

Power, Area and Thermal Prediction in 3D Network-on-Chip using Machine Learning

Abhijith C, Anand M K

Department of Computer Science and Engineering

National Institute of Technology Karnataka (NITK)

Surathkal, India

Email: {abhijithc.242cs003, anandmk.242cs008}@nitk.edu.in

I. PROPOSED METHODOLOGY

Machine learning (ML) algorithms are widely used in various real-time predictions. The Power, Area, and Temperature (PAT) prediction of Network-on-Chip can also leverage the ability of machine learning models. Most existing studies focus on power, area, or thermal analysis independently. Frameworks involving simultaneous analysis of all three parameters are rare in current research. This work proposes a hybrid model that combines ML and DL algorithms. The ML component of the hybrid model uses algorithms such as linear regression and decision trees to predict area and power, which have a linear relationship with NoC parameters. The DL component of the model uses convolutional neural networks (CNNs) that learn complex non-linear relationships in NoC to predict temperature. The algorithms can predict PAT values for unseen input NoC configuration based on their learning. The prediction of both models is combined to produce a single output in the hybrid model.

A. Dataset Creation

The dataset generation process for power, area, and thermal prediction in NoC systems involves using different parameters specified in Table I to generate different configurations for a NoC system. The different NoC parameters include topology, routing algorithm, network size, traffic pattern, buffer size, packet size, and packet injection rates (PIR). The dataset is created by changing the value of the parameters to generate different configurations for an NOC system.

The dataset creation is described in Algorithms 1 and 2. Step 1 involves generating different configurations of a NoC system by looping through different parameter values. Step 2 involves using a PAT-maxim simulator to simulate the different configurations generated in Step 1 and collecting output metrics, such as Average Power, Total Area, Average Temperature, etc., for each configuration. Step 3 involves storing the output metrics along with the configurations of the NoC system into a file such as a CSV file. This file will then be used to train the ML and DL models.

NoC Parameter	Parameter Values
Topology	3D Mesh Topology
Routing Algorithm	XYZ Routing, OE_3D (Odd-Even 3D Routing), Full Adaptive Routing
Network Size	dimX: 2 to 16, dimY: 2 to 16, dimZ: 2
Traffic Pattern	Random
Buffer Size	4
Packet Size	4
Packet Injection Rate (PIR)	0.01, 0.02, 0.03, ..., 0.1
Sample Period	200000

TABLE I: NoC Parameters and Values

Algorithm 1 Generate NoC Configurations

```
1: Input: Set of NoC parameters:
   • Topology: 3D Mesh Topology
   • Routing Algorithms:
     – XYZ Routing
     – OE_3D (Odd-Even 3D Routing)
     – Full Adaptive Routing
   • Network Size:
     – dimX: 2 to 16
     – dimY: 2 to 16
     – dimZ: 2
   • Traffic Pattern: Random
   • Buffer Size: 4
   • Packet Size: 4
   • Packet Injection Rate (PIR):
     – 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1
   • Sample Period: 200,000 cycles
2: Step 1: Generate Configurations
3: for dimX from 2 to 16 do
4:   for dimY from 2 to 16 do
5:     for each routing algorithm in Routing Algorithms do
6:       for each PIR in Packet Injection Rate do
7:         Create a configuration with parameters:
         {topology, dimX, dimY, dimZ, traffic pattern, PIR, buffer
         size, packet size, sample period, routing algorithm}
8:         Store the configuration in a list
9:       end for
10:    end for
11:  end for
12: end for
```

Algorithm 2 Simulate Configurations and Save Dataset (Continued)

- 1: **Step 2: Simulate and Gather Metrics**
 - 2: **for** each configuration in the list **do**
 - 3: Run the Noxim simulation with the current configuration
 - 4: Capture metrics:
 - Total Area (um²)
 - Avg Power (J/cycle)
 - Avg Cores Power (J/cycle)
 - Avg Routers Power (J/cycle)
 - Avg Power per Router (J/cycle)
 - Layer Area (um²)
 - Area per Core (um²)
 - Steady state temperature - L0
 - Steady state temperature - L1
 - Router Average Temperature - L0
 - Router Average Temperature - L1
 - Core Average Temperature - L0
 - Core Average Temperature - L1
 - Memory Average Temperature - L0
 - Memory Average Temperature - L1
 - 5: **end for**
 - 6: **Step 3: Save Dataset**
 - 7: Save the dataset with all configurations and metrics to a file (e.g., CSV)
-

B. Linear Regression

Linear regression is an ML algorithm that is used for predictions involving numerics. It learns the relationship between dependent (output) and independent (input) variables by fitting a linear equation. The equation is represented as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$

where Y is the predicted output, X_1, X_2, \dots, X_n are the input, β_0 is the intercept, $\beta_1, \beta_2, \dots, \beta_n$ are the coefficients, and ϵ is the error term. The coefficients of the equation are learned using Mean Squared Error (MSE), a loss function. In power, area, and thermal (PAT) prediction for networks-on-chip (NoC), linear regression can predict these metrics based on input parameters such as topology size, packet injection rate, etc. However, linear regression learns only the input-output linear relationships. Therefore, methods such as decision trees are needed to learn the non-linear and complex relationships present in NoC systems.

C. Decision Trees

Decision trees are powerful ML algorithms used for regression and classification tasks. These algorithms split the training dataset based on the most significant feature that separates the data most precisely. The internal nodes of the decision tree define the decision on an input feature, and the leaf node represents the predicted output value. Decision trees

can find complex input-output relationships as they are non-parametric. In power, area, and thermal prediction in NoC systems, complex relationships between NoC parameters can be learned using these algorithms. Decision trees help researchers understand which features impact the outputs the most by providing a visual structure. However, these algorithms can result in an overfitted model with increased generalization error. Generalization error can be reduced using techniques like pruning, making these algorithms perform well in large-scale NoC systems.

D. Convolutional Neural Networks (CNNs)

CNNs are used for tasks such as Image processing and Object detection. They apply filters using convolutional layers to extract features from the input dataset, making CNN the best algorithm for learning data structured in grid-like topologies, such as NoC systems. In power, area, and thermal prediction in the NoC system, CNNs can be used to understand non-linear relationships required for temperature prediction. CNNs can learn complex patterns in NoC designs that may not be possible for simpler algorithms like linear regression and decision trees. They work well with large datasets.

E. Design of Proposed Method

Algorithm 3 describes the flow of the proposed method. Step One involves dataset generation using the PAT-Noxim simulator, as described in Algorithms 1 and 2. The dataset is normalized to ensure that all the parameters equally contribute to the ML prediction. The duplicate data is removed to ensure the proposed model is trained on different data. The dataset is split into training (70%), testing (15%), and validation (15%).

The proposed hybrid model includes ML components for Power and Area prediction and DL components for Temperature prediction. The training of the linear regression model minimizes the MSE between the predicted and actual values of power and area.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Where:

- Y_i : Actual output
- \hat{Y}_i : Predicted output
- n : Number of samples

The ability of the model to predict power and area on unseen NoC configurations are evaluated using the MSE. The dataset is also trained on a decision tree algorithm to predict the power and area. This training phase involves splitting the dataset based on the most significant feature that separates the data most precisely. The MSE is used to calculate the ability of the model to predict accurately.

The CNN model is designed to train the dataset for temperature prediction. The architecture consists of the following layers:

- **Input Layer:** Accepts the input parameters.

- **Convolutional Layer 1:** Applies a set of convolutional operations to extract features from input. ReLU is used as an activation function.
 - ReLU returns the maximum of 0, and the input value introduces non-linearity in the model. It can be represented as:

$$f(x) = \max(0, x)$$

- **Max-Pooling Layer:** Reduces the size of the feature map without losing extracted information.
- **Convolutional Layer 2:** Extracts more features from the pooled feature map.
- **Flattening Layer:** Converts the feature map to a single-dimensional vector.
- **Fully Connected Layer:** Aggregates the learned features.
- **Output Layer:** Outputs the predicted thermal value with a linear activation function. The linear function output is the same as the input. It can be represented as:

$$f(x) = x$$

The predicted power area and temperature are combined, and the model is tested using a test dataset. The flowchart of the proposed method is given in Figure 1.

Algorithm 3 Algorithm for Proposed Framework

- 1: **Input:** Set of NoC parameters: Topology, Traffic Pattern, PIR (X, Y, Z), Buffer Size, Packet Size, Sample Period
 - 2: **Output:** Predicted values of Power, Area, and Temperature
 - 3: **Step 1: Dataset Creation**
 - 4: Create dataset using simulation tool (PAT Noxim) based on input NoC parameters
 - 5: Extract Power, Area, and Temperature as output labels
 - 6: **Step 2: Data Preprocessing**
 - 7: Normalize or scale the dataset
 - 8: **Step 3: Train-Test Split**
 - 9: Split the dataset into Training, Validation, and Test sets (e.g., 70% training, 15% validation, 15% testing)
 - 10: **Step 4: Model Training**
 - 11: **ML Component:**
 - 12: Train a **Linear Regression** model for Power and Area prediction
 - 13: Train a **Decision Tree** model for Power and Area prediction
 - 14: **DL Component:**
 - 15: Train a **CNN** for Temperature prediction
 - 16: **Step 5: Model Evaluation**
 - 17: Use validation data to evaluate both models
 - 18: **Step 6: Combine Predictions**
 - 19: Combine the outputs from the ML and DL models for final Power, Area, and Temperature predictions
 - 20: **Step 7: Testing and Final Prediction**
 - 21: Test the combined model on the test set and make the final prediction
 - 22: **End**
-

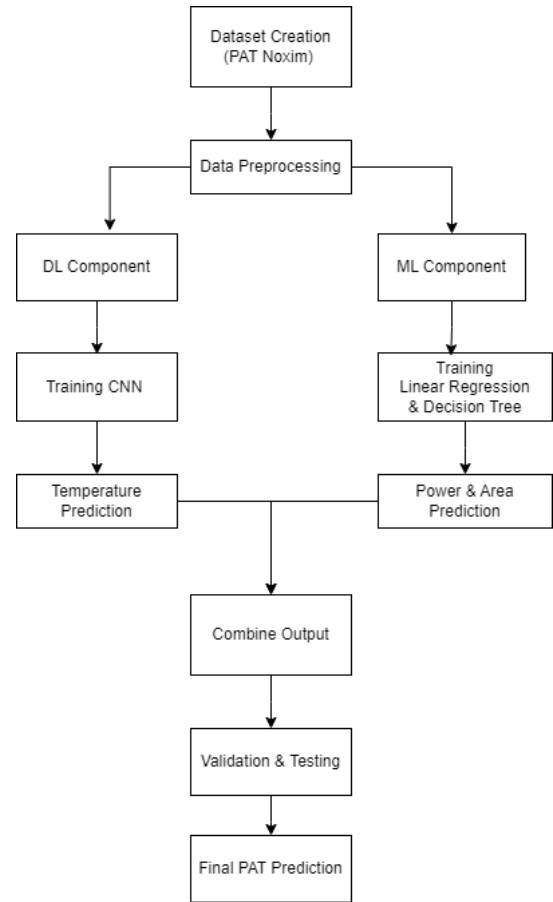


Fig. 1: Illustration of the Proposed Framework