ITUJournal
Future and evolving technologies

ITU Journal on Future and Evolving Technologies, Volume 6, Issue 2, June 2025

ARTICLE

Energy-efficient data-driven routing for hybrid interconnection networks: A machine learning approach

Md Tareq Mahmud (Graduate Student Member, IEEE)[1], Uma Maheswar Reddy Nagirireddi[1], Ke Wang (Member, IEEE)[1]

[1] *University of North Carolina at Charlotte, USA*

Corresponding author: Md Tareq Mahmud (Graduate Student Member, IEEE), mmahmud1@charlotte.edu

Modern high-performance computing systems have undergone a significant transformation with the adoption of chiplet-based multi-die integration. This approach enables scalable improvements in computational and communication performance while reducing energy consumption and manufacturing costs. However, existing chiplet-based interconnection network designs face challenges in meeting the stringent latency and energy efficiency requirements of edge computing systems, particularly for multicast and broadcast communication. Silicon interposers impose high costs in inter-chiplet communication, primarily due to their restricted throughput capacity. Similarly, wired interconnection designs are ill-suited for managing multicast and broadcast traffic, as their design limitations, such as high hop counts, inadequate bandwidth, and resource constraints, contribute to significant power consumption and latency. Chiplet-based hybrid interconnection designs overcome these challenges by integrating both wired and wireless interconnects that can adapt to diverse communication patterns and requirements. This paper introduces a data-driven, machine learning-based dynamic routing framework that intelligently adapts to communication needs and directs traffic to wired, interposer, or hybrid communication links by analyzing workload patterns. Several machine learning models are trained on hybrid interconnection network attributes under different communication types (unicast, multicast, and broadcast), and the best performing classifier is selected to dynamically determine the optimal communication link at runtime. The simulation results show that the proposed machine learning-based dynamic routing framework achieves a maximum 64% end-to-end latency reduction, a 2.28 × throughput improvement, and 3.14 × improved energy efficiency compared to the baseline wired Network-on-Chip (NoC). These findings underscore the potential of data-driven routing in advancing energy-efficient and high-performance interconnection network designs for future edge computing platforms.

Keywords: Chiplet, edge computing, energy efficient, hybrid interconnection, machine learning, wireless

1. INTRODUCTION

The accelerated evolution of semiconductor technology has brought about more complex System-on-Chip (SoC) architectures, with modern computing shifting to multicore and multi-chiplet designs [1]. This is motivated by the demand for increased processing capacity, energy efficiency, and low-latency data communication of contemporary Artificial Intelligence (AI) applications and big data analysis. Conventional interconnect methods are faced with scalability issues in these emerging designs, especially in managing increased traffic density and congestion. To address such concerns, Wireless Network-on-Chip (WiNoC) architectures have emerged, which provide wired and wireless communication for better flexibility [2, 3]. Despite such advantages, hybrid interconnect routing remains an issue when it comes to dynamic traffic patterns, congestion control, and power limitations, where static routing models cannot keep up with real-time optimization [4].

In traditional Network-on-Chip (NoC) designs, static routing methods are based on precomputed static routes that do not change at runtime [5]. Although efficient for deterministic workloads, they are poorly suited for contemporary heterogeneous workloads with variable packet injection rates and latency-sensitive traffic. The rigidity of static routing typically results in energy wastage and performance degradation, particularly under sudden traffic spikes. Therefore, adaptive routing methods have become an essential requirement for effective and scalable NoC operation under dynamic workloads [6].

A Machine Learning (ML) classification model is a model that predicts data in known classes, predicting the category of new data points based on patterns from historical data [7]. Common classifiers include Perceptron, Support Vector Machine (SVM), naive Bayes, decision trees, logistic regression, and K-Nearest Neighbors (KNNs). Predictive modeling is also a classifier that predicts the best behavior by analyzing historical and current data trends, applying data mining and machine learning techniques to guide decision-making. ML algorithms, such as linear regression, Support Vector Machine (SVM), and Artificial Neural Networks (ANNs) are optimizers that maximize future outcomes. These models have been widely applied for classification and predictive purposes, for example, prediction of stock market trends, prediction of natural disasters, weather prediction, prediction of viral outbreaks, disease prediction, fraud detection, optimization of crop yields, detection of anomalous behavior, detection of cyber threats, prediction of carbon emissions, and monitoring of air quality [8, 9].

Machine Learning (ML) has emerged as a viable solution to address routing decision optimization problems in hybrid WiNoC architectures [10, 11]. In contrast to static rule-based solutions, ML-based approaches learn and adapt dynamically to network dynamics based on real-time information to optimize routes, reduce latency, and improve energy efficiency. Through the analysis of network parameters such as channel availability, buffer level, and traffic patterns, ML-based algorithms make smart decisions about whether a packet must be transmitted over wired or wireless links and route packets dynamically based on existing conditions.

ML has been successfully applied in interconnection networks for fault-tolerant routing, traffic management, and dynamic voltage scaling[12]. For example, reinforcement learning has been shown to optimize NoC routing policies quite well, while supervised learning has been used for proactive path adaptation and traffic classification. Most of the ML-based work has mainly targeted wired NoCs with minimal or no work on hybrid interconnect designs that incorporate both wireless and wired communication.

This paper proposes a new ML-based routing framework for hybrid WiNoC architectures to combat the drawback of static routing with data-dependent adaptive decision-making. We aim to perform a comprehensive and comparative analysis of various machine learning models for dynamic routing applications, including decision tree, random forest, Support Vector Machine (SVM), logistic regression, and artificial neural networks. Our ML-model-based solution dynamically selects routing paths based on network conditions, optimally trading off performance and energy efficiency. In contrast to conventional static approaches, which adhere to pre-decided fixed paths, our solution continuously adapts with runtime network conditions, minimizing latency, as well as energy consumption. Hence, the key contributions of this research are as follows:

- **Comprehensive and comparative analysis of ML models:** This study performs a comprehensive and comparative analysis of various ML models to be applied in hybrid interconnection networks for dynamic routing decisions.
- **ML-based routing framework:** We proposed an ML-based routing framework for hybrid interconnection networks applying the efficient ML model, where a dynamic routing mechanism selects wired or wireless channels based on real-time network parameters.
- **Improved adaptability:** The proposed ML-based routing framework dynamically adjusts routing strategies to handle varying traffic loads and congestion. This ensures better adaptability compared to hybrid designs with static routing strategies.
- **Scalability and flexibility:** The proposed ML-based routing framework is extremely scalable with various hybrid designs and adaptable to different computing environments.
- **Energy efficiency optimization:** ML-based routing decisions of the proposed framework reduce packet retransmission and congestion, leading to significant energy savings.
- **Comprehensive performance evaluation:** We conducted a detailed comparative analysis against hybrid designs with static routing techniques to demonstrate the effectiveness of our proposed dynamic routing approach in terms of latency, throughput, and energy efficiency.

This paper highlights the revolutionary nature of ML-based interconnection networks, bridging the gap between static rule-based routing and adaptive data-driven routing, and paving the way for efficient and scalable NoC architectures. The simulation results show that the proposed machine learning-based dynamic routing framework achieves a maximum 64% end-to-end latency reduction, a 2.28 × throughput improvement, and 3.14 × improved energy efficiency compared to the baseline wired NoC. Moreover, the proposed framework achieves a significant performance improvement compared to basic hybrid interconnection design with human-

engineered routing (33% end-to-end latency reduction, 1.44 × throughput improvement, and 1.56 × improved energy efficiency).

The rest of the paper is organized as follows. Section 2 describes the previous interconnection topologies and justifies the use of machine learning application in adaptive routing. Section 3 presents the chiplet-based hybrid interconnection architecture and the proposed hybrid router architecture. Section 4 explains the process of generating data based on BookSim2 [13] and proposes the feature selection approach. In addition, it outlines the supervised models used, including the training and testing procedures. Section 5 describes the experimental results and compares the classifiers with a number of performance measures. Section 6 summarizes the key results and proposes future potential extensions to increase the flexibility and real-world application of the framework.

2. BACKGROUND AND MOTIVATION

In order to link high-performance and low-power computer systems, new interconnection networks that operate with multicore and multi-chiplet designs have been developed. Due to their ability to overcome the limitations of wireline networks, WiNoC designs have been recognized as the leading ones among them. Through wireless networks, WiNoC design helps boost bandwidth, reduce latency, and eliminate hop counts, but it can scale up to the amount of traffic. However, in routing and resource allocation, this hybrid feature creates additional issues where the more static designs are often inadequate.

Static models that use preplanned routing paths and pinned policies are the fundamental aspects of traditional NoC [14]. These models are typical for scenarios with stable and constant throughput load because of their conceptual and computational efficiency. However, in conditions where connected systems operate in dynamic and heterogeneous networks, it is hard to apply the principles of static models when the intensity of traffic, channel utilization, or workload varies [15]. For example, high traffic loads may time out on the static models, making effective rerouting of packets a challenge; hence congestion and increase in latency time.

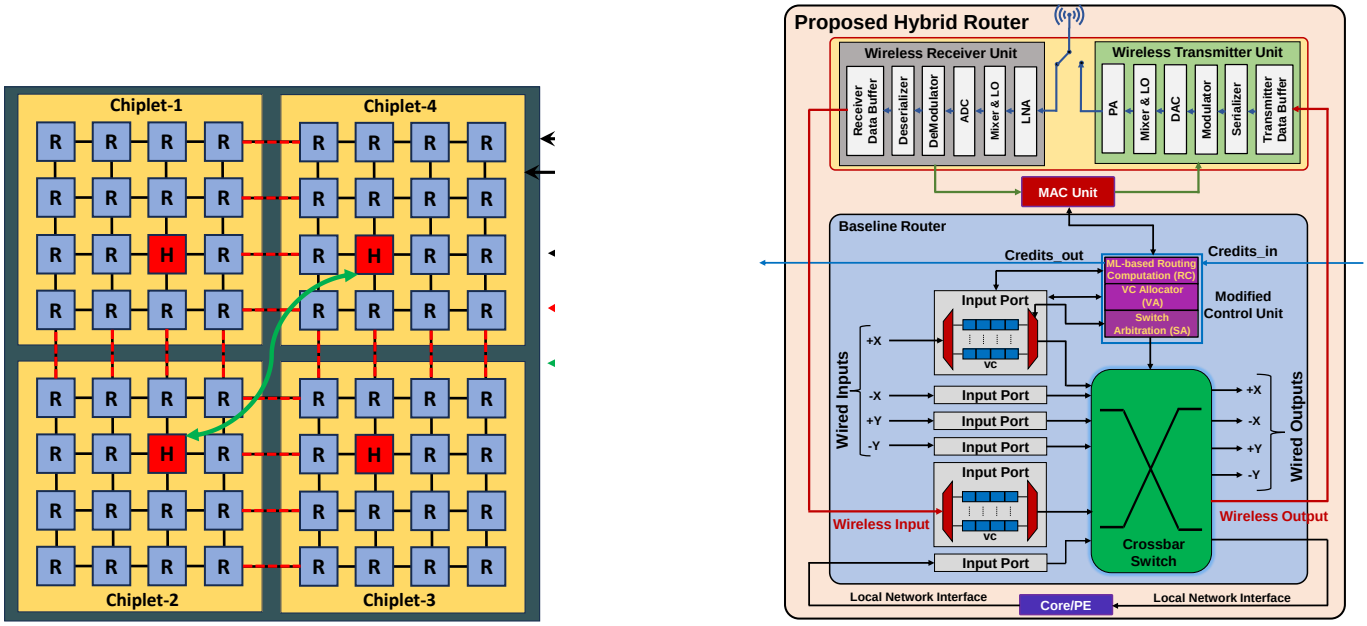
The major drawback of static models is, therefore, exaggerated in hybrid interconnection systems through which both wired and wireless links must be synchronized [2]. The use of wireless links increases the number of factors that affect the links, including interference, path loss, and energy limitations, which cause static models to lack the flexibility to accommodate. This gap has prompted researchers to look for other solutions that can adapt to changes within the hybrid network environment.

The application of machine learning to address these issues with interconnection networks, especially the capacity for data-driven decision-making, holds promise [16, 17]. Through processes such as data mining, traffic data can be analyzed using the diverse ability of ML algorithms to locate areas of congestion and make estimates on which path is most suitable for adoption. This capability can be most useful in WiNoC architectures where real-time adaptability is a key enabler to realize high deterministic performance and low power consumption. If implemented with ML, routing decisions can be made effective based on packet priority, buffer state, and channel quality, making the best use of the available network resources [18].

A recent research proposed a hybrid interconnection network design that considered threshold-based routing [2]. However, the application of ML can solve many issues in such static routing-based hybrid interconnection network design. For example, reinforcement learning has been applied to routing algorithms to self-adapt throughput and latency in different traffic conditions [19, 20]. Supervised learning methods have been used in traffic pattern analysis to qualify traffic and set up appropriate routing policies [21]. However, most of the previous work operates in a wired NoC environment and there is not enough work done to introduce wireless links into the hybrid architecture [22].

By incorporating significant advances from earlier research and using a machine learning technique to hybrid WiNoC systems, this study expands our knowledge. We suggest a more sophisticated approach that uses online analysis. Here, ML provides clear advantages over static models when it comes to scalability. However, static models are shown to be less efficient as the size and complexity of interconnection networks increase, and new structures need to be modeled and optimized by hand. On the other hand, with ML-based models, you can forecast the scenarios across different topologies and do not need constant configuration assistance; thus, new systems can be deployed much faster, with supervised learning to dynamically control packet delivery via wired and wireless channels in place of traditional techniques. By reducing the detrimental impacts of frequent transmissions and uneven traffic loads on network performance, this method also increases power efficiency.

Here, ML provides clear advantages over static models when it comes to scalability. However, static models are shown to be less efficient as the size and complexity of interconnection networks increase, and new structures need to be modeled and optimized by hand. On the other hand, with ML-based models, you can forecast the scenarios across different topologies and do not need constant configuration assistance, thus new systems can be deployed much faster.



(a) (b)

Figure 1 – Architectures: (a) A chiplet-based hybrid interconnection network [2], (b) Proposed hybrid router [2]

Table 1 – Comparison of machine learning classifiers exploited in this research

Model	Ensemble	Regularization	Non-Linearity	Depth / Complexity
Decision Tree	×	×	✓	Shallow model with a single interpretable tree
Random Forest	✓	×	✓	Moderate depth via 100+ averaged trees
SVM (RBF Kernel)	×	×	✓ (via kernel)	Implicitly deep through kernel-based transformation
Logistic Regression (L1/L2)	×	✓ (Lasso/Ridge)	×	Shallow linear model with regularization
Neural Network (MLP)	×	✓ (Dropout / L2)	✓ (ReLU, Softmax)	Deep architecture with multiple hidden layers

Our work is different from previous research on the basis of its focus on hybrid WiNoC architectures, the next generation of integrated networks. Although previous research had focused primarily on static and hardwired NoC networks, we leveraged the power of machine learning to address the difficulties of hybrid communication and forge new directions for interconnection networks.

3. PROPOSED HYBRID INTERCONNECTION NETWORK

An active interposer combines wired and wireless communication in a chiplet-based hybrid interconnection network. To facilitate inter-chiplet communication, the basic design implements a mesh topology on the interposer. Depending on performance requirements, the wired network can handle both intra-chiplet and inter-chiplet communication, while wireless communication is only used for data transfer between chips [2].

With 16 Processing Elements (PEs) per chiplet and 16 chiplets joined by an interposer, this hybrid design has 256 PEs in total. However, Fig. 1(a) only shows four chiplets for simplicity. Each chiplet contains 16 PEs organized in a 2D mesh, with one PE interacting with a Hybrid Router (HR) to enable hybrid communication, and 15 PEs connecting to separate Baseline Routers (BRs).

As demonstrated in Fig. 1(b), the components of the wireless interface, a MAC table, and a Baseline Router (BR) make up a Hybrid Router (HR) [2]. Every HR is outfitted with a single radio transmitter that communicates in half-duplex mode. In order to accommodate both directional and omnidirectional transmission modes, this study uses a wireless connection with a reconfigurable four-element planar array antenna.

4. PROPOSED MACHINE LEARNING-BASED DYNAMIC ROUTING FRAMEWORK

In this research, we implemented five different models to predict the optimal routes for hybrid interconnection designs based on the parameters of the hybrid interconnection network. The classification task was structured

Table 2 – Feature descriptions and quantification

Feature	Description	Quantification
Packet injection rate	Frequency of packet generation at source (typically ranged between 0.0 and 1.0 packets/node/cycle)	Used three 2-bit thresholds: 0.0 to 0.3 = Low (00) 0.31 to 0.6 = Moderate (01) 0.61 to 1.0 = High (10)
Packet type	Categorical variable indicating the type of packet	Used three 2-bit values: Unicast (00) Multicast (01) Broadcast (10)
Hop distance for hybrid link	Number of intermediate nodes between the source and destination of the hybrid route (ranged from 1 to 9 hops)	Used three 2-bit thresholds: 1–3 nodes: Low (00) 4–6 nodes: Moderate (01) 7–9 nodes: High (10)
Hop distance for wired link	Number of intermediate nodes between the source and destination of the wired/interposer route (ranged from 1 to 14 hops)	Used three 2-bit thresholds: 1–5 nodes: Low (00) 6–10 nodes: Moderate (01) 11–14 nodes: High (10)
Bandwidth utilization	Percentage of channel capacity being utilized (values ranged from 0% to 100%)	Used three 2-bit thresholds: 0%–30%: Low (00) 31%–60%: Moderate (01) 61%–100%: High (10)
Router buffer occupancy	Percentage of the router input buffer currently occupied (values ranged from 0% to 100%)	Used three 2-bit thresholds: 0%–30%: Low (00) 31%–60%: Moderate (01) 61%–100%: High (10)
Link type (Label)	Optimal route chosen: wired, interposer, or hybrid	Used three 2-bit values: Wired (00) Interposer (01) Hybrid (10)

as a three-class supervised learning problem. Table 1 represents a comparative summary of the machine learning models used for the design and development of the proposed dynamic routing framework.

4.1 Dataset generation

The dataset that was used contains around 100 000 samples that represent the traffic of interconnection networks in chiplet systems. Data was generated using BookSim2 [13], a cycle-accurate Network-on-Chip (NoC) simulator that models on-chip communication architectures. BookSim was governed by values that accurately represented realistic chiplet interconnection scenarios, comprised of multicore processing elements, mesh-based topologies, and their routing paths: wired, interposer (wired with interposer), and hybrid (wired with wireless). Each feature included is carefully chosen to reflect the critical parameters that influence routing decisions. Each data sample was encoded with the features demonestratied in Table 2.

The dataset was stored in CSV format, resulting in a clean structure of 100 000 rows and 7 columns. Table 2 represents the description and quantification of each feature.

4.2 Machine learning model implementation

The classification task was structured as a supervised learning problem of three classes. Decision tree classifier, random forest classifier, support vector classifier, logistic regression (Lasso and Ridge) classifier, neural network classifier (multilayer perceptron) are selected due to their complementing capabilities in regularization, non-linearity, interpretability, and managing structured data. These models were implemented and evaluated.

4.3 Decision tree classifier

Decision trees operate by learning decision rules inferred from data features to classify input samples into output classes. The fundamental structure consists of a root node, internal decision nodes, and leaf nodes. At each node, the feature that best splits the data is chosen (based on a criterion such as the Gini impurity). Recursive partitioning continues until a stopping condition is met. Decision

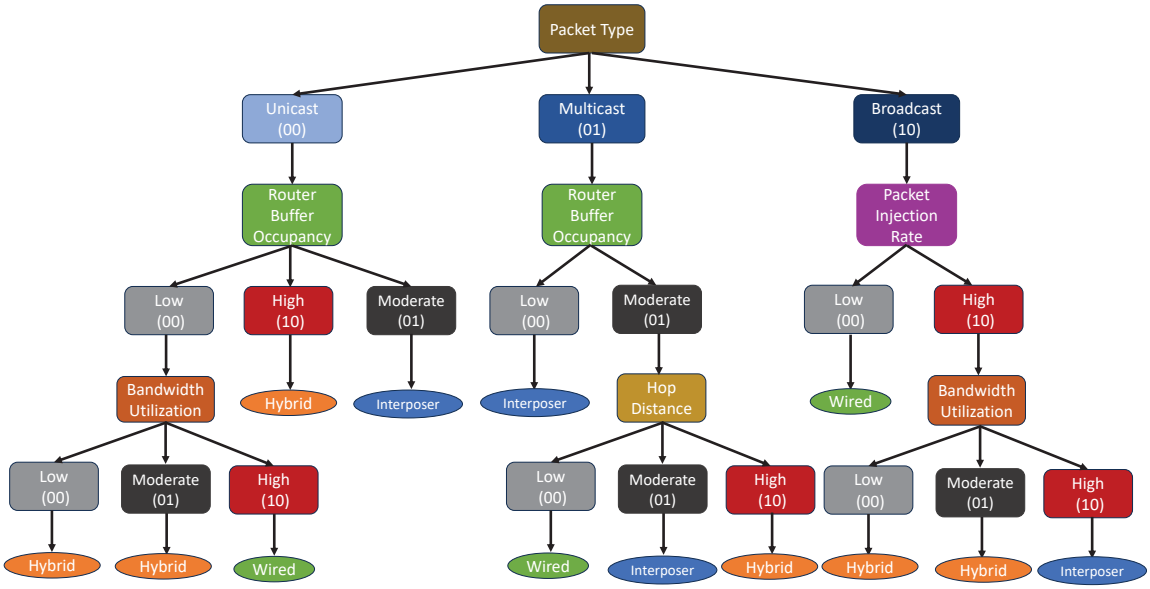


Figure 2 – Decision tree for multiclass classification of link types (wired/interposer/hybrid)

trees are widely appreciated for their interpretability, with the foundational algorithms introduced by Quinlan (ID3, C4.5) and Breiman (CART) [23, 24].

The proposed model used a maximum tree depth of 6 and used the Gini index as the splitting criterion. Experiments included using all the features of the dataset to build a full tree, restricting the depth to 3 for a shallow tree, removing the `Packet Type` feature to test its significance, and selecting only the top three important features. These variations allowed for an in-depth understanding of the feature contributions and trade-offs between depth and interpretability. Fig. 2 shows decision three with a maximum depth set to 5 to control the complexity of the model and prevent overfitting.

A comprehensive decision tree represents the raw learning process from all features. Every node divides according to a traffic attribute, such as the type of packet or the buffer level. The branching illustrates the methodical selection of routing paths. In contrast, a tree (shallow tree) was trained with a restricted depth of three to simplify the structure. This shows interpretability and captures essential logic with minimal complexity. Another version of the tree evaluates the impact of excluding the packet type. Although accuracy is slightly affected, it supports lightweight inference and feature pruning decisions. A further tree demonstrates classification performance when limited to the most important features, such as buffer occupancy and bandwidth utilization.

The tree algorithm partitions the feature space by minimizing impurity at each node. The Gini index for a node t is calculated as:

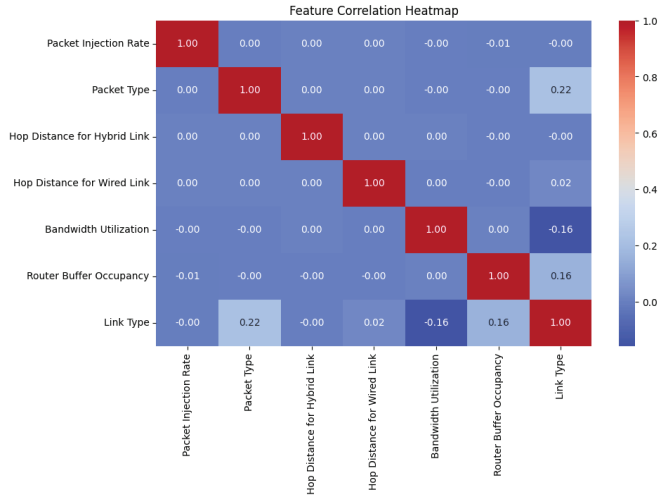


Figure 3 – Heatmap representing decision tree feature contributions

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2 \qquad (1)$$

where p_i is the fraction of samples of class i at node t. When a node is split:

$$Gini_{split} = \frac{N_{left}}{N} \cdot Gini(left) + \frac{N_{right}}{N} \cdot Gini(right) \qquad (2)$$

The algorithm selects the feature and threshold that minimize $Gini_{split}$. This selection process continues recursively until the predefined depth or leaf purity criteria are met.

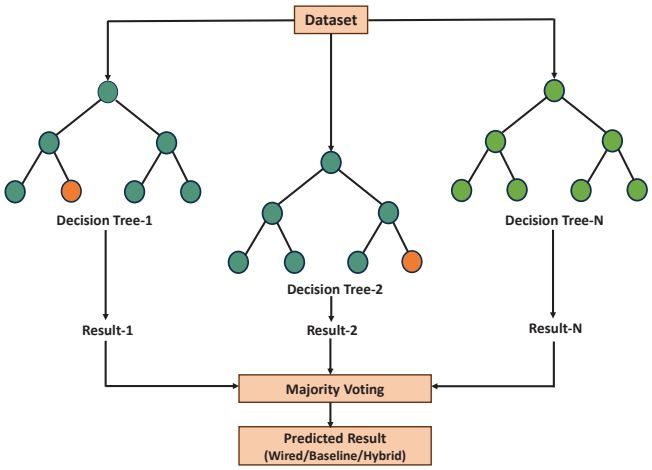


Figure 4 – Random forest for multiclass classification of link types (wired/interposer/hybrid)

The feature heat map in Fig. 3 visualizes the contribution of each input variable of the routing class decisions, highlighting real-time network dynamics.

4.4 Random forest classifier

Random forest is an ensemble learning method that builds multiple decision trees and merges their outputs to improve classification accuracy and control overfitting. It relies on bagging, where each tree is trained on a random subset of the data and features. The final classification is determined by majority vote from all trees. This method was introduced by Breiman (2001) and has been widely adopted for its robustness and interpretability [25]. Fig. 4 shows the operational diagram of a random forest classifier.

For the routing dataset, random forest was configured with 100 estimators and a maximum depth of 4. The classifier leverages multiple decision pathways, increasing the overall reliability of routing class prediction under variable network traffic patterns.

Fig. 5 highlights which traffic metrics had the greatest influence on routing predictions, validating the patterns observed in decision trees.

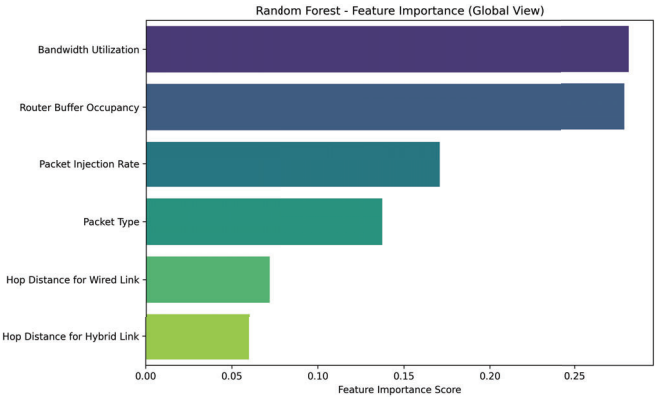


Figure 5 – Feature importance ranking from random forest classifier

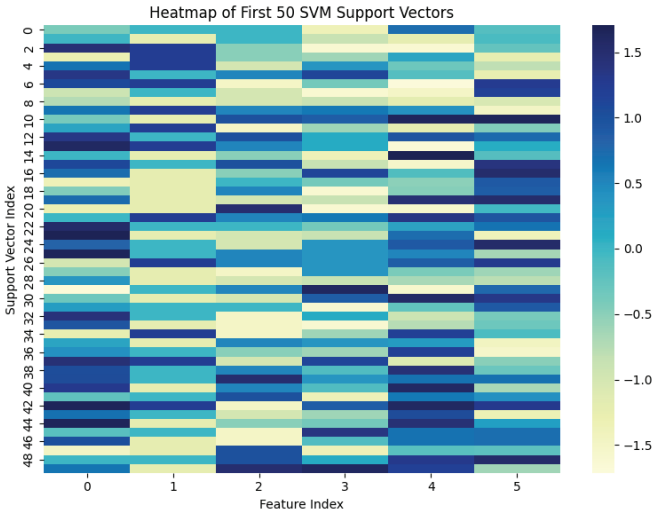


Figure 6 – Support vectors for two principal components of the routing dataset

Random forest improves generalization by aggregating predictions from multiple decision trees, reducing overfitting. According to Equation *(3)*, each node in a decision tree divides the dataset to maximize information gain. Random forest builds multiple such trees and aggregates their predictions:

$$\hat{y} = \text{mode}\left(\left\{h_j(X)\right\}_{j=1}^{M}\right) \tag{3}$$

4.5 Support Vector Classifier (SVC)

Support Vector Machines (SVMs) are supervised learning models that find an optimal hyperplane to separate classes in a high-dimensional space. SVMs attempt to maximize the margin between the closest data points of each class, known as support vectors. This margin-based approach allows for robust classification, even in non-linearly separable data when extended with kernel tricks. The method is grounded in the work of Cortes and Vapnik (1995) [26].

The SVM classifier was trained using the Scikit-learn implementation with an RBF kernel, after scaling the features using StandardScaler. The key parameters used were: (i) Kernel: Radial Basis Function (RBF), (ii) C: 1.0 (regularization parameter), and (iii) Gamma: scale (inverse of the number of features).

Given training vectors $x_i \in \mathbb{R}^n$, and labels $y_i \in \{-1, 1\}$, the SVM classifier solves:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{4}$$

Figure 7 – Heatmap of lasso coefficients across classes and features

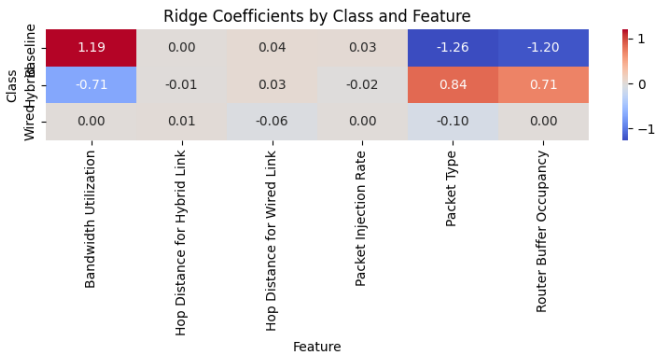


Figure 8 – Heatmap of ridge coefficients across classes and features

subject to:

$$y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{5}$$

Here, $\phi(x_i)$ is a kernel function mapping the input space to a higher-dimensional space, and ξ_i are slack variables allowing misclassification with penalty.

The use of SVM for routing classification offers strong decision boundaries in cases where class separation is subtle. For example, support vectors effectively distinguish hybrid and wired routes despite their feature overlap. The RBF kernel facilitates non-linear separation in a high-dimensional space, simulating real routing dynamics influenced by traffic burstiness and queuing behaviors.

The count and distribution of the support vector also provide interpretability, where fewer support vectors indicate clearer separation and higher confidence in classification. These insights make SVM an excellent choice for understanding marginal traffic behavior in chiplet networks. Fig. 6 contains SVM visualization showing support vectors for two principal components of the routing dataset. The larger dots represent support vectors that lie on or near the margins of the separating hyperplanes and determine the final decision boundary.

4.6 Logistic regression

Logistic regression is a statistical model that employs a logistic function to model a binary or multiclass dependent variable. It is particularly well-suited for classification problems where the outcome is discrete. For multiclass classification, the model is extended using the one-vs-rest scheme. In this research, regularized logistic regression models L1 (Lasso) and L2 (Ridge) were used to classify the routes. These models offer advantages in terms of feature interpretability and sparsity [27].

Logistic regression models were implemented using Scikit-learn with the following settings: (i) Solver: (supports both L1 and L2 penalties), (ii) regularization strength (C): 1.0 and (iii) multiclass: one-vs-rest.

Logistic regression models the probability $P(y = k \mid x)$ of a class k given input vector x as:

$$P(y = k \mid x) = \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x)} \tag{6}$$

For binary classification:

$$P(y = 1 \mid x) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-w^T x - b)} \tag{7}$$

The loss function with regularization is given as:

$$\min_w \left[-\frac{1}{n} \sum_{i=1}^{n} \log P(y_i \mid x_i) + \lambda \|w\|_p \right] \tag{8}$$

where $\|w\|_p$ is either L1 norm (Lasso) or L2 norm (Ridge).

Lasso regression eliminated some less important features, offering insight into the most influential routing

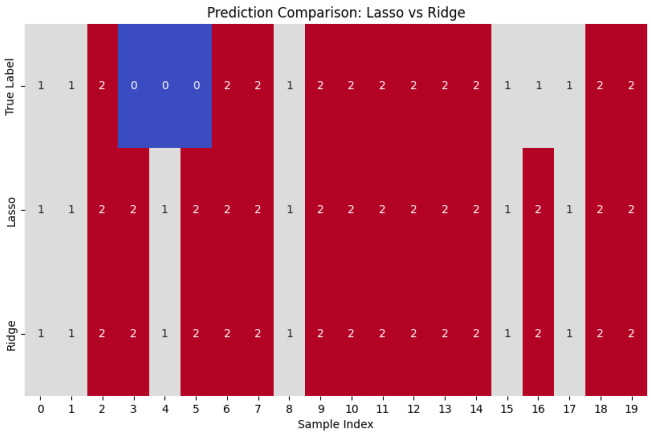


Figure 9 – Comparison of predictions from lasso and ridge logistic regression for a subset of test samples

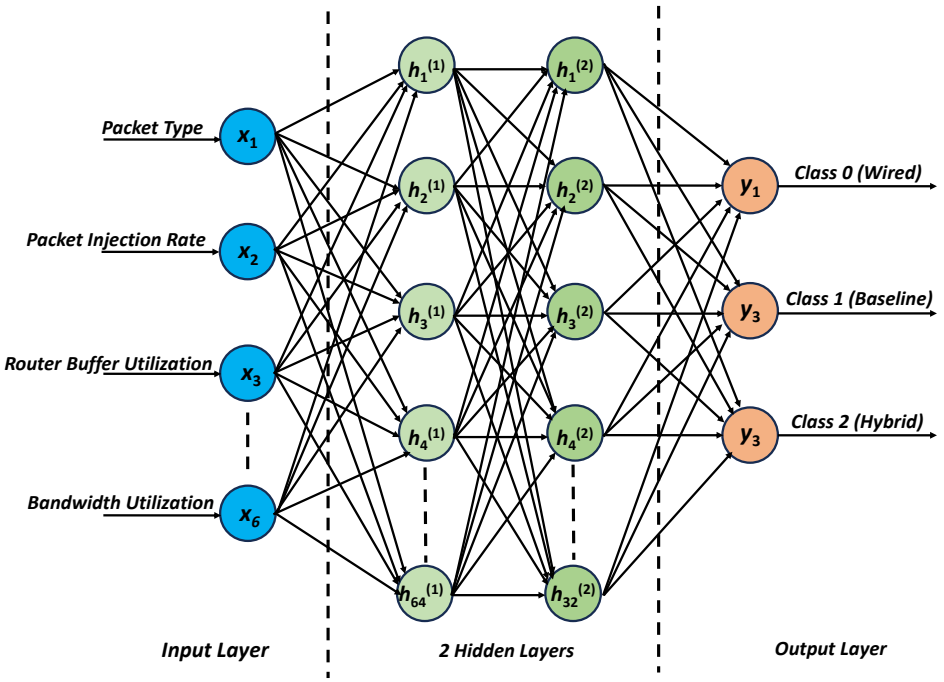


Figure 10 – Architecture of a neural network classifier with two hidden layers

characteristics. Fig. 7 shows the heatmap of lasso coefficients across classes and features. It emphasizes sparsity by pushing insignificant feature weights to zero. This visualization highlights which features were the most influential for each routing type under the L1 penalty.

In contrast, Ridge regression preserved all features, offering a more stable but less sparse solution. The comparative heatmap clarified that Ridge captured general patterns while Lasso emphasized discriminative features, especially in distinguishing hybrid routing. The heatmap of the Ridge coefficients across classes and features is shown in Fig. 8. This shows how each feature contributes to class separation without enforcing sparsity, resulting in more distributed weight magnitudes. These models are lightweight and interpretable, making them suitable for systems with limited computational resources where model transparency is essential. The heatmap in Fig. 9 reveals where the models agree or diverge in their predictions, offering insight into robustness and feature sensitivity.

4.7 Neural network classifier (multilayer perceptron)

Neural Networks (NNs) are computational models inspired by the interconnected neuron structures of the human brain. Each neuron receives input, processes them using weights and activation functions, and passes the output to the next layer. Feedforward Neural Networks (FNNs), the most common type, consist of an input layer, one or more hidden layers, and an output layer. They are effective in learning non-linear relationships between features and target labels [28].

In this research, we implemented a neural network (NN) with two hidden layers. The input layer has six neurons followed by two hidden layers of 64 and 32 neurons, respectively (as shown in Fig. 10). This structure was selected to enhance the model's ability to capture complex interactions across features in chiplet-based hybrid interconnection network data.

The output layer used a softmax activation function for multiclass classification, while the hidden layers used ReLU activation. The Adam optimizer was used to optimize the model after it was trained using the categorical cross-entropy loss function. A batch size of 128 was used for training in 50 epochs. Input features were standardized prior to training to ensure uniformity and stability of convergence.

Given input vector $\mathbf{x}$, weights $\mathbf{W}$, biases $\mathbf{b}$, and activation function f, the output of a layer is:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{9}$$

The final output probabilities for classes are computed using the softmax function:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{10}$$

Loss is calculated using the categorical cross-entropy function:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik}) \tag{11}$$

where $\hat{y}_{ik}$ is the predicted probability and y_{ik} is the true label indicator.

The neural network (NN) model ensures superior performance among all classifiers. The layered architecture allowed for capturing non-linear patterns and intricate feature interactions within chiplet network routing data. The additional hidden layers enabled the model to learn hierarchical representations, resulting in improved overall accuracy and macro-F1-score.

Moreover, the visual representation of the network structure helps in model transparency. For example, node sizing and weight annotations illustrate relative feature influence, which improves interpretability. The strong classification metrics and high adaptability to unseen patterns make this neural network especially well-suited for dynamic routing classification tasks in chiplet-based hybrid interconnection networks.

4.8 Training setup and preprocessing

The target variable link type is a categorical variable that indicates the optimal route for each packet. To evaluate the generalization performance of the model, the dataset was divided into training sets (80%) and testing sets (20%). Stratified sampling was used to maintain the proportional representation of all routing classes in both training and test sets, preventing potential bias due to class imbalance. All input features with continuous values are normalized using the StandardScaler transformation prior to training. This scaling process ensured that the features contributed equally to distance-based model computations and stabilized gradient-based learning for neural networks.

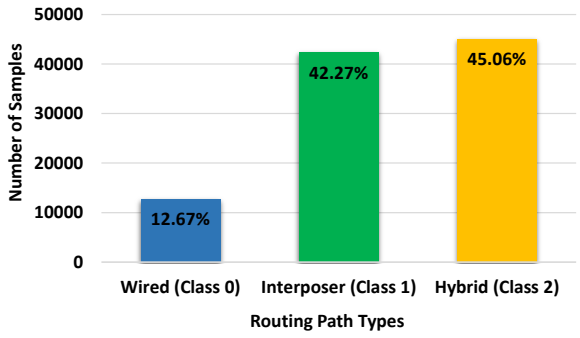


Figure 11 – Statistics of class distribution of dataset

4.9 Class distribution statistics

Here, the dataset has 100 000 samples to represent different routing decisions within a hybrid interconnection network. Fig. 11 graphically illustrates the comparison of how many samples have been associated with three types of routing path: wired (Class 0), interposer (Class 1), and hybrid (Class 2). The class distribution statistics are as follows:

- **Wired routing (Class 0):** 12,673 samples (12.67%).
- **Interposer routing (Class 1):** 42,271 samples (42.27%).
- **Hybrid routing (Class 2):** 45,056 samples (45.06%).

4.10 Model training

All machine learning classifier models are trained offline. Through this process, all of the training data is pre-dispatched before the learning takes place and is trained in batches or epochs with no real-time updates. It is a standard procedure in cases where computational resources may be scheduled in advance and the data partitioning is constant. After training, the performance of the classifier is tested on a separate test set, and upon verification it is applied to inference for novel, unseen data. Applications where real-time data is not required or retraining can be performed intermittently instead of constantly are appropriate for offline training.

5. SIMULATION AND PERFORMANCE EVALUATION

5.1 Simulation environment

The evaluation of the proposed ML-based routing framework is carried out using a customized version of the cycle-accurate network simulator BookSim2 [13]. The framework combines flow c ontrol, a M AC p olicy, an intelligent machine learning–driven routing approach, and a modified hybrid router architecture that integrates pretrained ML models. These models are trained offline

Table 3 – Simulation parameters

Simulation Parameters	Values
Network Topology and Size	2D Mesh, 16×16
Chiplet Size, Number	4×4 PEs, 16
Number of Hybrid Router	16
Buffer Size of Router	4 flits/VC
Virtual Channels (VCs)	16 (4VCs/port)
Tx/Rx Buffer Size of Antenna	16
Flit Size	128 bits
Packet Size	4 flits
Maximum Wireless Data Rate	7 Gbps/Ch
Number of Wireless Channel	2

Table 4 – Performance metrics of all classifiers (macro-averaged)

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	81.7%	79.4%	80.2%	80.1%
Random Forest	86.2%	85.4%	85.8%	85.7%
SVM	84.9%	83.5%	84.4%	84.1%
Logistic Regression	82.3%	81.0%	81.6%	81.8%
Neural Network	**88.6%**	**87.9%**	**88.1%**	**88.2%**

and are deployed sequentially across routers during testing. Furthermore, real-world application traces from the PARSEC [29] benchmark suite are incorporated into the simulation using Netrace [30]. Table 3 contains the parameters of our simulation designs.

We compare the performance of the proposed ML-based routing designs (Hybrid+DTC, Hybrid+RFC, Hybrid+SVC, Hybrid+LRC and Hybrid+NNC) with the traditional wired baseline NoC design and the chiplet-based hybrid design [2]. The performance metrics considered are end-to-end latency, throughput, and energy efficiency. All performance metrics here are presented in a normalized form with respect to the baseline wired NoC.

5.2 Machine learning model performance comparison

The supervised machine learning models trained for routing categorization in hybrid chiplet-based interconnection networks are thoroughly evaluated. The performance of these models were evaluated in the unseen testing set using the following metrics, where TP, FP, TN, and FN denote true positives, false positives, true negatives, and false negatives, respectively.

- **Accuracy:** Total accuracy of forecasts for every class.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

- **Precision:** Prediction quality is indicated by the percentage of genuine positive predictions of all expected positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{13}$$

- **Recall:** The sensitivity of the model is its capacity to detect all pertinent events.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{14}$$

- **F1-score:** Precision and recall are balanced into a single number by taking the harmonic mean of the two measures.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

- **Macro-averaging:** Used to assign each class the same weight, notwithstanding sample imbalance.

$$\text{Macro-Averaged Score} = \frac{1}{C} \sum_{i=1}^{C} \text{Score}_i \tag{16}$$

where C is the number of classes and Score_i refers to the Precision, Recall, or F1-score for class i.

The macro-averaged metrics derived from each of the five models in the test dataset are compiled in Table 4. Among all the models evaluated, the neural network achieved the highest accuracy and F1-score.

5.2.1 Justification on model selection

On evaluating several machine learning models, including decision tree, random forest, Support Vector Machine (SVM), logistic regression (L1 and L2 regularized) and neural networks, it was found that the neural network consistently outperformed all other models in key metrics such as accuracy (88.6%), F1-score (88.2%), precision (87.9%) and recall (88.1%) (as shown in Table 4). According to these criteria, the neural network outperformed conventional models in its ability to identify intricate nonlinear patterns in the dataset.

On the other hand, despite being interpretable and computationally fast, models such as decision tree and logistic regression showed somewhat poorer generalization performance. As an ensemble approach, random forest trailed the neural network by a small margin but produced competitive results with strong feature importance analysis.

The neural network was chosen as the final model for deployment and additional testing based on this evaluation. Particularly in multiclass classification settings, when other models tend to oversimplify the decision boundaries, its higher performance demonstrates its capacity to learn intricate feature interactions.

5.3 Deployment feasibility for edge devices

The proposed framework is highly promising for deployment in edge devices, especially if lightweight models

such as decision tree or logistic regression are considered that are memory-frugal and computationally inexpensive at inference time. The runtime overhead is minimal as the training is offline. Although neural networks are more precise, their use in edge devices might involve the use of other optimizations like pruning or quantization, or the use of dedicated hardware accelerators. With appropriate model selection and deployment mechanisms, the approach is perfectly compatible with real-time edge deployments.

5.4 Performance evaluation

The performance of the proposed dynamic routing framework is assessed through simulation results, based on the following metrics:

- **End-to-end latency:** The total time it takes for a packet to travel from the source node (injection point) to the destination node (ejection point) within the simulated network (measured in cycle).
- **Throughput:** The average number of flits successfully delivered across the network per cycle per node (measured in flits/cycle/node).
- **Energy efficiency:** The total power consumption consists of static and dynamic power usage. The total energy consumption of a benchmark application is calculated by multiplying its final execution time by the total power used during that application's execution. The following formula is used to estimate energy efficiency [2]:

$$E_{efficiency} = \frac{1}{T_{executin} \times \left(P_{static} + P_{dynamic}\right)} \qquad (17)$$

5.4.1 End-to-end latency

Fig. 12 shows the normalized end-to-end packet latency comparison for a range of application benchmarks and three designs: a traditional wired baseline NoC, a basic hybrid interconnection design with human-engineered routing, and the basic hybrid design enhanced with machine learning classifiers for optimized routing. The baseline wired design consistently shows the highest latency in most workloads. The basic hybrid chiplet-based design with human-engineered adaptive routing achieves 46% reduced end-to-end latency than the wired baseline NoC. Since models with a small computation overhead like decision trees (Hybrid+DTC) and logistic regression (Hybrid+LRC) have minimal computing overhead and inference speed, they achieve the lowest end-to-end latency (reduced 60% and 64%, respectively, than the baseline wired design). More complex models, like neural networks (Hybrid+NNC) and ensemble methods like random forests (Hybrid+RFC), allow for more

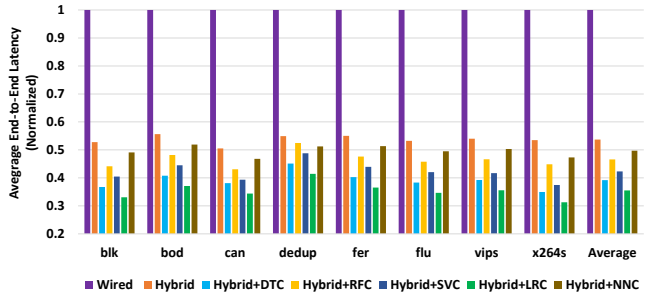


Figure 12 – Comparison of average end-to-end latency (normalized)

accurate predictions but suffer from increased end-to-end latency due to larger parameter spaces, non-linear functions, and multipath execution. Even Hybrid+RFC and Hybrid+NNC achieved 53% and 50% reduced end-to-end latency compared to the baseline wired design. The hybrid design with neural network classifier-based routing (Hybrid+NNC) induces almost the same end-to-end latency (50%) as the hybrid design with human-engineered routing due to higher inference time for more computations, especially with non-linear activations and more parameters. For smaller datasets with fewer support vectors, the support vector classifier (Hybrid+SVC) works well, but as the number of support vectors increases, the latency increases noticeably.

5.4.2 Throughput

The graph in Fig. 13 uses a variety of routing schemes, including a traditional wired baseline NoC, a basic hybrid interconnection design with human-engineered routing, and the basic hybrid design enhanced with machine learning classifiers for optimized routing, to evaluate network throughput in various application benchmarks. These results demonstrate the superior performance of advanced models such as neural networks and ensemble methods in optimizing throughput for NoC systems. Among these, Hybrid+NNC (with a neural network classifier) consistently has the highest average throughput (2.28 × improved throughput compared to wired baseline NoC), leveraging its ability to learn advanced, non-linear traffic patterns to provide adaptive and efficient routing. Hybrid+RFC with a random forest classifier also ranks well, with 2.10 × improved throughput and robustness

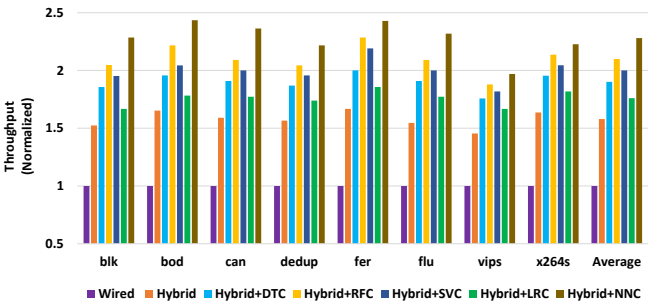


Figure 13 – Comparison of network throughput (normalized)

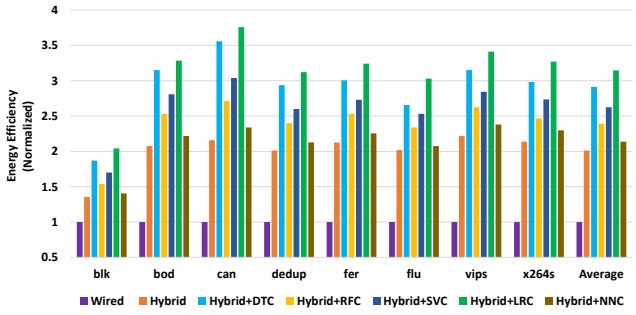


Figure 14 – Comparison of energy efficiency (normalized)

against traffic variation compared to the wired baseline NoC. Hybrid+SVC (SVM-based) achieves moderate gains (2.00 × improved throughput compared to baseline wired NoC), limited by scalability and adaptability. Decision trees are sensitive to noise and overfitting, which is reflected in the unstable and inconsistent performance of the Hybrid+DTC technique (with a decision tree classifier). Finally, Hybrid+LRC (logistic regression classifier) is effective with linearly separable data, but does not generalize to a complex, non-linear NoC traffic landscape, offering minimal (2.00 × improved) throughput improvement compared to a baseline wired NoC. These results indicate the high performance of sophisticated models such as neural networks and ensemble methods to maximize throughput for such interconnection design. In addition, the basic hybrid design achieves a substantial improvement in throughput (1.58 × improved) compared to the wired baseline design. However, the basic hybrid design achieves lower throughput than when enhanced with machine learning classifiers for optimized routing.

5.4.3 *Energy efficiency*

Fig. 14 shows a comparison of energy efficiency for different application benchmarks using multiple routing strategies: a traditional wired baseline NoC, a basic hybrid interconnection design with human-engineered routing, and the basic hybrid design enhanced with machine learning classifiers for optimized routing. The energy efficiency of the proposed interconnection designs is closely tied to the trade-off between computational complexity and performance gains in latency and throughput. Lightweight models such as decision trees (Hybrid+DTC) and logistic regression (Hybrid+LRC) are the most energy efficient (achieve an improvement of 2.91 × and 3.14 × in energy efficiency compared to baseline wired NoC) because they significantly reduce end-to-end latency with a minimal inference overhead. These models are ideal for resource-constrained environments where low power consumption is critical. On the other hand, more complex models like neural networks (Hybrid+NNC) and ensemble methods (Hybrid+RFC) are higher in throughput in terms of non-linear traffic pattern capture, but at increased computational expense in terms of increased energy consumption (achieve an improvement of 2.14 × and 2.39 × in energy efficiency compared to baseline wired NoC). The support vector classifier (Hybrid+SVC) is a moderately energy-efficient model with adequate performance for small workloads but reduces efficiency with increasing support vectors. In general, hybrid models with simpler classifiers strike the best balance between performance and energy consumption, whereas models with higher throughput often incur higher energy costs. The basic hybrid design is more energy efficient than the traditional wired baseline due to its lightweight routing design and achieves an improvement of 2.01 × in energy efficiency. However, the lack of adaptive routing decisions for heterogeneous workload introduces packet drop or iterative retransmission. Thus, the basic hybrid design can cause dynamic power consumption and be less energy efficient than the hybrid designs based on ML (Hybrid+LRC achieves a maximum 56% and Hybrid+NNC achieves a minimum 6% improvement in energy efficiency compared to the basic hybrid design). The traditional wired baseline is simpler and potentially lower in power per operation. However, the wired baseline NoC design is less energy efficient than hybrid designs because more time and cycles are needed to complete the same tasks, resulting in a higher energy cost.

6. CONCLUSION

This work presents a custom machine learning-based dynamic routing framework for chiplet-based hybrid interconnect networks that addresses the limitations of traditional static routing schemes. This research shows that neural networks offer higher classification performance, particularly for interposer and hybrid routes, by using real-time traffic data and evaluating multiple classifiers within a cycle-accurate simulation environment. In addition to increasing routing adaptability, the proposed framework significantly improves system performance: the simulation results show a maximum reduction of 64% in end-to-end latency, a 2.28 × improvement in throughput and a 3.14 × gain in energy efficiency compared to the baseline wired NoC. In addition, it outperforms basic hybrid interconnection designs with human-engineered routing, achieving 33% latency reduction, 1.44 × throughput improvement and 1.56 × energy efficiency enhancement. Furthermore, the lightweight and scalable nature of the framework positions it well for practical integration into next-generation chiplet-based architectures. This research establishes the foundation for future exploration in adaptive interconnect strategies, including reinforcement learning, fault-tolerant routing, and hardware-aware co-design.

REFERENCES

[1] Xiaohang Wang, Yifan Wang, Yingtao Jiang, Amit Kumar Singh, and Mei Yang. "On Task Mapping in Multi-chiplet Based Many-Core Systems to Optimize Inter- and Intra-chiplet Communications". In: *IEEE Transactions on Computers* 74.2 (2025), pp. 510–525. DOI: 10.1109/TC.2024.3500354.

[2] Md Tareq Mahmud and Ke Wang. "A Flexible Hybrid Interconnection Design for High-Performance and Energy-Efficient Chiplet-Based Systems". In: *IEEE Computer Architecture Letters* 23.2 (2024), pp. 215–218. DOI: 10.1109/LCA.2024.3477253.

[3] Md Tareq Mahmud and Ke Wang. "A Chiplet-Based High-Performance and Secure Hybrid Interconnection Network Design Against DoS and Sniffing Attacks". In: *SoutheastCon 2025*. 2025, pp. 874–879. DOI: 10.1109/SoutheastCon56624.2025.10971688.

[4] Ayodeji Ireti Fasiku, Benson Olabode Ojedayo, and Oghenerukevwe Elohor Oyinloye. "Effect of Routing Algorithm on Wireless Network-on-Chip Performance". In: *2020 Second International Sustainability and Resilience Conference: Technology and Innovation in Building Designs(51154)*. 2020, pp. 1–5. DOI: 10.1109/IEEECONF51154.2020.9319964.

[5] Khurshid Ahmad and Muhammad Athar Javed Sethi. "Review of Network on Chip Routing Algorithms". In: *EAI Endorsed Transactions on Context-aware Systems and Applications* 7.22 (Dec. 2020), e5. DOI: 10.4108/eai.23-12-2020.167793. URL: https://eudl.eu/doi/10.4108/eai.23-12-2020.167793.

[6] J. Fang, Z. Wei, Y. Liu, et al. "TB-TBP: A Task-Based Adaptive Routing Algorithm for Network-on-Chip in Heterogeneous CPU-GPU Architectures". In: *Journal of Supercomputing* 80 (2024), pp. 6311–6335. DOI: 10.1007/s11227-023-05700-7. URL: https://doi.org/10.1007/s11227-023-05700-7.

[7] Biplab Thapa Magar, Subin Mali, and Eman Abdelfattah. "App Success Classification Using Machine Learning Models". In: *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. 2021, pp. 0642–0647. DOI: 10.1109/CCWC51732.2021.9376021.

[8] Iqbal H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". In: *SN Computer Science* 2 (Mar. 2021). DOI: 10.1007/s42979-021-00592-x.

[9] Md. Tareq Mahmud, Hasan Mahmud, Md Habibullah Belali, Md. Obaidur Rahman, and Ke Wang. "Machine Learning-based Rainfall Prediction from Weather Data: A Comparative Analysis". In: *2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*. 2023, pp. 1–6. DOI: 10.1109/NCIM59001.2023.10212732.

[10] Xiaoyun Zhang, Dezun Dong, Cunlu Li, Shaocong Wang, and Liquan Xiao. "A survey of machine learning for Network-on-Chips". In: *Journal of Parallel and Distributed Computing* 186 (2024), p. 104778. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2023.104778. URL: https://www.sciencedirect.com/science/article/pii/S074373152300148X.

[11] Md Farhadur Reza. "Machine Learning Enabled Solutions for Design and Optimization Challenges in Networks-on-Chip based Multi/Many-Core Architectures". In: *J. Emerg. Technol. Comput. Syst.* 19.3 (June 2023). ISSN: 1550-4832. DOI: 10.1145/3591470. URL: https://doi.org/10.1145/3591470.

[12] Samala Jagadheesh, P. Veda Bhanu, J. Soumya, and Linga Reddy Cenkeramaddi. "Reinforcement Learning Based Fault-Tolerant Routing Algorithm for Mesh Based NoC and Its FPGA Implementation". In: *IEEE Access* 10 (2022), pp. 44724–44737. DOI: 10.1109/ACCESS.2022.3168992.

[13] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, D. E. Shaw, John Kim, and William J. Dally. "A detailed and flexible cycle-accurate Network-on-Chip simulator". In: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2013, pp. 86–96. DOI: 10.1109/ISPASS.2013.6557149.

[14] Tobias Bjerregaard and Shankar Mahadevan. "A survey of research and practices of Network-on-chip". In: *ACM Comput. Surv.* 38.1 (June 2006), 1–es. ISSN: 0360-0300. DOI: 10.1145/1132952.1132953. URL: https://doi.org/10.1145/1132952.1132953.

[15] Farshid Yazdanpanah, Reza Afshar Mazayejani, Mohammad Alaei, et al. "An energy-efficient partition-based XYZ-planar routing algorithm for a wireless network-on-chip". In: *The Journal of Supercomputing* 75 (2019), pp. 837–861. DOI: 10.1007/s11227-018-2617-x. URL: https://doi.org/10.1007/s11227-018-2617-x.

[16] Y. Asadi. "A comprehensive study and holistic review of empowering network-on-chip application mapping through machine learning techniques". In: *Discover Electronics* 1 (2024), p. 22. DOI: 10.1007/s44291-024-00027-w. URL: https://doi.org/10.1007/s44291-024-00027-w.

[17] Ramapati Patra, Prasenjit Maji, Yogesh Raj, and Hemanta Kumar Mondal. "Enhancing predictive accuracy using machine learning for network-on-chip performance modeling". In: *Comput. Electr. Eng.* 123 (2025), p. 110041. URL: https://api.semanticscholar.org/CorpusID:275362008.

[18] F. Al-Obaidy and F. A. Mohammadi. "Predictions optimal routing algorithm based on artificial intelligence technique for 3D NoC systems". In: *Microsystem Technologies* 27 (2021), pp. 3313–3323. DOI: 10.1007/s00542-020-05084-1. URL: https://doi.org/10.1007/s00542-020-05084-1.

[19] Jagadheesh Samala, Veda Bhanu, Joshi Soumya, and Linga Reddy Cenkeramaddi. "Reinforcement Learning Based Fault-Tolerant Routing Algorithm for Mesh Based NoC and Its FPGA Implementation". In: *IEEE Access* 10 (Jan. 2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3168992.

[20] Ke Wang and Ahmed Louri. "CURE: A High-Performance, Low-Power, and Reliable Network-on-Chip Design Using Reinforcement Learning". In: *IEEE Transactions on Parallel and Distributed Systems* PP (Apr. 2020), pp. 1–1. DOI: 10.1109/TPDS.2020.2986297.

[21] R. Patra, P. Maji, D. S. Srivastava, et al. "Machine learning-driven performance assessment of network-on-chip architectures". In: *Journal of Supercomputing* 80 (2024), pp. 24483–24519. DOI: 10.1007/s11227-024-06340-1. URL: https://doi.org/10.1007/s11227-024-06340-1.

[22] K Balamurugan, S Umamaheswaran, Tadele Mamo, S Nagarajan, and Lakshmana Rao Namamula. "Roadmap for machine learning based network-on-chip (M/L NoC) technology and its analysis for researchers". In: *Journal of Physics Communications* 6.2 (Feb. 2022), p. 022001. DOI: 10.1088/2399-6528/ac4dd5. URL: https://dx.doi.org/10.1088/2399-6528/ac4dd5.

[23] J. R. Quinlan. "Induction of Decision Trees". In: *Mach. Learn.* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125. DOI: 10.1023/A:1022643204877. URL: https://doi.org/10.1023/A:1022643204877.

[24] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees (1st ed.)* Chapman and Hall/CRC, 1984. DOI: 10.1201/9781315139470. URL: https://doi.org/10.1201/9781315139470.

[25] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[26] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

[27] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, 2013. URL: https://www.wiley.com/en-us/Applied+Logistic+Regression%2C+3rd+Edition-p-9780470582473.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/.

[29] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. "The PARSEC benchmark suite: characterization and architectural implications". In: *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. PACT '08. Toronto, Ontario, Canada: Association for Computing Machinery, 2008, pp. 72–81. ISBN: 9781605582825. DOI: 10.1145/1454115.1454128. URL: https://doi.org/10.1145/1454115.1454128.

[30] Thiem Van Chu, Kenji Kise, and Kiyofumi Tanaka. "Dependency-Driven Trace-Based Network-on-Chip Emulation on FPGAs". In: *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '20. Seaside, CA, USA: Association for Computing Machinery, 2020, pp. 211–221. ISBN: 9781450370998. DOI: 10.1145/3373087.3375309. URL: https://doi.org/10.1145/3373087.3375309.

AUTHORS

Md Tareq Mahmud (Graduate Student Member, IEEE) received B.Sc. and M.Sc. degrees in computer science and engineering from Dhaka University of Engineering and Technology, Gazipur, Bangladesh, in 2015 and 2019, respectively. He is an Assistant Professor at the Institute of Information & Communication Technology (IICT), Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. He is currently pursuing a Ph.D. degree in electrical engineering from the Electrical and Computer Engineering department at the University of North Carolina at Charlotte, USA. His research interests include AI/ML and data-driven on-chip interconnection networks, reliable and high-performance computer architecture, machine learning and wireless communication.

Uma Maheswar Reddy Nagireddi received a B.Tech. degree from the School of Electronics and communication Engineering, Vellore Institute of Technology (VIT), Vellore, India, in 2020. He is pursuing an M.S. degree in computer engineering from the University of North Carolina at Charlotte, USA. His research interests include data-driven routing in hybrid chiplet interconnections and machine learning approaches for chip-level communication optimization.

Ke Wang (Member, IEEE) received a Ph.D. degree in computer engineering from the George Washington University in 2022. He received M.S. degree in electrical engineering from Worcester Polytechnic Institute in 2015, and B.S. degree in electrical engineering from Peking University in 2013. He is currently Assistant Professor of Electrical and Computer Engineering at the University of North Carolina at Charlotte. His research work focuses on parallel computing, computer architecture, interconnection networks, and machine learning.