**NoCeption: A Fast PPA Prediction Framework for Network-on-Chips Using Graph Neural Network :**

The paper *NoCeption: A Fast PPA Prediction Framework for Network-on-Chips Using Graph Neural Network* presents a novel framework for predicting the power, performance, and area (PPA) of Network-on-Chips (NoCs). Here's a summary of its key points:

**Background**

* **Network-on-Chip (NoC)** is an essential technology for communication within modern multiprocessor system-on-chips (MPSoCs). It provides scalability and modularity compared to traditional on-chip buses.
* Optimizing application-specific NoCs requires evaluating vast design spaces involving parameters like topology and buffer depth. This is an NP-hard problem and typically requires iterative design and evaluation cycles, which can be time-consuming.

**Problem**

* Traditional methods for predicting PPA (e.g., synthesis and simulation) are accurate but computationally expensive. Analytical and machine learning-based methods provide faster estimates but have limitations such as being overly tailored to specific NoC architectures.

**Proposed Solution: NoCeption**

* **NoCeption** is a framework based on Graph Neural Networks (GNNs) that offers fast and accurate PPA prediction for NoCs with **irregular topologies**.
* The method converts the NoC and its application into attributed graphs, which the GNN processes to generate predictions.

**Contributions**

1. **Flexible Prediction**: NoCeption generalizes well to different NoC architectures, making it more versatile than previous models, which often struggled with irregular topologies.
2. **Graph Representation**: The framework models NoCs as attributed graphs and uses GNNs to learn patterns that affect PPA, such as traffic patterns and congestion.
3. **High Accuracy**: Experiments on commercial NoC IPs show that NoCeption can predict PPA with high accuracy (over 97% for power and area) and outperforms baseline methods in network and system-level performance prediction.

**background and related work in Network-on-Chip**

The section outlines the background and related work in Network-on-Chip (NoC) systems, with a focus on PPA (Power, Performance, and Area) prediction. Here’s a breakdown of the content:

**A. NoC Basics**

* A classic NoC consists of **network interfaces (NIs)** and **routers**. NIs connect heterogeneous cores and convert protocols like AXI into internal network packets. Routers multiplex communication flows over the NoC interconnect.
* **Topology** plays a significant role in determining PPA by defining the connection pattern between NIs and routers.
* **Customization**: The heterogeneous nature of MPSoCs requires **irregular topologies** and customized NoCs with features like synchronizers for clock domain crossing and variable buffer depths in routers, which impact the PPA significantly.

**B. PPA Prediction for NoC Design**

* **Pre-silicon PPA estimation** is critical during NoC design to avoid time-consuming synthesis and simulations. Various methods have been proposed to accelerate PPA estimation.
* **Analytical model-based methods**: These methods rely on mathematical models. For example:
  + **ORION 2.0** models NoCs at the circuit level to estimate power and area, while performance is modeled using queuing theory.
  + Some models, like [9], view routers' channels as M/M/1 queues, which assumes exponentially distributed inter-arrival and service times.
  + Others extend traffic inter-arrival and service times to **general distributions** to capture more complex behavior.
* **Machine learning-based methods**: These do not require low-level implementation details and have better generalization capabilities.
  + **MARS (Multivariate Adaptive Regression Splines)**: Provides significant error reductions compared to ORION 2.0.
  + **SVR (Support Vector Regression)**: Can predict packet latency accurately and with a speed-up of 120× over traditional simulation.
  + **Industry use**: Companies like Arteris employ machine learning-based PPA prediction frameworks.
* However, traditional machine learning methods typically encode NoC topologies as **type and size vectors** (e.g., [0, 4] for a 4×4 mesh), which limits their ability to handle **irregular NoCs** that are common in real-world applications.

**C. Graph Neural Networks (GNNs) for NoC Modeling**

* **GNNs** offer a powerful way to model NoCs by representing them as **attributed graphs** and learning patterns that influence PPA. Unlike traditional methods like CNNs, GNNs can capture the **non-Euclidean nature** of the NoC’s topology and communication patterns.
* GNNs can encode factors such as **router radix, channel contention, and link workload** into a graph structure, enabling effective learning for tasks like classification and clustering.
* This inspires the application of GNNs to the domain of **PPA prediction for NoCs**, using minimal domain knowledge.

**Summary**

The section compares **analytical models** and **machine learning-based methods** for NoC PPA prediction, highlighting the limitations of each, particularly in handling **irregular NoC topologies**. The use of **Graph Neural Networks (GNNs)** is introduced as a promising approach for accurate PPA prediction in NoCs, thanks to their ability to process complex, graph-like structures inherent in NoCs.

The section outlines key concepts and methodology for evaluating Network-on-Chip (NoC) designs using Graph Neural Networks (GNNs). Here's a breakdown of the major points:

**1. NoC Description and Assumptions:**

* The NoC design focuses on application-specific NoCs with arbitrary topologies and parameters.
* The NoC system uses a transaction-based interface protocol, dividing Network Interfaces (NIs) into master and slave types.
* It assumes buffered wormhole flow control with deterministic routing, which impacts packet transfer across the NoC.
* The packet length is constant for the specific NoC.

**2. Task Graph (Definition 1):**

* An application is modeled as a **task graph** denoted as AG(C,F)AG(C, F)AG(C,F), where:
  + CCC represents cores.
  + FFF represents directed communications between cores.
  + Each communication edge has a weight representing the communication volume between cores.

**3. NoC Topology Graph (Definition 2):**

* The NoC topology is modeled as a directed graph TG(N,L)TG(N, L)TG(N,L), where:
  + NNN represents NIs or routers.
  + LLL represents directed communication links between NIs or routers.
* The deterministic routing algorithm provides the total traffic across these links, and a **workload graph (WG)** is derived based on the accumulated traffic on links, indicating network load and congestion.

**4. Graph Neural Network (GNN) Input (Definition 3):**

* The GNN uses an **attributed graph** G(V,E,X,Xe)G(V, E, X, X\_e)G(V,E,X,Xe​), where:
  + VVV and EEE represent nodes and edges, respectively.
  + XXX is a node feature matrix.
  + XeX\_eXe​ is an edge feature matrix, indicating features of edges (like link weight in NoC).
* The node and edge attributes are critical for predicting performance metrics such as power, area, and latency.

**5. Converting NoC Descriptions into Attributed Graphs:**

* **Power Estimation**: The NoC topology is converted into an attributed graph where node features reflect router or NI types, and edge features represent the workload of links. Power consumption is modeled based on the dynamic behavior of components like input buffers, crossbars, and arbiters.
* **Area Estimation**: Similar to power estimation, but without considering the traffic workload.
* **Performance (Latency)**: The latency of the NoC is based on end-to-end latency between master-slave pairs. A detailed **output port graph (OPG)** is used to model contention at router output ports.

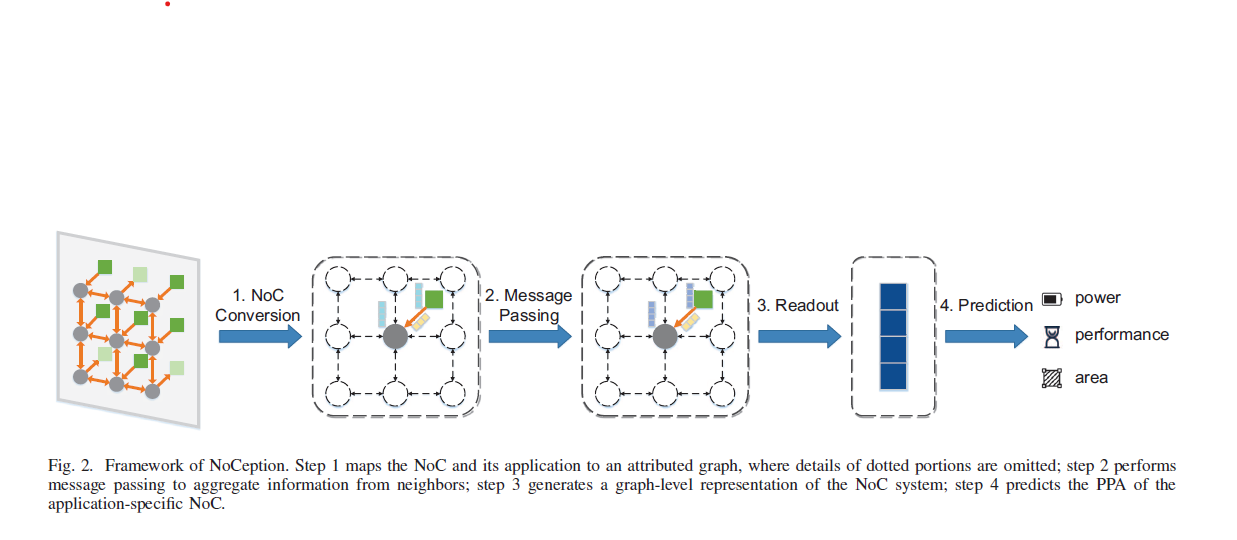
**6. Extracting Subgraphs for Latency Prediction:**

* End-to-end latency depends on both the routing path and the neighboring nodes around routers. Therefore, the framework extracts subgraphs from the overall NoC topology to better model congestion and predict latency more accurately.

**Summary:**

The methodology relies on converting the NoC's application task graph and topology into attributed graphs, which are then processed by a GNN. The GNN aggregates information through message passing to predict key metrics such as **power**, **area**, and **performance (latency)**. Special emphasis is placed on accurate modeling of router contention and congestion through the use of **output port graphs**.

This approach offers a structured way to analyze complex NoC designs and predict important performance metrics.



Framework of NoCeption. Step 1 maps the NoC and its application to an attributed graph, where details of dotted portions are omitted; step 2 performs message passing to aggregate information from neighbors; step 3 generates a graph-level representation of the NoC system; step 4 predicts the PPA of theapplication-specific NoC.

**Overview of NoCeption GNN Architecture**

1. **Purpose of NoCeption**:  
   NoCeption is a model that uses Graph Neural Networks (GNNs) to transform graphs (which represent systems like networks) into numerical representations (called embeddings) that can help predict performance metrics like power, area, and latency.
2. **Key Components**:  
   The architecture is divided into two main parts: **Message Passing Phase** and **Readout Phase**.

**1. Message Passing Phase**

* **What Happens Here?**
  + Each node in the graph (like a router or network interface) has some features (like its workload or congestion level).
  + In this phase, nodes send and receive messages to and from their neighbors to share information.
* **How Does It Work?**
  + Each node starts with its features and updates its state based on messages received from neighboring nodes. This is done over several iterations (let's say TTT times).
  + The formula used for updating a node’s state looks like this:

hvt+1=α(hvt+mvt+1)h^{t+1}\_v = \alpha(h^t\_v + m^{t+1}\_v)hvt+1​=α(hvt​+mvt+1​)

* + - Here, hvth^t\_vhvt​ is the current state of the node, and mvt+1m^{t+1}\_vmvt+1​ is the message it receives from its neighbors. The function α\alphaα helps to activate or transform this information.
* **Direction of Messages**:
  + There are two types of messages: incoming and outgoing. This helps to track how nodes are affected by their upstream (incoming) and downstream (outgoing) neighbors.

**2. Readout Phase**

* **What Happens Here?**  
  After the message passing is done, we need to summarize the information from all nodes to get a single representation for the entire graph.
* **How Does It Work?**
  + This summarization is done using a function called ρ\rhoρ, which can combine the hidden states of all nodes into one vector HGH\_GHG​.

HG=ρ({hvT∣v∈G})H\_G = \rho\left(\{h^T\_v | v \in G\}\right)HG​=ρ({hvT​∣v∈G})

* + - For instance, a simple way to do this could be just summing up all the hidden states of the nodes.

**3. Model Training**

* **How Do We Train the Model?**
  + To make the model learn better, it shares the message functions (which define how messages are generated) across different iterations. This reduces the number of parameters we need to train, making it more efficient.
  + The goal is to minimize the difference between the predicted performance metrics and the actual metrics. This is done using a loss function called Mean Squared Error (MSE).

**IV. Experimental Results**

**A. Datasets Generation and Experimental Setup**

1. **Data Generation**:
   * The process starts by generating synthetic task graphs based on the number of cores in a system using a tool called TGFF.
   * **Network Interfaces (NIs)**: If a network interface handles both incoming and outgoing traffic (which is not allowed in AXI protocol), it is split into two separate interfaces.
   * Random network topologies (like torus and tree structures) and architectural parameters (like clock domains) are created.
   * The task graph is mapped onto the NoC (Network-on-Chip) to get routing paths. If the mapping creates any cyclic dependencies that could cause deadlocks, those paths are discarded, and the process restarts.
   * Finally, a commercial NoC IP (intellectual property) design is synthesized and simulated to gather data on power, area, and latency (PPA).
2. **Experimental Setup**:
   * The maximum number of cores in the generated task graphs is 20, leading to a maximum of 40 network interfaces after splitting.
   * Different configurations of regular and irregular topologies are used. For homogeneous configurations, all routers have a buffer depth of 4 and run at a clock frequency of 100 MHz. For heterogeneous configurations, buffer depths can vary from 4 to 12, and clock frequencies can be 100, 150, or 200 MHz.
   * **Routing and Packet Size**: Shortest path routing is applied, with packets set to a size of 64 bytes and packet injection modeled to follow a Poisson distribution.
   * The experiments generated a dataset with a total of **21,000 samples**.

**B. Experimental Results**

1. **Accuracy Evaluation**:
   * To evaluate prediction accuracy, the Mean Absolute Percentage Error (MAPE) is calculated. This helps measure how close the predicted values are to the actual values:

MAPE=100%×1k∑i=1k∣yi−y^iyi∣\text{MAPE} = 100\% \times \frac{1}{k} \sum\_{i=1}^{k} \left|\frac{y\_i - \hat{y}\_i}{y\_i}\right|MAPE=100%×k1​i=1∑k​​yi​yi​−y^​i​​​

* + Where kkk is the number of samples, yiy\_iyi​ is the actual value, and y^i\hat{y}\_iy^​i​ is the predicted value.

1. **NoCeption on Synthetic Applications**:
   * The first set of tests involved using randomly generated task graphs and network topologies.
   * The results showed that the prediction accuracy for power and area was high (specific values not provided in the excerpt).
   * For latency prediction, NoCeption was compared with a baseline method called SVR-NoC, which only considers one-hop neighboring routers and is limited to mesh topology and homogeneous configurations.
   * **Findings**: NoCeption showed significant improvements in latency prediction, achieving about **6.19% better for end-to-end latency** and **2.56% better for global latency** compared to SVR-NoC.
   * It was noted that using the output port graph for latency modeling performed significantly better than using the workload graph.
2. **NoCeption on Synthetic Traffic Patterns**:
   * Models trained on synthetic applications were tested with various unseen traffic patterns (like complement and reverse).
   * NoCeption consistently outperformed SVR-NoC, particularly at higher packet injection rates. The experiment found that as the injection rate increased, the latency leveled off due to the network's transaction mechanisms.
3. **NoCeption on Real Applications**:
   * The models were also tested on real-world benchmarks (like MMS, MPEG4) by mapping them randomly onto NoCs.
   * Results showed very high accuracy: **97.36% for power** and **97.83% for area** on average, with **93.44% and 95.92%** in the worst cases.
   * Additionally, NoCeption improved end-to-end and global latency predictions by about **6.52%** and **4.73%** on average compared to SVR-NoC.

**Summary of Findings**

* **General Performance**: NoCeption demonstrates strong prediction capabilities for power, area, and latency across both synthetic and real-world applications.
* **Robustness**: The model can generalize well to different network sizes and architectures.
* **Improvement Over Baseline**: Compared to SVR-NoC, NoCeption achieves better accuracy in predicting latency, particularly in heterogeneous networks and under varying traffic conditions.

**Summary of Contributions**

1. **Proposed Framework**:
   * The paper introduces a Graph Neural Network (GNN)-based framework named NoCeption specifically designed for predicting Power, Area, and Latency (PPA) in application-specific NoCs.
   * This framework can handle arbitrary topologies and heterogeneous parameters, making it versatile for different design scenarios.
2. **Transformation into Attributed Graphs**:
   * The framework transforms the NoC and its associated tasks into an attributed graph. This graph representation allows the GNN to effectively learn and make predictions about the PPA of the NoC.
3. **Validation**:
   * The effectiveness of the proposed method is validated using datasets generated from a commercial NoC Intellectual Property (IP) design.
   * The results indicate that NoCeption is effective across a variety of applications, demonstrating its practical applicability in real-world scenarios.

**Conclusion**

The NoCeption framework represents a significant advancement in the prediction of PPA for NoCs, providing a reliable and efficient approach to optimize designs based on varying architectures and workloads. By leveraging GNNs and attributed graph representations, the framework shows promise for enhancing the design and performance assessment of application-specific NoCs.

Limitations:

**Dependency on Synthetic Data**

* **Explanation**: The paper primarily relies on synthetic datasets generated from specific NoC IPs, which may not encapsulate the full variability of real-world applications, workloads, or unexpected behavior in live systems.
* **Implication**: The findings may not fully transfer to real-world scenarios, leading to less reliable predictions when applied to actual NoC implementations.