Congestion Prediction in Integrated Circuit Design Using Graph Neural Network

Yuyang Pang[1], Michelle Tong[1], Hang Liu[1], Vivian Chen[1]
yupang@ucsd.edu m1tong@ucsd.edu hal064@ucsd.edu vnchen@ucsd.edu
[1] Halicioglu Data Science Institute, University of California San Diego

Mentors: Lindsey Kostas[2], Guillaume Shippee[2], Lea Hagen[2]
lkostas@qti.qualcomm.com shippee@qti.qualcomm.com lhagen@qti.qualcomm.com
[2] Qualcomm Inc.

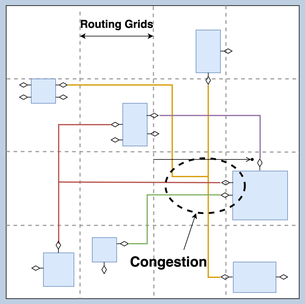
UC San Diego™
HALICIOĞLU DATA SCIENCE INSTITUTE
Qualcomm

Project Website

Introduction

Chip design is crucial for technological advancement, for it directly influences the performance of electronic devices.

A key component of chip design is the **netlist**, which can be seen as a giant graph that connects different logic functions to each other, guiding the assembly of circuits. However, as more logic functions are added, the **demand** for more connections **exceeds** the **capacity** available, which leads to a major challenge in chip design called **congestion**, and this can impede chip manufacturing.

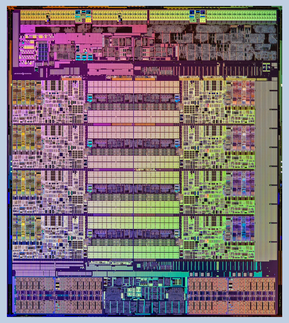


△ Figure 1: Congestion in the Design Flow of a Netlist

Motivation

This project aims to **predict congestion** in Integrated Circuit (IC) Design using **Graph Neural Networks** (GNNs).

GNNs can model the intricate, non-Euclidean relationships between circuit components represented as graphs. This allows for effective learning and forecasting of congestion by capturing the complex dependencies inherent in IC layouts.



△ Figure 2: Complexity of Modern chips

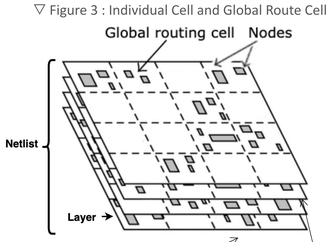
Dataset

We utilized the NCSU-DigIC-GraphData, which comprises 13 netlists, each exhibiting unique **placement** and **congestion** characteristics.

For each netlist, the dataset contains **node features**, **node connectivity**, and Global Route Cell (GRC) based **congestion** (demand minus capacity).

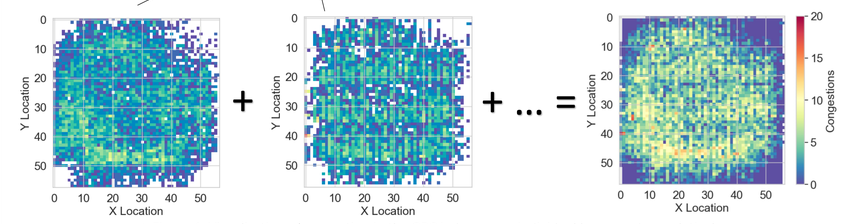
Node features and node connectivity refer to the characteristics and interconnections of individual elements within the netlist. GRC-based congestion calculates congestion at a GRC index with multiple nodes.

▽ Figure 3 : Individual Cell and Global Route Cell



Netlists	Cells	Edges	Layers	Edges Per Cell				Congestions			
				Min	Max	Mean	STD	Min	Max	Mean	STD
XBar_1	3952	4482	4	15	862	583.39	330.37	-48.0	14.0	-9.42	6.12
XBar_2	6872	7404	7	8	1253	166.06	253.38	-86.0	3.0	-28.55	12.52
XBar_3	6913	7445	7	8	1277	166.45	251.10	-86.0	-11.0	-35.14	10.01
XBar_4	7323	7855	7	8	1274	144.56	238.81	-86.0	-11.0	-34.64	9.73
XBar_5	7258	7790	7	8	1254	145.72	233.62	-86.0	-13.0	-35.23	9.73
XBar_6	7120	7652	6	8	1253	136.50	187.04	-78.0	-8.0	-28.69	9.13
XBar_7	7879	8411	5	7	1277	128.23	212.35	-65.0	20.0	-15.78	9.55
XBar_8	7626	8158	5	7	1274	135.84	224.04	-67.0	18.0	-18.43	9.22
XBar_9	7620	8153	5	6	1229	132.03	206.79	-76.0	-8.0	-29.67	9.23
XBar_10	7772	8304	6	8	1292	130.48	216.23	-78.0	-11.0	-31.68	8.36
XBar_11	7814	8346	5	8	1283	118.99	174.60	-67.0	22.0	-16.33	8.73
XBar_12	6529	7061	5	8	1280	187.29	271.56	-67.0	15.0	-14.10	9.83
XBar_13	6548	7082	5	6	1283	178.85	248.45	-65.0	29.0	-11.19	9.76

△ Table 1: Dataset Statistic



△ Figure 4: Visualization of congestion in the GRC index across individual layers and cumulatively.

Methodology

Features

- **xloc** and **yloc**, **width** and **height**: Represent the location and size of the nodes and can be used to discern congestion and elemental spatial occurrence.
- **orient**: Might indicate the preferred direction of routing, potentially revealing congestion-prone areas.
- **edge index**: Information obtained from the adjacency matrix. Helps the model learn relationships and pathways within the graph structure.

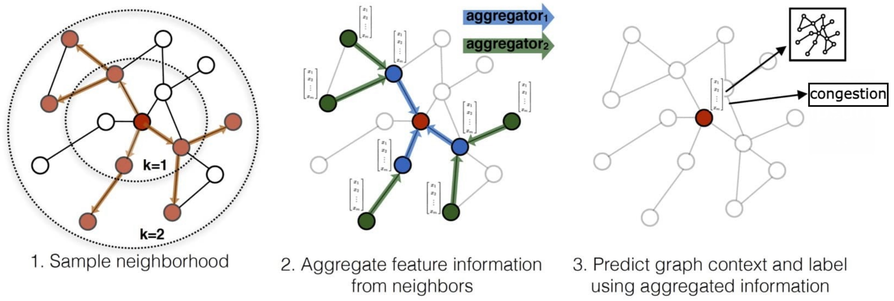
Positional data and target congestion is normalized so that the models generalize across designs with differing dimensions

	xloc	yloc	width	height	orient	congestion
0	41984	44544	2176	1536	0	-8.0
649	62336	72192	896	1536	6	4.0

△ Table 2: Node Features

Model Selection

We use GNNs to predict congestion, given that netlists can be represented as graphs. Each node represents a cell, and an edge exists when two cells are connected by a route.



1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

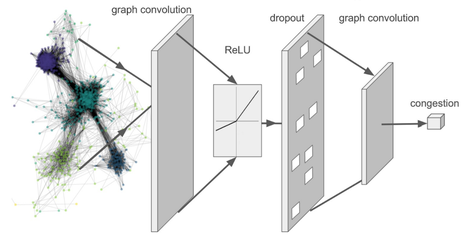
△ Figure 5: Graph Neural Network Flowchart

Graph Neural Networks

Model Architectures

Graph Convolutional Networks (GCN):

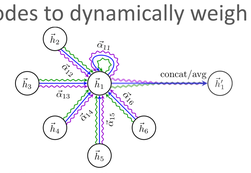
- Generalize convolutional operations from grid data to graph data



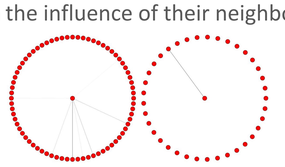
△ Figure 6: GCN Architecture

Graph Attention Networks (GAT):

- Enhance GCN by incorporating attention mechanisms, allowing nodes to dynamically weigh the influence of their neighbors



△ Figure 7: GAT Attention Mechanism



△ Figure 8: Node Attention

Model Training

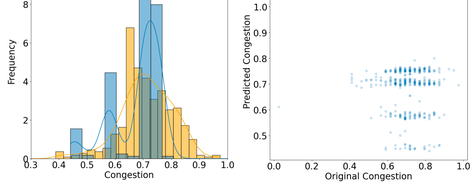
Data balancing

- To solve the lack of variance of predicted congestion, we implemented **a weighted loss criterion,** where our targets are split into bins and their weight is the inverse frequency of their respective bin.

Train Test Split

- **Training within a single design**. We randomly pick 80%/10%/10% of nodes within one netlist to serve as our training, validation and testing set.
- **Training across designs**. We leave one netlist out and utilize the other 12 netlists as our training set. This cross-files' training method broadens the scope of our model's learning to a wider array of data variations, which enhance its generalization capabilities.

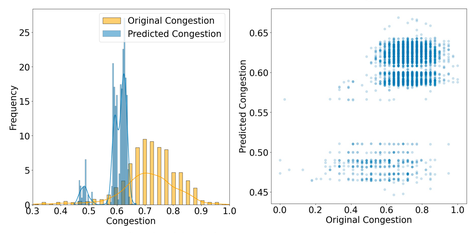
GCN-Trained on a Single Design



△ Figure 9 : Predicted and Original Congestion Histogram

△ Figure 10 : Predicted and Original Congestion Scatterplot

GCN-Trained Across Designs



△ Figure 11 : Predicted and Original Congestion Histogram

△ Figure 12 : Predicted and Original Congestion Scatterplot

Insights

Based on the results in table 3, we found that GNNs tend to predict mean values for the congestion in the NCSU datasets, which are relatively smaller by the netlist sizes. Even though we conducted hyperparameter tuning on the models to experiment with different numbers of layers, numbers of hidden channels, and so on, we failed to improve the models' performance significantly better than a naïve mean predictor.

Model		Single-Design		Cross-Design	
		RSME	Pearson	RSME	Pearson
Cell-Based	GCN	0.140	-0.015	0.144	0.445
	GAT	0.010	-0.128	0.151	0.463

△ Table 3: Model Performance

Therefore, we believe GNNs in general may require more training data to understand the connectivity of a graph.

Moreover, we noticed that GATs tend to perform better than GCNs, even though GATs take longer time to train.

Future Work

Given our suspicion on the scaling issue of GNNs, we want to test our model on the larger netlists from the SuperBlue datasets, which contains hundreds of thousands of nodes.

Furthermore, we are interested in trying other ML models like XGBoost and random forest regressor for predicting congestion.

More node features, such as the layer information of nodes and routing information, could be collected and included in the model..