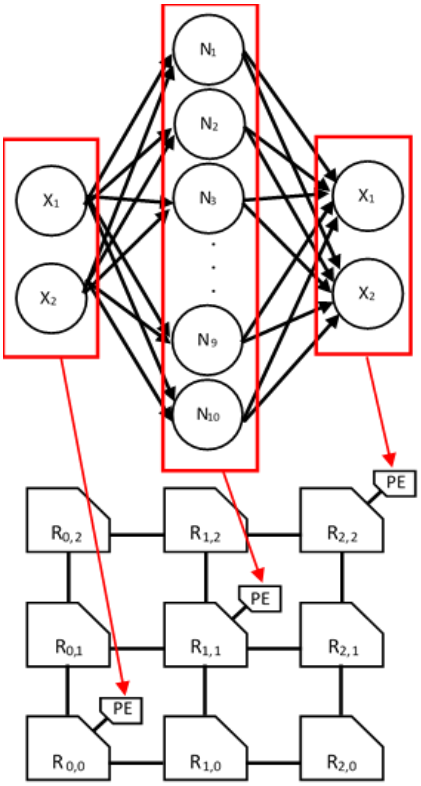


Figure 7 Clustering and mapping the proposed NN on 3x3 NoC.

The proposed algorithm aims to position clusters of layers that communicate with each other as far apart as possible on both axes to take advantage of adaptive XY routing algorithms. To accomplish this, the algorithm calculates the placement of the PE corresponding to the cluster of the middle layer situated between the edge layers. It then divides that section into 2 parts, where the middle layer serves as an edge layer in each part. The algorithm then calculates the placement of the PE corresponding to the cluster of the middle layer in each part. Initially, the input and output layers serve as edge layers. With each iteration, the algorithm divides the NN into 2 parts, where the first part has the first edge layer and the middle layer whose coordinates were just calculated, and the second edge layer and middle layer as the second part's edge layers. For each part, the algorithm repeats the same steps to calculate where to attach the cluster of the its middle layer. The process is then repeated for each part until all clusters are attached to the appropriate routers. Using Eqs. (1) - (2), the coordinates of the router to which the PE core of the middle layer should be attached can be calculated. The variables x and y represent the coordinates of the router where the PE core of the middle layer should be attached, while x_s, x_d, y_s, and y_d represent the coordinates of the routers to which the PE cores of the edge layers are attached. Initially, the edge layers are the input layer and output layer, which are placed in the bottom-left corner and top-right corner, respectively. With each iteration, the NN is divided into 2 parts and the edge layers are updated so that all PE cores are attached to the appropriate routers. It should be noted that if the number of neurons in a layer exceeds the capacity of a single PE core, the tasks for these neurons are assigned to multiple PE cores, and one PE core is placed using the mapping algorithm while the other is placed on the same Y-axis.

$$x = \frac{x_s + x_d}{2} \quad (1)$$

$$y = \frac{y_s + y_d}{2} \quad (2)$$

As shown in **Figure 6**, the mapping algorithm starts by connecting the PEs of the input and output layers to the bottom-left and top-right routers of the mesh NoC, respectively. The input and output layers are then set as the left and right edge layers. The next step involves connecting the PEs of the middle layer to the appropriate router using Eqs. (1) - (2). If the NN can be divided into 2 parts, the algorithm divides it into 2 halves, where the middle layer becomes the right edge layer in the first half and the left edge layer in

the second half. For each half, the middle layer is connected to the appropriate router using the previously mentioned equations, and the algorithm checks if it can be divided further into 2 halves. This process is repeated until it's no longer possible to divide each part into 2 halves, meaning all PEs have been placed within the mesh NoC. The middle layer is the layer in the center between the edge layers. For example, if the NN has the 3 hidden layer and the input and output layers are the edge layers, then in the first iteration, the second hidden layer is the middle layer. In the second iteration, the second hidden layer serves as the edge layer for the 2 halves, the 1st hidden layer is the middle layer for the first half, and the third hidden layer is the middle layer for the second half.

The first step of the process involves placing the PE core that serves as the input and output layers in the bottom-left corner and top-right corner, respectively. In the second step, the PE core of the layer between the edge layers whose PEs are connected to the router. The PE core should be attached to the router in the center of the 2 routers where the edge layers have been attached. Using Eqs. (1) and (2), the coordinates of the router to which the PE core should be attached can be calculated. The variables x and y represent the coordinates of the router where the PE core should be attached, while x_s, x_d, y_s, and y_d represent the coordinates of the routers to which the PE cores of the edge layers are attached. In other words, first, the edge layers correspond to the input layer and the output layer is placed in the bottom-left corner and top-right corner, respectively. Then, we calculate where to put the layer in the middle of the input and output layers. Then, we repeat the same process but this time the edge layers correspond to the input layer and the layer located in the middle of the input layer and output layer. Basically, we consider the 2 layers whose PE are attached to a router as the 2 layers on the edge, then we calculate to which router we should connect the PE of the layer which is in the middle of those edge layers. As mentioned above, this step should be repeated until all PE cores are attached to their router. It should be noted that in the case where the number of neurons in a layer exceeds the capacity of the PE core, then the tasks of neurons of this layer are assigned to multiple PE cores and we placed one of those PE cores using the mapping algorithm and the other should be placed on the same Y-axis.

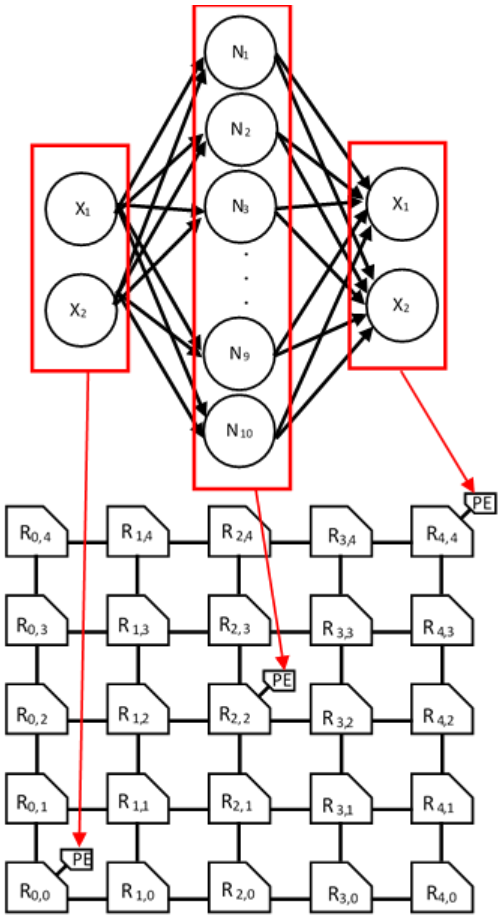


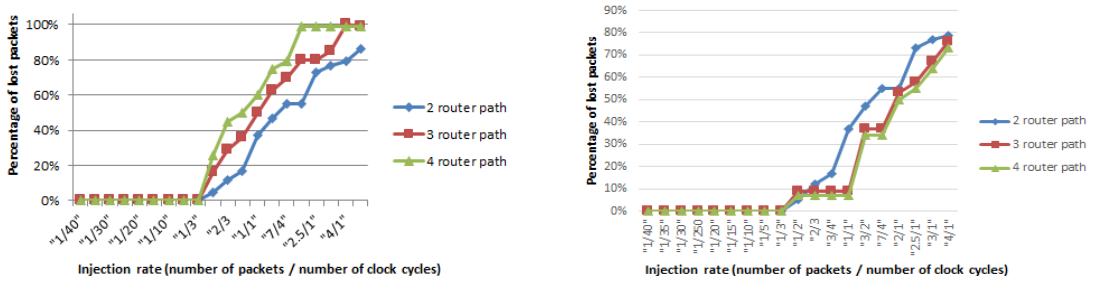
Figure 8 Clustering and mapping the proposed NN on 5x5 NoC.

For instance, **Figure 7** shows the NN obtained from the training phase mapped onto a 3×3 NoC. The first step is to connect the PE of the input layer to the router in the bottom left corner, and the PE of the output layer to the top right corner. Specifically, the PE of the input layer is connected to the router with coordinates (0,0), and the PE of the output layer is connected to the router with coordinates (2,2). In the next step, the PE of the hidden layer is connected to the appropriate router. Based on the equations presented, the PE of the hidden layer is connected to the router with coordinates (1,1).

In **Figure 8**, the same NN is mapped onto a 5×5 NoC. The PE of the input layer and the output layer are placed at the corner routers with coordinates (0,0) and (4,4), respectively. The hidden layer is connected to the router with coordinates (2,2). As we can see, the difference is in the number of paths between the PEs. In the first mapping, packets have to cross 3 routers to reach their destination and there are only 2 possible paths. However, in the second mapping, packets have to cross 5 routers, but there are 5 possible paths, reducing the chance of packets being lost.

Result and discussion

In this study, we employed a technique of crossing data to create a dataset to train a NN. The results showed that the trained NN provided remarkable precision, as it was able to predict the values of UV measurements based on XRD values with an average error of ≈6 for wavelength and ≈0.6 for absorbance. However, a more accurate prediction could be achieved by increasing the size of the training set through additional laboratory XDR and UV measurements.



(a) (b)

Figure 9 Packet loss rate per packet injection rate (in number of clock cycles); (a) using traditional XY routing algorithm (b) our architecture with the Adaptive XY routing algorithm.

Regarding the NoC, we implemented the NN using a mesh topology and an adaptive XY routing algorithm. To further enhance performance, we proposed a mapping algorithm that aims to extract the best performance from the adaptive XY routing algorithm. Additionally, the use of a clustering technique reduced the number of packet circulations within the NoC. To evaluate the performance of our proposed approach, we used the percentage of lost packets as a performance indicator. In regards to the NoC, we implemented a NN using a mesh topology and the adaptive XY routing algorithm. To further improve performance, we proposed a mapping algorithm that aims to extract the best performance from the adaptive XY routing algorithm. Additionally, the use of clustering technique reduces the number of packet circulation within the NoC. To evaluate the performance of our proposed approach, we used the percentage of lost packets as a performance indicator. As the packet injection rate and path length (i.e. the number of routers a packet must pass through to reach its destination) impact the percentage of received packets, we calculated the percentage of received packets for various injection rates and path lengths for both the traditional XY routing algorithm and adaptive routing algorithm using our proposed approach. The results are summarized in **Figure 9**. From the graphs, we can see that as the packet injection rate increases, the packet loss rate also increases due to the router's inability to forward a packet stored in its buffer before the arrival of another packet. Additionally, as the path becomes longer, the possibility of this event increases, resulting in a greater loss of packets. As for the adaptive XY algorithm, it allows for the distribution of traffic over a mesh network. This algorithm allows for multiple paths of the same length for packets to reach their destinations, resulting in improved performance compared to the traditional XY algorithm when the source and destination are located on different X and/or Y-axes. However, if the source and destination

are located on the same X or Y-axis, only one path is possible and thus, the performance of both the adaptive and traditional XY algorithms will be the same.

As we can see from the graph in **Figure 9**, the adaptive XY algorithm does not provide an enhancement for paths with only 2 routers, as in this case, only one path is possible to forward the packet. However, for paths with 3 or 4 routers, where there are respectively 2 and 4 possible paths in the case of the adaptive algorithm with our proposed mapping algorithm, we noticed some improvements in terms of received packets. For example, at a rate of 1 packet every clock cycle, we saw that the packets lost decreased from around 50 % in the traditional XY routing algorithm to only 10 % with the adaptive XY routing algorithm for a path with 3 routers, and from 60 to 10 % for a path with 4 routers. Even in the case of a higher injection rate such as 4 packets every clock cycle, where all packets were lost in the traditional XY routing algorithm, we saw that by using the adaptive XY routing algorithm, the packets lost dropped to around 75 %. This represents a significant improvement compared to the EMBRACE architecture mentioned above in section 2, where almost all packets were lost at a rate of 1 packet in each clock cycle. It should be noted that the greater the distance between the sending and the receiving router, as long as they are not situated on the same X or Y axis, the lesser the effect of packet loss compared to the traditional XY algorithm, this is due to the increasing number of paths that a packet can follow to reach its destination. Instead of packets being lost because all buffers are occupied by packets waiting to be forwarded to the next destination so the new arriving packets can't be stored and consequently, they will be lost, in the adaptive XY routing algorithm, those waiting packets can be forwarded through other available paths, and thus the input buffers are free to store the new incoming packets. It should be noted that this algorithm does not introduce any latency since all suggested paths by this algorithm are of the same length.

Using the XY algorithm allows the reduction of the time required to calculate the next destination of the packet since it is determined by a simple comparison between the destination coordinates stored in the packet with those of the router rather than accessing a routing table. Additionally, the use of the adaptive XY algorithm enables the exploitation of all possible paths to forward packets to their final destination, resulting in the avoidance of congestion without introducing any latency as all paths have the same length. This further increases the performance and efficiency of the proposed architecture. The mapping algorithm enables the placement of PEs that communicate with each other as far apart as possible on both axes, resulting in many paths that packets can follow to reach their destinations. This allows for the full potential of the adaptive XY routing algorithm to be utilized. Furthermore, neuron clustering reduces the number of packets circulating in the NoC, which improves the performance of the proposed architecture. Additionally, the use of buffers with 3 levels allows packets to be stored and reduces the possibility of losing a packet before it is transmitted to the appropriate outgoing port.

Conclusions

In this paper, we cross-referenced data from both UV and XRD measurements based on their frequency, which allowed us to form a training set for the NN. We then used MATLAB tools to train the NN and chose the best-fit NN to solve the problem of predicting UV measurements based on given XRD measurements. We then reviewed our NN NoC-based architecture with a mesh topology in which we used a router with buffers to store incoming and outgoing packets, which allowed us to minimize the number of lost packets. As for the routing algorithm, we used an adaptive XY routing algorithm, which is well-suited for the mesh topology and provides multiple paths for packets to reach their destination compared to the traditional XY algorithm. This enhances the performance of the proposed architecture without introducing any latency.

Regarding the placement of neurons in the NoC platform, we first demonstrated our clustering strategy that reduces the number of packets injected into the NoC and allows for better resource sharing between the different neurons. Then, we presented our mapping algorithm that distributes clusters in a way to exploit as many paths as possible for the adaptive XY algorithm, thus enabling better traffic management within the NoC and resulting in further enhancement in terms of reducing lost packets. The use of an adaptive XY routing algorithm alongside the proposed mapping algorithm and clustering strategy allows for increasing the number of paths that packets can take to reach their destination. Since all paths are of the same length and the shortest paths possible, the proposed method doesn’t introduce any additional latency. The length of the path may affect the time needed to obtain the final output of NN. However, in cases where packets are lost, it results in not producing the final output of NN which makes the proposed approach provide a good compromise between the time needed to obtain the output of NN and minimizing the effect of lost packets. The results show that the proposed architecture has an excellent packet reception rate even at higher packet injection rates. This work is a contribution towards designing a robust biosensor. In the

future, we aim to evaluate the performance of the proposed architecture by studying other characteristics such as energy consumption.

References

[1] A Gevaerd, SF Blaskievicz, AJG Zarbin, ES Orth, MF Bergamini and LH Marcolino-Junior. Nonenzymatic electrochemical sensor based on imidazole-functionalized graphene oxide for progesterone detection. *Biosens. Bioelectron.* 2018; **112**, 108-13.

[2] Z Yaari, Y Yang, E Apfelbaum, C Cupo, A Settle, Q Cullen, W Cai, KL Roche, D Levine, M Fleisher, L Ramanathan, M Zheng, A Jagota and D Heller. A perception-based nanosensor platform to detect cancer biomarkers. *Sci. Adv.* 2021; **7**, eabj0852.

[3] SK Krishnan, E Singh, P Singh, M Meyyappan and HS Nalwa. A review on graphene-based nanocomposites for electrochemical and fluorescent biosensors. *RSC Adv.* 2019; **9**, 8778-881.

[4] SI Wong, H Lin, J Sunarso, BT Wong and B Jia. Optimization of ionic-liquid based electrolyte concentration for high-energy density graphene supercapacitors. *Appl. Mater. Today* 2020; **18**, 100522.

[5] O Sarmanova, K Laptinskiy, M Khmeleva, S Burikov, S Dolenko, A Tomskaya and T Dolenko. Development of the fluorescent carbon nanosensor for pH and temperature of liquid media with artificial neural networks. *Spectrochim. Acta Mol. Biomol. Spectros.* 2021; **258**, 119861.

[6] M Pumera. Graphene in biosensing. *Mater. Today* 2011; **14**, 308-15.

[7] CIL Justino, AR Gomes, AC Freitas, AC Duarte and TAP Rocha-Santos. Graphene based sensors and biosensors. *TrAC Trends Anal. Chem.* 2017; **91**, 53-66.

[8] TQ Trung, HS Le, TML Dang, S Ju, SY Park and NE Lee. Freestanding, fiber-based, wearable temperature sensor with tunable thermal index for healthcare monitoring. *Adv. Healthc. Mater.* 2018; **7**, e1800074.

[9] X Zhu, P Liu, T Xue, Y Ge, S Ai, Y Sheng, R Wu, L Xu, K Tang and Y Wen. A novel graphene-like titanium carbide MXene/Au-Ag nanoshuttles bifunctional nanosensor for electrochemical and SERS intelligent analysis of ultra-trace carbendazim coupled with machine learning. *Ceram. Int.* 2021; **47**, 173-84.

[10] S Feero and PP Pande. Networks-on-Chip in a three-dimensional environment: A performance evaluation. *IEEE Trans. Comput.* 2009; **58**, 32-45.

[11] M Kadari, EH Belarbi, T Moumene, S Bresson, B Haddad, O Abbas and B Khelifa. Comparative study between 1-Propyl-3-methylimidazolium bromide and trimethylene bis-methylimidazolium bromide ionic liquids by FTIR/ATR and FT-RAMAN spectroscopies. *J. Mol. Struct.* 2017; **1143**, 91-9.

[12] SK Singh and AW Savoy. Ionic liquids synthesis and applications: An overview. *J. Mol. Liq.* 2020; **297**, 112038.

[13] JC Kun-Chih and L Yuan-Hao. Adaptive machine learning-based temperature prediction scheme for thermal-aware NoC system. *In*: Proceedings of the 2020 IEEE International Symposium on Circuits and Systems, Seville, Spain. 2020.

[14] A Amani and D Mohammadyani. *Artificial neural networks: Applications in nanotechnology*. *In*: CLP Hui (Ed.). Artificial Neural Networks. IntechOpen, London, 2011.

[15] D Vainbrand and R Ginosar. Network-on-Chip architectures for neural networks. *In*: Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip, Grenoble, France. 2010.

[16] PC Holanda, CRW Reinbrecht, G Bontorin, VV Bandeira and RAL Reis. DHyANA: A NoC-based neural network hardware architecture. *In*: Proceedings of the 2016 IEEE International Conference on Electronics, Circuits and Systems, Monte Carlo, Monaco. 2016.

[17] M Heil and C Tanougast. Fault-tolerant self-organized mechanism for networked reconfigurable MPSoC. *In*: Proceedings of the 2014 International Conference on Control, Decision and Information Technologies, Metz, France. 2014.

[18] Z Marrakchi, H Mrabet, U Farooq and H Mehrez. FPGA interconnect topologies exploration. *Int. J. Reconfigurable Comput.* 2009; **2009**, 259837.

[19] MAAE Ghany, MA El-Moursy and M Ismail. High throughput architecture for high performance NoC. *In*: Proceedings of the 2009 IEEE International Symposium on Circuits and Systems, Taipei, Taiwan. 2009.

[20] G Reehal, MAAE Ghany and M Ismail. Octagon architecture for low power and high performance NoC design. *In*: Proceedings of the 2012 IEEE National Aerospace and Electronics Conference, Ohio. 2012.
[21] S Pande, F Morgan, G Smit, T Bruintjes, J Rutgers, B McGinley, S Cawley, J Harkin and L McDaid. Fixed latency on-chip interconnect for hardware spiking neural network architectures. *Parallel Comput.* 2013; **39**, 357-71.
[22] VI Vaishali and AG Mahendra. Review of mesh topology of NoC architecture using source routing algorithms. *Int. J. Comput. Appl.* 2013; **2**, 101-4.
[23] P Kalpana and MA Gaikwad. Review of different topologies for noc architecture using NS2. *Int. Res. J. Eng. Tech.* 2018; **5**, 2885-7.
[24] Y Ouyang, F Tang, C Hu, W Zhou and Q Wang. MMNNN: A tree-based multicast mechanism for NoC-based deep neural network accelerators. *Microprocessors Microsystems* 2021; **85**, 104242.
[25] A Ghazi, M Belarbi, A Ghazi, M Belarbi and A Chouarfia. Contribution to the development of a rapid prototyping platform model for reconfigurable nano-biosensors based on nanotechnologies. *Procedia Comput. Sci.* 2019; **151**, 891-6.
[26] H Daoud, C Tanougast, M Belarbi, M Heil and C Diou. Formal proof of the dependable bypassing routing algorithm suitable for adaptive networks on Chip QnoC architecture. *Systems* 2017; **5**, 17.