A Multineuron-Based Routing Algorithm of Tile-Based 2-D Mesh

Neetu Faujdar,1 Akash Punhani,2 Ankush Agarwal,1 Pradeep Kumar,3 and Ajitesh Kumar1

1Department of Computer Engineering and Application, GLA University, Mathura, India
2Department of Computer Science and Engineering, SRM Institute of Science and Technology, Delhi-NCR Campus, Ghaziabad, Uttar Pradesh, India
3Discipline of Electrical, Electronic and Computer Engineering, University of KwaZulu-Natal, Durban 4041, South Africa

Correspondence should be addressed to Neetu Faujdar; neetu.faujdar@gmail.com

Received 14 February 2022; Revised 9 April 2022; Accepted 27 April 2022; Published 30 May 2022

Copyright © 2022 Neetu Faujdar et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Tile-based architecture is broadly utilized in the structuring of the system-on-chip by different vendors. Nevertheless, the performance of the system-on-chip (SoC) is incredibly influenced by the performance of the network hidden on a chip named network-on-chip (NoC). Routing of network-on-chip network chips plays a crucial role in the overall performance of the NoC. In this paper, we have proposed a routing algorithm that utilizes the neural network to perform the routing. This routing algorithm updates the route dependent on the port execution of the switch. From the outcome, the execution of the directing has worked successfully and has the option to deal with the enormous burden viably. The result obtained shows that the performance metric for the uniform traffic is slightly better in comparison to XY routing at the higher loads of 80%. In the case of neighbor traffic, bit complement traffic, and tornado traffic, these values are higher on 80% of the load. The reason for better handling of the loads is due to the parallelization due to the pipeline created by the neural network routing decision.

Tile-based architecture is broadly utilized in SoC, which depends on NoC for its correspondence [1]. The NoC comprises three prime factors that influence the exhibition of the total system; they are topology, routing algorithm, and flow control component. The tile design is called two-dimensional work. The mesh architecture is well-known among the inquires because of its reality. It is basic in structure and profoundly adaptable plan and requires less territory on the chip [2]. As the driving part of the topology is the steering calculation, in this manner, the different directing calculations are accessible for two-dimensional work topology. The most well-known steering calculation for two-dimensional work topology is XY directing calculation. The primary preferred position of XY steering calculation is that the port determination calculation chooses the port dependent on the source and goal of the parcel. This straightforward rationale of port choice aids in taking the choice continuously and does not require any tables. The significant impediment of this routing algorithm is the fixed way chosen by the calculation and will not change with congestion or failures in the topology. Different routing algorithms have been recommended in the past to make the adaptive routing algorithms like odd-even routing algorithm, north last, and west first based on the turn model. The adaptive algorithms restore the pair of ports as opposed to restoring a solitary [3]. The choice of the port is dependent on the obligations defined in the NoC. In the greater part of the cases, congestion in the network goes about as the premise of the port choice. The committed channels help in passing on the data in the network. The purpose behind choosing the committed channel is that if the data about the congested paths is transmitted on the congested network. At that point, the congested information is hindered in the system, and substitutes may not be chosen promptly [4]. The structure of NoC appears in Figure 1.

Another significant issue in the versatile steering calculation is that the choice of the port relies upon dodging the blocked switch at the later degree of directing. In any case,
If the parcel is directed to the goal in the beginning bounces, this will expand the presentation enormously. The reality behind this hypothesis is that at a point there is a blockage in certain switches, its backpressure will make a clog-like impact on the neighboring switches [5].

This research study focuses on a proposed routing algorithm that utilizes the neural network to perform the routing. This routing algorithm updates the route dependent on the port execution of the switch. From the outcome, the execution of the directing has worked successfully and has the option to deal with the enormous burden viably.

2. Literature Survey

NoC (network-on-chip) is used to put up unproductive shared bus architecture on SoC (system-on-chip). Routers are used to build NoC that controls the traffic in the middle of devices and wires which is used to connect routers to the devices and each other. Topology, switching, and routing are the three most difficult aspects of NoC. Evaluating the performance of the network routing algorithm is an effective factor. In this paper, the authors examined the best solution for routing and introduced a relevant routing algorithm that uses the heuristic method. The proposed routing algorithm reduced power consumption and latency in routing time and increased bandwidth. The proposed routing algorithm is compared with deterministic routing. The comparison showed that the proposed algorithm improved the network parameter such as latency, power consumption, and bandwidth [6].

In today’s era, multimedia applications are very susceptible to the quality of service (QoS) framework. The multimedia data is given to the QoS and received the appropriate throughput of packets. This paper’s objective is to enhance the throughput of multimedia data, especially for smart cloud networks by splintering the packets into optimal size. In the proposed approach, throughput of the packets is increased and calculation time is decreased [7].

Wireless sensor network (WSN) is promptly growing in today’s era. Sensor nodes are built battery-powered and used in dangerous or unreachable circumstances. The sensor node’s battery cannot be replaced and recharged. A clustering method is a powerful approach to gaining energy efficiency in WSN. Cluster heads are chosen for all sensor nodes in cluster-based routing algorithms. The major drawback is that there is no control over the distribution of cluster heads over the network. In this paper, authors proposed swarm intelligent-based routing protocol (SIF) which overcomes the major drawback. A fuzzy C-means clustering algorithm is used in the SIF. The proposed algorithm is an assurance to achieve balanced clusters over the network. The main motive of the proposed algorithm is to prolong the network lifetime based on the application specifications. SIF simulates over 10 heterogeneous networks and evaluated that SIF outperforms the existing ones [8].

Ant colony optimization (ACO) is a nature-inspired, population-based metaheuristic. ACO is used to solve conditionally tough problems. In this paper, the authors described the parallelization strategy to utilize the inherently stochastic and distributed nature of the method. The proposed approach is used to balance workload effectively by assigning jobs dynamically to heterogeneous resources. The implementation results show that significant improvement has been achieved in terms of both solution quality and energy expenditure [9].

The quality of service (QoS) is supported by multicast-constrained routing for real-time multimedia in wireless mesh networks (WMNs). The major drawback is the difficulty of applying strict admission control to a public WMN. In this paper, the authors proposed a multiconstrained routing algorithm for WMNs. The proposed algorithm achieved 19.6% good put of live video streaming applications and reduced 33% routing overhead with the comparison of existing techniques [10].

The multicast is an essential method in wireless mesh network (WMN). Nowadays, many applications in WMN use multicast TV, audio, and video conferencing. The major challenge in multicast transmissions is security, and without this network, services are always disordered. The existing routing protocols are vulnerable due to the subtle nature of flaws in protocol design. In this paper, the authors proposed Secure Multicast Routing Algorithm for Wireless Mesh Network (SEMRAW). This algorithm is resistant to all known active threats including wormhole attacks. SEMRAW pays digital signatures to check a malicious node from gaining illegitimate access to the message contents. The security of SEMRAW is estimated using the replication paradigm method [11].

Nowadays, IoT is having opportunistic networks. These networks are important based on the historical encounters of these network routing which can improve message delivery quality where nodes meet regularly. The major drawback of the opportunistic networks is the inability to obtain sufficient encounter information, i.e., cannot accurately predict whether there is an encounter between nodes. In this paper, the author’s motive is to improve the accuracy in the environment of a sparse opportunistic network. So the authors proposed the ONBTM algorithm and defined the concept related to node intimacy. By comparing with existing techniques, ONBTM outperforms better in the case of...
adaptability and stability. ONBTM is more suitable for sparse opportunistic networks [12].

High performance of the system can be achieved by the network-on-chip (NoC) on a single chip. The famous technologies are mesh and torus to achieve the performance by the NoC. More technologies can explore and reduce the latency of the packet delivered from one node to another node on the chip. In this paper, the authors proposed a new variant of torus topology, and the performance of the network is measured on the simulator by using various traffics. The latency of cubic torus has been compared with various topologies. It has been observed that the proposed torus topology has the least latency compared with the existing one [13].

Congestion is a very important issue in wireless sensor networks (WSN). Congestion energy loss, packet delay, and packet drops are major disadvantages of WSN. So it is compulsory to propose an optimal congestion routing that considers network parameters. In this paper, the authors have proposed congestion aware routing using fuzzy rule sets (CARF). CARF is having the advantage to handle excess traffic conditions, and also, it alleviated the path to the sink node using fuzzy rule prediction. Due to this, packet loss and energy utilization have been increased. CARF is a framework into two segments, namely, (1) multiple path identification by positioning nonlocalized nodes and (2) congestion-mitigated routing of data packets to sink node. The first segment is used for positioning a nonlocalized node which is used to compute the unknown coordinates of the sensor node. The second segment used an enhanced fuzzy-based congestion mitigation (ECFM) algorithm for the estimation of congestion levels in nodes using fuzzy rule sets. The concluded results proved that the proposed CARF is mainly used to reduce conjunction as well as reduced energy cost [14].

Nowadays, agricultural research is on-demand using sensor networks and data mining techniques. One research gap in agriculture is the regulation of the quantity of water in a cultivated field. WSN is an emerging technology in the field of an agricultural field but also has a major problem which is the utilization of energy and enhancing the lifetime of the sensor nodes. In this paper, the authors proposed a new routing protocol that provides the data to the irrigation system. The proposed routing algorithm used the fuzzy rules and terrain-based routing protocol. The proposed algorithm is compared with two routing protocols that are region-based routing and equalized cluster head election routing protocol. The comparison shows that the proposed algorithm performs better than the existing ones [15].

WSN is used to design the IoT system. In today’s life, IoT is integrated into digital devices, sensing equipment, and computing devices to sense data and communicate the data to the base station. But in WSN based IoT systems, security and energy efficiency are the design challenges and require enhancement of the network lifetime. To overcome these challenges, in this paper, author proposed a secure energy-aware cluster-based routing algorithm named trusted energy-efficient fuzzy logic-based clustering algorithm (TEEFCA). The proposed algorithm considered the two objectives. First, reliable nodes are recognized which work as candidate nodes for the cluster-based routing. Secondly, the fuzzy inference system is earned under the two situations, namely, selection of optimal cluster leader (CL) and cluster formation process by considering the following three parameters such as (i) node’s residual energy level, (ii) cluster density, and (iii) distance node BS. Experimental results show that the proposed algorithm performs better in terms of power conservation, network stability, and lifetime than the existing one [16].

Mobile ad hoc network has dynamic nature. In dynamic nature, it contains a lack of centralized control and management, severe resource constraints, and frequent changes in topology. The dynamic nature of mobile ad hoc requires extra overhead in the provision of secured and stable routing. To overcome this issue, authors proposed an innovative integrated approach for secure routing. The proposed algorithm used two algorithms which are fuzzy-based stable and secure routing algorithm and trust-based next forwarding node selection algorithm. The main advantage of the proposed node selection procedures is isolating the malicious nodes from the routing process which enhances the security. From the experimental results, it has been proved that the proposed algorithm increases the network performance [17].

The wireless mesh network (WMNs) is growing rapidly nowadays. The multi-interface multichannel (MIMC) technique has been used as the backbone for WMNs. The ad hoc on-demand distance vector (AODV) routing algorithm is used for WMN. The main challenge occurs when the legacy AODV is used in MIMC WMNs. To resolve this problem, an interface assignment-based AODV (IA-AODV) is used. IA-AODV is having PREQ prediction scheme, the PREQ sender assignment scheme. The detailed network conditions have been analysed and a detailed operation is introduced. The proposed approach to MIMC WMNs is compared with the existing AODV routing protocol. The comparison shows that proposed MIMC WMNs using the IA-AODV routing protocol outperform better from existing AODV [18].

The wireless sensor network (WSN) is having two important parameters which are energy efficiency and fault tolerance. Fault tolerance can be improved by multipath routing in WSN. In this paper, the authors considered energy consumption and proposed the multipath routing algorithm for WSN based on a genetic algorithm. The fitness function has been computed by the proposed algorithm using the distance between nodes in the network and then generating the routing scheme at the base station. The proposed scheme is shared with all nodes in the network. The experimental result shows that the proposed scheme performs better compared to existing ones [19].

Many routing algorithms have been seen in the wireless sensor network (WSN). The cluster-based protocol is one of the important routings in WSN. The challenge in this protocol is the uneven distribution of cluster heads because some nodes may run out of energy too early which is unsuitable for WSN. To overcome this problem in this paper, authors proposed distributed clustering algorithm which is based on fuzzy-weighted attributes. The proposed algorithm
ensures energy efficiency. The experimental results show that the proposed algorithm has a longer life expectancy and better extensibility than LEACH-like algorithms [20].

3. Proposed Model

As the router is structured utilizing the neural network, we start our conversation from the very building block of the neural network, that is, the neuron. Figure 2 portrays a straightforward neuron, which comprises two primary highlights. In the first stage, the aggregation of all inputs to the neuron is done. Condition (1) depicts the collection of weighted data sources.

\[ y_i = \sum_{j=1}^{n} x_j y_j. \]  

(1)

Here, \( x_i \) is the input parameter and \( y_i \) is the weight related to the specific.

Fundamentally, the router comprises three layers, the initial layer is the input layer that will peruse the four data sources \( X, X_2 \) which are the \( X \) coordinates of the source, and destination \( Y, Y_2 \) is the \( Y \) coordinates of the source and destination. The hidden layer is structured by four neurons \( N_1 \) to \( N_4 \). Additionally, the output neuron comprises neurons from \( N_5 \) to \( N_7 \). The activation function for the input layer is a personality work and the actuation capacity of neurons \( N_1 \) to \( N_7 \) is a binary step function. Condition (2) depicts the binary step function.

\[ f(x) = \begin{cases} -1, & \text{if } x < 0, \\ 1, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases} \]  

(2)

The portrayal of the binary step function is depicted in Figure 3, which shows that the result of the step function remains at zero for all values less than equal to zero yet changes from zero to one for a value greater than one.

3.1. Weight Assignment Procedure for Neural Network

3.1.1. Input Layer–Hidden Layer Weight Assignment Procedure. Figure 4 portrays the architecture of the neural network of the router. Loads of the various layers are allocated numerically dependent on the congestion condition of the router. The inputs to the neural network are the coordinates and the maximum of the two ports are in the

![Figure 2: The neuron.](image)

![Figure 3: The binary step function.](image)

![Figure 4: The architecture of the neural network model for the router.](image)
direction of the shortest path. The first two neurons are representing the \(X\) coordinates, and the remaining two neurons are representing the \(Y\) coordinates; therefore, there is no relation between the neuron representing \(X\) and \(Y\) coordinates. In Figure 4, we can see that there is no link between them.

(a) \(\{X_1, X_2\} \) with \(\{N_3, N_4\}\)
(b) \(\{Y_1, Y_2\} \) with \(\{N_1, N_2\}\)

Such links are represented with the weight zero. In our assumption, we have assigned neuron \(N_1\) for the west port, neuron \(N_2\) for the east port, neuron \(N_3\) for the north port, and neuron \(N_4\) for the south port. If the shortest path is selected, then the maximum of two neurons results in the output as one. For getting the output as per the desired configuration, we have hardcoded weights and the weight matrix \(W\) is represented by

\[
W = \begin{bmatrix}
  1 & -1 & 0 & 0 \\
  -1 & 1 & 0 & 0 \\
  0 & 0 & 1 & -1 \\
  0 & 0 & -1 & 1
\end{bmatrix}.
\]  

(3)

3.1.2. Hidden Layer–Output Layer Weight Assignment Procedure. The weight assignment in the hidden layer is the learning result, which is based on the XY routing algorithm outcome. The XY routing algorithm is an oblivious routing algorithm that fixes the decision parameters for the topology. The result obtained from the hidden layer is represented in a bipolar representation, and the table for the result can be described as given below.

Presently, from Table 1, our calculation has three yields as in the work. We have four directions and the fifth is the local port that associates the processing element to the router. These five states require at least three bits to be represented uniquely. The output \(Y_3\) is utilized to characterize the router or local port. If the bit is 1, then the data packet is transferred to the local port; otherwise, the packet is routed according to the direction represented in Table 1.

Presently, the learning procedure of each perceptron is done independently. The training process for Perception O1: Primarily based on the XY routing algorithm, \(Y_1\) is just influenced by two input sources \(X_1\) and \(X_2\), that is, the packet needs to travel first in the \(X\)-direction than in \(Y\)-direction. Therefore, to reduce the number of epochs and reduce the complexity of learning, the other two inputs are not considered and their weights might be set to constant as zero. The learning process for the perception is done by keeping two weights and bias as zero. The value of \(\alpha\) is set to one. The weights are adjusted until there is no change in their values of weight and bias, initially considering the output \(Y_1\). The minimized table after ignoring the inputs \(X_3\) and \(X_4\) and outputs \(Y_1\) and \(Y_2\) is presented in Table 2.

**Table 3:** The weights are updated in the given order.

<table>
<thead>
<tr>
<th></th>
<th>Epoch 1</th>
<th>Epoch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_1)</td>
<td>1 -1 -1</td>
<td>1 -1 -1</td>
</tr>
<tr>
<td>(X_2)</td>
<td>-1 -1 1</td>
<td>-1 -1 1</td>
</tr>
<tr>
<td>(t)</td>
<td>-1 -1 -1</td>
<td>-1 -1 -1</td>
</tr>
<tr>
<td>(w_1) old</td>
<td>0 -1 -2</td>
<td>0 -1 -2</td>
</tr>
<tr>
<td>(w_2) old</td>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>(b) old</td>
<td>0 -1 0</td>
<td>0 -1 0</td>
</tr>
<tr>
<td>(w_1) new</td>
<td>-1 -2 -1</td>
<td>-1 -2 -1</td>
</tr>
<tr>
<td>(w_2) new</td>
<td>1 0 -1</td>
<td>1 0 -1</td>
</tr>
<tr>
<td>(b) new</td>
<td>-1 0 -1</td>
<td>-1 0 -1</td>
</tr>
</tbody>
</table>
Algorithm 1 shows the learning algorithm for the perceptron and its parameters.

The observation is fruitful to learn on the chance that we can get the line of straight separability. The learning procedure is finished utilizing Python 3, and the plot for straight separability after each adjustment of the weight has been introduced. The training of the perceptron is finished in two epochs itself, as there is no adjustment in weight in the second epoch which is listed in Table 3.

This algorithm is implemented on the HNOC using OMNet++. To update the weights, the function is being presented that updates the proportion of the packets on the X and Y dimension of the chip to control the congestion in the network. This congestion control depends on the proportion of the packet sent acknowledgment received by the particular router on a specific port. To keep this log, we have utilized eight variables that get the track of the sent packets on the specific port and received acknowledgment on the specific port. Based on these values, the proportion is assessed and the weight is controlled to increment to choose the desired port, either in the X direction or in the Y direction. Figure 5 describes the learning of the neural network at different epochs. From Figure 5(a) we can observe that in the initial stage, the line of separability is having a different trend, as the training of the network increases, the line
changes its orientation in Figure 5(b) and finally change to a
different orientation that can give exact discrimination for
the selection of the ports in Figure 5(c).

The flowchart of the implementation of proposed model
has been described in Figure 6.

An example describing the implementation of the rout-
ing algorithm is demonstrated in Figure 7.

Let us consider the figure assuming the node (1,1) wants
to communicate to the node (3,3). The router of node (1,1)
receives the packet which is the neural network as described
in Figure 4.

The values input layer are assigned as follows:

\[ X_1 = 1 \]
\[ X_2 = 3 \]

Table 4: The various parameter used for the simulation of the topology.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh configuration (row × columns)</td>
<td>8 × 8</td>
</tr>
<tr>
<td>Bandwidth of channel (in Gbps)</td>
<td>16</td>
</tr>
<tr>
<td>Virtual channels allowed</td>
<td>2</td>
</tr>
<tr>
<td>Size of flit in bytes</td>
<td>4</td>
</tr>
<tr>
<td>No. of packets per message</td>
<td>2</td>
</tr>
<tr>
<td>Flits per packet</td>
<td>8</td>
</tr>
<tr>
<td>Interflit arrival delay</td>
<td>Based on the load factor</td>
</tr>
<tr>
<td>Warm-up time in nanoseconds</td>
<td>20</td>
</tr>
<tr>
<td>Total time of simulation</td>
<td>200</td>
</tr>
<tr>
<td>Queue size for packets</td>
<td>16</td>
</tr>
</tbody>
</table>
| Traffic patterns that are used                  | Uniform traffic, bit comple-
tment traffic, neighbor traffic,
and tornado traffic |
| Number of test run (averaged)                   | 10                          |

Figure 8: Various features on uniform traffic: (a) throughput, (b) latency, and (c) performance metric.
Y1 = 1
Y2 = 3

Based on the values and weights provided by the matrix W, the perceptron computes the input values for N1 to N4 in the figure that is hidden layer.

\[ N_{1\text{inp}} = X_1 \ast w_{11} + X_2 \ast w_{12} + Y_1 \ast w_{13} + Y_2 \ast w_{14} \]
\[ = 1 \ast 1 + 3 \ast -1 + 1 \ast 0 + 3 \ast 0 = 1 - 3 = -2 \]

\[ N_{2\text{inp}} = X_1 \ast w_{21} + X_2 \ast w_{22} + Y_1 \ast w_{23} + Y_2 \ast w_{24} \]
\[ = 1 \ast -1 + 3 \ast 1 + 1 \ast 0 + 3 \ast 0 = 1 - 3 = 2 \]

\[ N_{3\text{inp}} = X_1 \ast w_{31} + X_2 \ast w_{32} + Y_1 \ast w_{33} + Y_2 \ast w_{34} \]
\[ = 1 \ast 0 + 3 \ast 0 + 1 \ast 1 + 3 \ast -1 = 1 - 3 = -2 \]

\[ N_{4\text{inp}} = X_1 \ast w_{41} + X_2 \ast w_{42} + Y_1 \ast w_{43} + Y_2 \ast w_{44} \]
\[ = 1 \ast 0 + 3 \ast 0 + 1 \ast 1 + 3 \ast -1 = 1 - 3 = 2 \]

Once we get the compute the output using the activation function:

\[ N_{1\text{out}} = -1 \]
\[ N_{2\text{out}} = 1 \]
\[ N_{3\text{out}} = -1 \]
\[ N_{4\text{out}} = 1 \]

These inputs are forwarded to the next layer of the neuron (output layer); in the discussion, we have learned that a neural network is trained to give the input according to Table 1 and is equivalent to the 7th row so the output is o1 = −1, o2 = 1, and o3 = −1. Meaning the packet is moving in the east direction. The current location of the packet is now (2,1). Again computing using the formulae X1 = 2, X2 = 3, Y1 = 1, and Y2 = 3, then we will get N1 = −1, N2 = 1, N3 = −1, and N4 = 1 which means again move in the east direction. The new current location of the packet is (3,1); using the formula, we will now get the value of N1 = −1, N2 = −1, N3 = −1, and N4 = 1; this implies that from Table 1, we have to move towards the south direction. Therefore, the new address will be (3,2). Applying the same computations, the values generated are N1 = −1, N2 = −1, N3 = −1, and N4 = 1 which can be inferred as the south direction from Table 1. The new location of the packet is (3,3) now, the value of N1 = N2 = N3 = N4 = −1 that is the core from Table 1. Hence, the packet has reached its destination successfully.

3.2 Experimental Setup. To evaluate the routing algorithm’s performance, OmNet++4.4.1 with the HNOCS version was used to simulate an 8 × 8 mesh topology [21, 22]. Table 4 shows the complete setup of the topology design.

The test parameters are considered for the analysis of the performance of average latency, average throughput, and performance metrics.

For average throughput, several packets received at the sink to the number of packets sent by the source in the stipulated amount of time under observation are called average
throughput. Equation (4) shows the formula to calculate the average throughput.

\[
\text{Average throughput} = \frac{\sum \text{packets received at different nodes}}{\sum \text{packets send from different source}}.
\tag{4}
\]

Average latency is the total time taken by packets to travel from source to destination. The formula of average latency has been described in

\[
\text{Average latency} = \frac{\sum \text{timestamp } \text{pkt received} - \text{timestamp } \text{pkt source}}{\text{Total number of pkt}}.
\tag{5}
\]

Performance metric is computed as the ratio of the average throughput to the average latency of the packets measured in a certain period [15]. Equation (6) shows the formula to calculate performance metric.

\[
\text{Performance metric} = \frac{\text{Avg. throughput}}{\text{Avg. latency}}.
\tag{6}
\]

The various graphs portray the performance of the routing algorithm on different traffic patterns at different loads.

In Figure 8(a), the performance of the NN model-based approach is slightly less than that of the XY routing algorithm, but it is better than the odd-even adaptive routing algorithm with a load greater than 70%. Figure 8(b) portrays the latency of the packet delivered in the network; this can be observed from the figure that the performance of the NN model is better in comparison to XY and the odd-even routing algorithm but has a slightly high load of 90% for IX/Y routing algorithm. Figures 8(a) and 8(b) portray the distinctive picture to consider the consolidated impact of the two; we have examined the performance metric of the equivalent which has been described in Figure 8(c).

Figures 9(a)–9(c) portray the performance of the network analysis on neighbor traffic. The premise of the NN model is the XY routing and we have a limited number of hops from the graph; it can be identified that the performance of NN routing remains identical to the XY routing. It is observed that the adaptive routing like odd-even tries to transfer the packet on a different path that there is high latency. However, in the case of odd-even routing, the performance is poor due to congestion created in other parts of the topology by the adaptive nature of the routing algorithm.
The bit complement traffic is designed by taking the complement of the source id. In this case, Figures 10(a)–10(c) describe the performance of the network with different parameters. We observed that the NN model is much better than the other three routing algorithms at a higher load above 70%. The adaptive routing algorithm like odd-even has poor performance; this fall in performance can be seen in both the throughput and latency. The main reason for the slag in the performance is the movement of all packets to the particular nodes. Those are having low load and even moving far away from the shortest path intended to reach before the deadline of the packets.

From Figures 11(a)–11(c), the performance in terms of throughput for XY routing is better; however, if we consider the latency of the network, minimal latency is for the IX/Y routing algorithm. Again, by observing the performance metric, we observed that NN routing is having good performance with higher loads greater than 70%. In the case of tornado traffic, the traffic is moving from almost half of the topology and creating an almost similar scenario to the bit complement traffic and even harder in terms of congestion. Hence, the performance of odd-even routing is affected in a similar pattern as seen in terms of bit complement.

There are latest algorithms like [23] that have compared their routing algorithm with the traditional algorithms like XY, IX/Y, odd-even routing, west first, and north last routing algorithm which have used machine learning KNN for the routing implementation. The major issue with this study is that such a study is based on the application and requires complex architecture or extra knowledge about the congestion.

4. Conclusion

From the observation, it is reasoned that the neural network can be successfully used to control the routing of the network on a chip. We can see that it works better for higher burden factors. Even though the performance of XY routing is better than that of NN routing for the lower load. However, it is better in contrast with the odd-even routing, which is versatile. Another significant perception is that the throughput and latency are high if there should arise an occurrence of XY routing algorithm. However, the consolidated performance of the routing algorithm is observed when the execution metric is being figured. Accordingly, it is recommended that the exhibition metric is the better instrument to examine the performance of the network viable. In the future, we are planning to add fuzziness to the determination of the routing algorithm, hence getting a progressively versatile nature of the routing algorithm. The result obtained shows that the performance
metric for the uniform traffic is slightly better in comparison to XY routing at the higher loads of 80%. In the case of neighbor traffic, bit complement traffic, and tornado traffic, these values are higher on 80% of the load. The reason for better handling of the loads is due to the parallelization due to the pipeline created by the neural network routing decision.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

There are no conflicts of interest declared by the authors for the publication of this manuscript.

References


