Machine Learning Techniques for Routability-driven Routing in Application-specific Integrated Circuits Design

by

Yuxuan Pan

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

August 2022

© Yuxuan Pan, 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Machine Learning Techniques for Routability-driven Routing in Application-specific Integrated Circuits Design

submitted by **Yuxuan Pan** in partial fulfillment of the requirements for the degree of **Master of Applied Science** in **Electrical and Computer Engineering**.

Examining Committee:

André Ivanov, Professor, Department of Electrical and Computer Engineering, UBC
Supervisor

Konrad Walus, Associate Professor, Department of Electrical and Computer Engineering, UBC
Supervisory Committee Member

Guy Lemieux, Professor, Department of Electrical and Computer Engineering, UBC
Supervisory Committee Member

Abstract

Routing is a challenging stage of the Integrated Circuit (IC) design process. A routing algorithm often adopts the two-stage approach of global routing followed by detailed routing. One of the routing objectives is the routability, which requires completing all the required connections without causing routing overflows or wire-shorts. Otherwise, the chip would not function well and may even fail. Moreover, detours need to be taken to avoid overflows and wire-shorts, which may increase the wire length and number of vias in the physical design, affecting the overall performance of the circuit. Predicting the existence and locations of routing overflows and wire-shorts before routing takes place helps the router to improve the routability and circuit performance. Here, we present two Machine Learning (ML) techniques that improve the routability of routing by predicting the number and locations of overflows and wire-shorts.

First, we design and develop GlobalNet, a Fully Convolutional Network (FCN) based global routing congestion predictor that estimates the density of wires and vias of global routing in 3-Dimensional (3D) from a placed netlist. The locations of overflows are derived from the prediction result. A global router is also implemented to utilize the congestion estimation result to improve the performance of global routing.

Second, a Convolutional Neural Network (CNN) based wire-short predictor, VioNet, is developed. VioNet replaces the global router with global routing congestion estimation (GlobalNet) so that the runtime is significantly reduced. To improve the prediction accuracy, we adopt a top-down iterative strategy where a low-resolution prediction first gives the approximate locations of wire-shorts, followed by a high-resolution prediction which determines the precise wire-shorts'

locations.

Experimental results show that both GlobalNet and VioNet achieve high accuracy on ISPD 2018 and ISPD 2019 benchmarks. Moreover, UBC-GR increases the routability of global routing by reducing the number of overflows.

Lay Summary

Routing is a vital stage in the process of IC design, where a router connects the components in a design using wires, with certain rules to follow. Routability, which refers to the ability of a design being routed by a specific router without violating routing rules, is one of the most important objectives to optimize in the routing stage. This thesis presents two ML frameworks that predict the result of routing before routing takes place. The predicted rule violations can be avoided in the routing stage, and the routability of a design can be improved. Experimental results show that our prediction is faster and more accurate than previous works, and improvements are achieved in terms of routability.

Preface

Chapter 3. A version of this material has been published as Pan Y, Zhou Z, Ivanov A. Routability-driven Global Routing with 3D Congestion Estimation Using a Customized Neural Network. In 2022 23rd International Symposium on Quality Electronic Design (ISQED) 2022 Apr 6 (pp. 1-6). IEEE. I am the first author, responsible for the code implementation, parameter tuning, data gathering and result analysis with the assistance of Zhonghua Zhou. I performed the experiments. Zhonghua Zhou and I conceived the experiments and I wrote the majority of the manuscript for the published paper under the supervision of my supervisor, Prof. André Ivanov. Prof. André Ivanov also provided research methodology guidance.

Chapter 4. Zhonghua Zhou provided the idea of iterative structure of the VioNet. I designed the rest of the VioNet and finished the code implementation, parameter tuning, data gathering and result analysis with the assistance of Zhonghua Zhou. The experiments were conducted by me. I wrote the majority of the manuscript under the supervision of my supervisor, Prof. André Ivanov. Prof. André Ivanov also provided research methodology guidance.

Table of Contents

List of Tables

List of Figures

List of Acronyms

2D	2-Dimensional
3D	3-Dimensional
ASIC	Application-Specific Integrated Circuit
CNN	Convolutional Neural Network
DRV	Design Rule Violation
EDA	Electronic Design Automation
FCN	Fully Convolutional Network
FPGA	Field-programmable Gate Array
GAN	Generative Adversarial Network
HPWL	Half Perimeter Wirelength
IC	Integrated Circuit
MARS	Multivariate Adaptive Regression Splines
ML	Machine Learning
MLP	Multilayer Perceptron
NMSE	Normalized Mean Square Error)
NN	Neural Network
PCC	Pearson Correlation Coefficient
RUDY	Rectangular Uniform wire DensitY
RL	Reinforcement Learning

RSMT Rectilinear Steiner Minimal Tree

VLSI Very-Large-Scale Integration

Acknowledgments

Words cannot express my gratitude to my supervisor, Prof. André Ivanov for his expert advice and encouragement throughout this challenging project. Also, I could not have undertaken this journey without Dr. Zhonghua Zhou, who generously guided me in doing this project. I also would like to express my deepest appreciation to Prof. Guy Lemieux and Prof. Konrad Walus for being my committee members. Special thanks to UBC, NSERC and Huawei for the financial support. Lastly, I'd like to mention my family, especially my mom. Their belief in me has kept my spirits and motivation high during this process.

Chapter 1

Introduction

Routing is an essential stage in the design of Integrated Circuits (ICs) [15, 33]. The design of modern Application-Specific Integrated Circuits (ASICs) usually adopts a standard cell methodology, where a low-level Very-Large-Scale Integration (VLSI) layout is represented by standard cells, which are groups of transistors and interconnect structures that provide boolean logic functions (e.g., NAND, OR, inverters) or storage functions [32]. The *routing* process determines the precise paths required for metal wires on the chip layouts interconnecting the standard cells under certain constraints. Objectives, such as routability, wire length, area, power consumption, are optimized in the process of routing. Some constraints need to be obeyed in the process of routing, such as two wires cannot be to close to each other, wires cannot overlap, etc. [15, 69]. Here, we define *routability* as an attribute that describes the ability of a placed netlist to be routed by a specific router without violating the constraints of routing. This thesis focuses on improving the routability of a placed netlist by predicting the existence and locations of routing congestion and wire-short to improve the performance of a router.

1.1 Electronic Design Automation (EDA) of Physical Design

Physical design of ICs is the process of turning a IC design into manufacturable geometries [14, 33]. Most of the work of physical design was done "by hand"

until the late 1960s [32, 48, 66], when the complexity of IC increased, and manual design became inefficient. The concept of EDA is to use automatic tools to improve the efficiency and the quality of physical design. It was first introduced in the 1970s for automatic circuit simulation and routing [3, 9, 53, 60]. After decades of development, the workflow of physical design has become very modular, and EDA is an essential part of all the stages of the physical design process. Figure 1.1 shows the workflow of EDA. The EDA of physical design comprises a number of stages, including synthesis, floorplanning, placement, and routing [38].

1.1.1 Synthesis

As shown in block (1) in Figure 1.1, in *synthesis* stage, a physical design written in hardware description languages, such as Verilog, is compiled and mapped into a netlist. A netlist is a description of the gate-level connectivity of an electronic circuit.

1.1.2 Floorplanning

The block (2) in Figure 1.1 shows the progress of *floorplanning*. A floorplanner takes the netlist as the input and partitions the design into multiple functional blocks [58]. The size of the chip region, called die area, is also determined in floorplanning. Figure 1.1 (c) shows an example of floorplanning result.

1.1.3 Placement

The *placement* stage is demonstrated in block (3) in Figure 1.1, a placer is used which assigns exact locations for various circuit components within the chip's core area after floorplanning. The output of the placer is called a "placed netlist".

1.1.4 Routing

Shown in block (4) in Figure 1.1 is the *routing* stage, which determines the location for metal wires on the chip area to interconnect the pins of standard cells. In modern VLSI design, billions of wires are to routed. For example, the Apple's ARM-based dual-die M1 Ultra system on a chip contains more than 114 billion transistors [5]. Due to the complexity of routing, it is usually divided into two

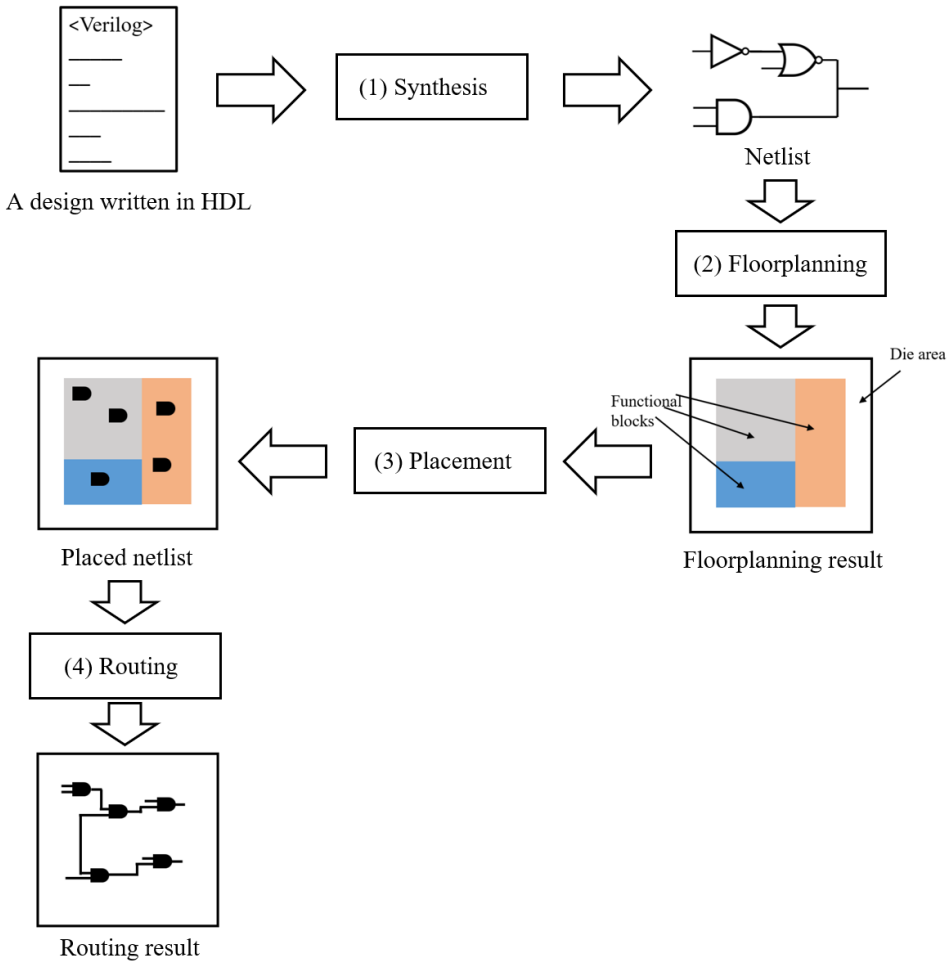


Figure 1.1: Workflow of EDA.

cascaded steps: global routing and detailed routing [15]. The global routing determines the approximate location of each wire, while the detailed routing sets the final and precise path for each wire based on the result of global routing stage.

1.2 Motivation

The routing phase of IC design determines the locations of wires interconnecting the cell pins under certain constraints, with multiple objectives to be optimized.

It has been proven to be an NP-complete problem [61], a class of computational problems that is the hardest of the problems to which solutions can be verified quickly. As the semiconductor process keeps advancing, the complexity of routing also increases. For example, the number of transistors on a microchip has been increasing tremendously since the 1960s [50]. Unfortunately, the time complexity of the routing problem increases exponentially as the number of cell pins increases. The increased number of design rules adds to the problem. Design rules refer to the constraints that the router must follow in order to create reliable and functional circuits. If such rules are violated, a rip-up-reroute [20] stage will be performed where the routed wires are removed and a new round of routing is done. The rip-up-reroute is done iteratively until the router finds a solution without causing routing rule violations, and the placed netlist is considered "routable" to the router. If the router cannot remove all the violations in a feasible time, the placed netlist is considered "unroutable" and needs to be replaced with a new placement strategy. Both rip-up-reroute and replacing are time consuming and cannot guarantee a result without any routing rule violations. Therefore, predicting the existence of routing rule violations can be used to evaluate the routability of a placed netlist before routing, preventing the waste of time on unroutable placed netlists.

Moreover, predicting the locations of routing rule violations can improve the routability of routing. The router routes all the wires sequentially. As the routing keeps evolving, the die region gets more congested with routed wires. Because two different wires cannot overlap, the unrouted wires have to take detours due to the blockages caused by the previously routed wires, resulting in a longer wire length for the unrouted wires. The longer wires require more space in the die region, making the die region more congested. Violations are more likely to happen in congested areas. Suppose the locations of routing rule violations can be predicted before routing. In that case, the router can avoid routing through the areas that are prone to routing rule violations in early stages, thus preventing routing rule violations from happening.

Many research efforts have been put into routing prediction that can predict the existence and locations of routing rule violations. Traditional prediction approaches like Rectangular Uniform wire DensitY (RUDY) [59] and probabilistic-based methods [46, 68] have existed for decades and no longer suit the modern

VLSI technology because of their low prediction accuracy and speed. In recent years, significant improvements have been achieved by applying Machine Learning (ML) techniques to routing congestion estimation and wire-short prediction. However, most previous works, for example, [4, 16, 41, 54], on routing congestion estimation are 2-Dimensional (2D), which means that they do not predict on which metal layer the congestion happens. Also, the relationship between routing congestion estimation and wire-short prediction has not been investigated. This thesis aims to improve the routability of placed netlists by predicting routing congestion and wire-shorts fast and accurately before routing takes place using ML techniques.

1.3 Approaches and Contributions

The goal of this thesis is to predict the existence and locations of routing rule violations from a placed netlist (*i.e.*, before routing takes place). As technology nodes get smaller, traditional prediction approaches, such as RUDY and probabilistic based methods, become less effective in terms of both runtime and performance. Recently, ML has shown its potential to improve routing runtime and routability in EDA. Many research efforts have been put into ML-based routing congestion estimation and wire-short prediction. Although the prediction of routing results is believed to be able to help routers improve routing performance, such application of routing congestion prediction is hardly seen. This thesis proposes two ML frameworks to predict the routing congestion and wire-shorts from a placed netlist. A global router that can utilize the prediction result to improve the routability is also proposed.

Chapter 3 presents "GlobalNet", a 3-Dimensional (3D) routing congestion estimation algorithm that uses a Fully Convolutional Network (FCN) [45] to predict the density of wires and vias of each metal layer from a placed netlist. Via refers to the physical connection that interconnects wires on different metal layers. Pearson Correlation Coefficient (PCC) and Normalized Mean Square Error) (NMSE) of the prediction results and the ground truth are used to evaluate GlobalNet. Experiment results show a high average PCC (0.8449) and a low NMSE (0.0396) on ISPD 2018 [18] and ISPD 2019 [2] benchmarks, which indicates a high accuracy. Chapter 3

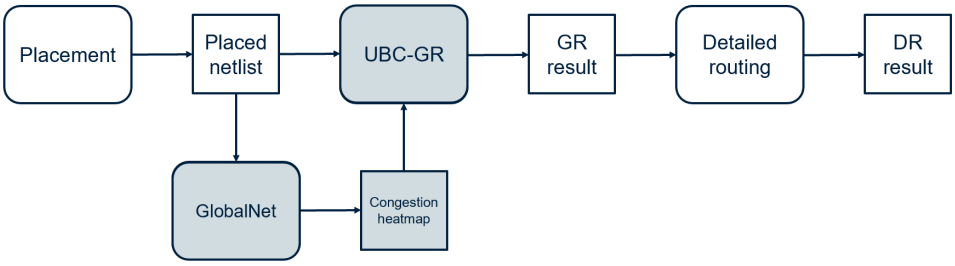


Figure 1.2: Workflow of GlobalNet.

also introduces UBC-GR, a global router modified from an open-sourced global router: CU-GR [42]. UBC-GR is able to use the estimation result of GlobalNet to guide the router to avoid potential congestion. Experiments of UBC-GR on ISPD 2018 and ISPD 2019 benchmarks show an improvement in terms of the number of vias and number of overflows. Moreover, two designs that are unroutable to CU-GR are successfully routed by UBC-GR, meaning that the routability of a placed netlist is improved, and the risk of routing failure is decreased by UBC-GR. Figure 1.2 summarizes the workflow of the proposed GlobalNet. It is done after the placement stage to predict the routing congestion of global routing. The results, called "congestion heatmaps", are used to improve the performance of global routing.

Chapter 4 describes VioNet, a customized Convolutional Neural Network (CNN) [51] that predicts the locations of detailed routing wire-shorts. As shown in Figure 1.3, the input of VioNet is the results of global routing congestion estimation (the congestion heatmap generated by GlobalNet). Different from other wire-short predictors that do prediction based on global routing results. VioNet utilizes congestion heatmap, which is an estimation of global routing results. Therefore, the long runtime of global routing can be saved and the wire-short prediction can be performed in an earlier stage, i.e., after placement. Moreover, to improve the accuracy of prediction, VioNet adopts an iterative strategy where a low-resolution prediction is done first to determine the approximate locations of wire-shorts, followed by a high-resolution prediction that spots the exact locations of wire-shorts. Experiments on ISPD 2019 benchmarks show that 74% of the wire-shorts are successfully predicted by VioNet. Besides the high accuracy, VioNet is also 92 times faster than Dr. CU, an open sourced detailed router, making it a good tool to eval-

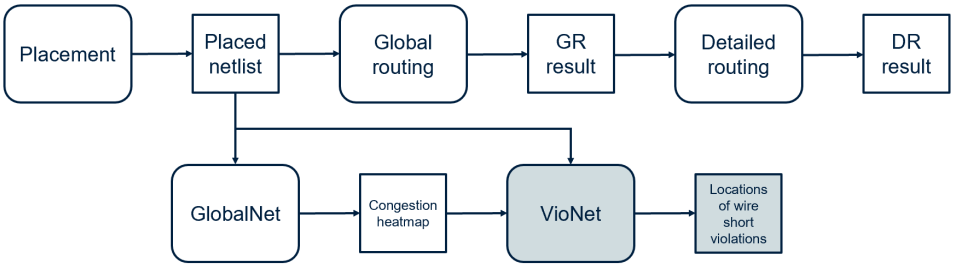


Figure 1.3: Workflow of VioNet.

uate the routability of a placed netlist without routing.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 1 presents the background material regarding the routing algorithm and previous efforts on routing congestion estimation and wire-short prediction. Chapter 3 presents GlobalNet and UBC-GR, which aims to improve the routability of global routing using ML based congestion estimation. Based on the work of Chapter 3, Chapter 4 describes VioNet, a CNN based detailed routing wire-short predictor that utilizes a global routing congestion estimation result to improve prediction accuracy. Lastly, Chapter 5 concludes the thesis.

Chapter 2

Background

2.1 Routing

2.1.1 Problem Formulation

The goal of all routers is the same. They are given some pre-existing pins (also called terminals) of standard cells, and optionally some pre-existing wiring called preroutes. Each of these pins is associated with a net, usually by name or number. The primary objective of the router is the routability, which requires the router to create geometries such that all pins assigned to the same net are connected by wires, no pins assigned to different nets are connected, and all design rules are obeyed [15]. A router can fail by not connecting pins that should be connected (an open), or by mistakenly connecting two pins that should not be connected (a short). In 3D IC design, the wires are distributed on different metal layers, and vias are used to connect the wires on two different metal layers [24, 55].

Figure 2.1 illustrates the relationship between pins, vias, wires and nets. P_1 and P_2 are two pins on metal layer 1, w_1, w_2 and w_3 are three metal wires connecting P_1 and P_2. w_1 and w_2 are on metal layer 1, while w_3 is on metal layer 2. Therefore, four vias: v_1, v_2 on metal layer 1 and v_3, v_4 on metal layer 2 are used to connect w_1, w_2 and w_3. All the pins, wires and vias shown in Figure 2.1 form a net as they are all physically connected.

In addition, to correctly connect the nets, routers may also be expected to make

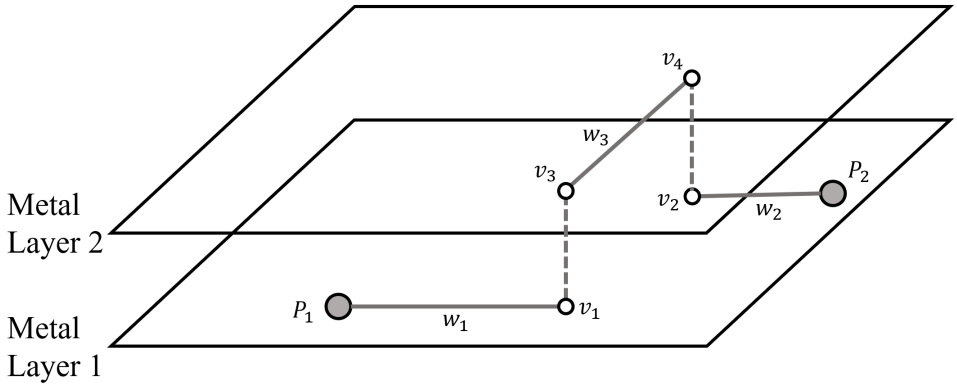


Figure 2.1: An example of pins, vias, wires and a net.

sure the design meets timing [6], has no crosstalk problems [36], meets any metal density requirements [40, 64], does not suffer from antenna effects [47], and so on. This long list of often conflicting objectives is what makes routing extremely difficult.

To conclude, the inputs and output of the general routing problem are listed as follows:

Inputs:

- A placed layout with fixed locations of standard cells and cell pins on the chip.
- A netlist, which is a description that consists of a list of the electronic components in a circuit and a list of the nodes they are connected to.
- A set of design rules and other constraints for a manufacturing process, such as resistance, capacitance, and the wire/via width and spacing of each layer.

Output:

A wire connection map for each net presented by actual geometric layout objects that meet the design rules and optimize the given objective, such as routability, timing, power, area, wire length, etc., if specified.

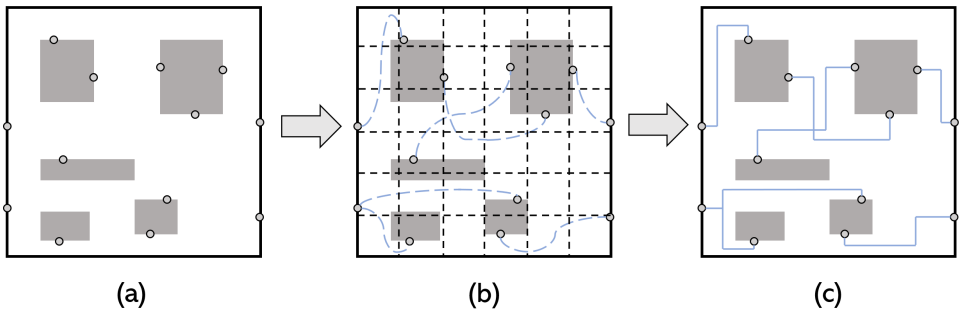


Figure 2.2: An explanation of global routing and detailed routing.

2.1.2 Two-step Approach

Due to the complexity of routing, a two-step approach of global routing followed by detailed routing is adopted. Generally [15], global routing first partitions the routing region into tiles which are called "G-cells" and decides tile-to-tile paths for all nets while attempting to optimize some given objective function (e.g., total wire length and circuit timing). Then, guided by the paths obtained in global routing, detailed routing assigns actual tracks and vias for nets. Figure 2.2 illustrates the process of global routing and detailed routing. Figure 2.2 (a) shows a placement result that contains the information about the exact locations of standard cells, pins of standard cells, and I/O pads at chip boundaries. The placed netlist also describes a list of connections by indicating which pins should be electrically connected to form a set of nets. Figure 2.2 (b) shows the result of global routing. The design is partitioned into multiple G-cells. The global router provides the approximate route for each connection by finding the tile-to-tile paths to connect pins. Figure 2.2 (c) illustrates the detailed routing result, which describes the exact route for each net by searching within the tile-to-tile path obtained in global routing.

2.1.3 Routability

Due to the limitation of foundry technology, certain constraints must be applied to routing in order to fabricate reliable and functional chips. For example, some of the constraints set a minimum space between wires and a maximum density of metal areas. Routability is commonly defined as the ability of a placed netlist to

be routed by a specific router without violating the constraints. Routability is essential because any violated constraint may lead to a failure in fabrication or the malfunctioning of the chip. As a result, the design needs to be rerouted or even replaced, and the time of the design process is significantly increased. Different routers adopt different routing strategies. Therefore, the routability of a placed netlist can be improved by using a better router. The number of constraint violations is one of the objectives to optimize in routing stage - the goal of all routers is to achieve a routing solution without violations. Therefore, routability is usually considered as a continuous metric, which is measured by the number of violations of routing constraints. Specially, a placed netlist is "routable" if no constraint is violated after routing, and is considered "unroutable" if any constraints are violated. In this thesis, two constraint violations, global routing overflow and detailed routing wire-short, are taken into consideration.

Global Routing Overflow

It is common to use the number of routing overflows to evaluate the routability of global routing. As discussed in Section 2.1.2, in the process of global routing, a design layout is partitioned into multiple G-cells. The fabrication technology and the placement layout determine the maximum number of wires and vias that can exist in each G-cell. An overflow happens when the actual number of wires and vias exceeds the maximum limit of that G-cell. Here, we use the number of overflows to measure the routability of a placed netlist in global routing. A placed netlist with fewer overflows after global routing is considered to have a better routability than a placed netlist with more overflows.

Detailed Routing Wire-short

Design rules must be obeyed for detailed routing. Some of the design rules include minimum space between wires and vias, the width of wires, etc. The number of Design Rule Violations (DRVs) after detailed routing is commonly used to measure the routability of a placed netlist in the detailed routing stage. The netlists with fewer DRVs after detailed routing are considered to have a better routability. In this thesis, to simplify the problem, only one design rule, wire-short, is taken into

consideration. Wire-short happens when two or more wires overlap. It is one of the most commonly existing DRVs and therefore, one of the most representative ones among all the DRVs.

2.2 Routing Congestion Estimation

It is essential to predict the routability of a placed netlist before routing takes place or during the process of routing. A placed netlist that is predicted to have a low routability can be replaced for a better routability without going through the lengthy routing stage. Moreover, the prediction of routability can be used to guide a router to avoid potential wire-shorts or overflows in the process of routing. In industry, fast trial global routing is often employed for routability prediction at placement or routing stage [67]. Fast trial global routing is a routing stage to estimate the distribution of wires and vias after global routing. The "fast" here is relative to the full-fledged global routing that generates solutions for further detailed routing. Compared to the routing congestion estimation, such as RUDY, probabilistic-based algorithms, etc., such trial global routing is still too time-consuming.

For a given area on a design layout, routing congestion estimation predicts the information of wires and vias in an area, such as the wire length, number of vias, *etc*.

2.2.1 RUDY

Wires are some metal segments that connects the components in a design. Wires have certain width and length, and therefore take up space. RUDY estimates the probability that a point is located within a wire region, which is also referred to as "wire density".

Given a design, an *x*-*y* plane can be build as shown in Figure 2.3. The lower left corner O of the design is the origin of the plane. For any point P in the design with its coordinate being (x_p, y_p). $RUDY(x_p, y_p)$ computes the wire density of point P, which is an estimation of the probability that P is located within a wire region.

With the wire density, we can estimate the wire area of any given region in a design. As shown in Figure 2.3, the grey area S is a region within the design area, the area of wires in S can be estimated using the following equation:

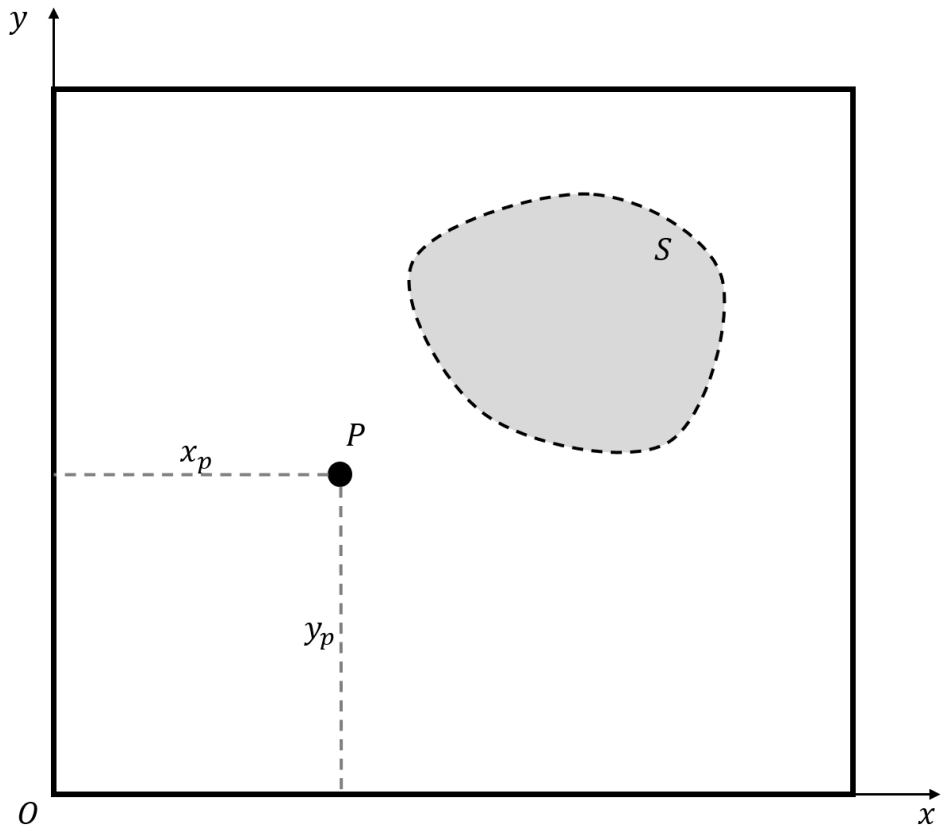


Figure 2.3: The coordinate system of RUDY.

$$WA_S = \iint_S RUDY(x,y)dS \tag{2.1}$$

where WA_S is the estimated wire area in region S. A large wire area in a region indicates that the region is congested.

In detail, given a placed netlist containing n nets, RUDY is defined per net. d_n refers to the wire density of a single net n. It is the ratio of the wire area of net n, which is referred to as WA_n, and the net area of net n, which is referred to as NA_n:

$$d_n = \frac{WA_n}{NA_n} \tag{2.2}$$

Given a multi-pin net n, algorithms like Half Perimeter Wirelength (HPWL) [56] or Rectilinear Steiner Minimal Tree (RSMT) [31] are performed to obtain the locations of wires. Then, the wire area can be computed by the total wire length of net n, denoted by L_n and the wire width of net n, denoted by p_n:

$$WA_n = L_n \cdot p_n \tag{2.3}$$

NA_n, the net area of net n, is the area of the enclosing rectangle of the net n, which is denoted by E_n, and is a rectangle that includes all the pins of net n. Let (x_n, y_n) refer to the coordinate of the lower left corner of E_n. Let w_n refer to the width of E_n, h_n refer to the height of E_n. The coordinates of the four corners of E_n are $(x_n, y_n), (x_n + w_n, y_n), (x_n, y_n + h_n), (x_n + w_n, y_n + h_n)$, respectively. Suppose net n has k pins: $p_1, \cdots, p_k$ with coordinates being $(x_{p_1}, y_{p_1}), \cdots, (x_{p_k}, y_{p_k})$, respectively. In this case, the lower left corner of E_n (x_n, y_n) is calculated as follows:

$$x_n = \min_{i=1,\cdots,k} x_{p_i} \tag{2.4}$$

$$y_n = \min_{i=1,\cdots,k} y_{p_i} \tag{2.5}$$

The width w_n and the height h_n of enclosing rectangle can be computed by:

$$w_n = \max_{i=1,\cdots,k} x_{p_i} - \min_{i=1,\cdots,k} x_{p_i} \tag{2.6}$$

$$h_n = \max_{i=1,\cdots,k} y_{p_i} - \min_{i=1,\cdots,k} y_{p_i} \tag{2.7}$$

Thus the net area NA_n is defined by:

$$NA_n = w_n \cdot h_n \tag{2.8}$$

Figure 2.4 gives an example of a five-pin net n. The area within the dash lines are metal wires that are obtained by RSMT algorithm. The width of the wire is p_n. The total wire length L_n is the total length of the solid black lines. The grey rectangle that covers the pins and wires is the enclosing rectangle of the net.

To calculate RUDY, a rectangle function $R(x, y; x_{ll}, y_{ll}, w, h)$ is needed. Given a rectangle with the coordinate of its lower left corner being (x_{ll}, y_{ll}), and its width

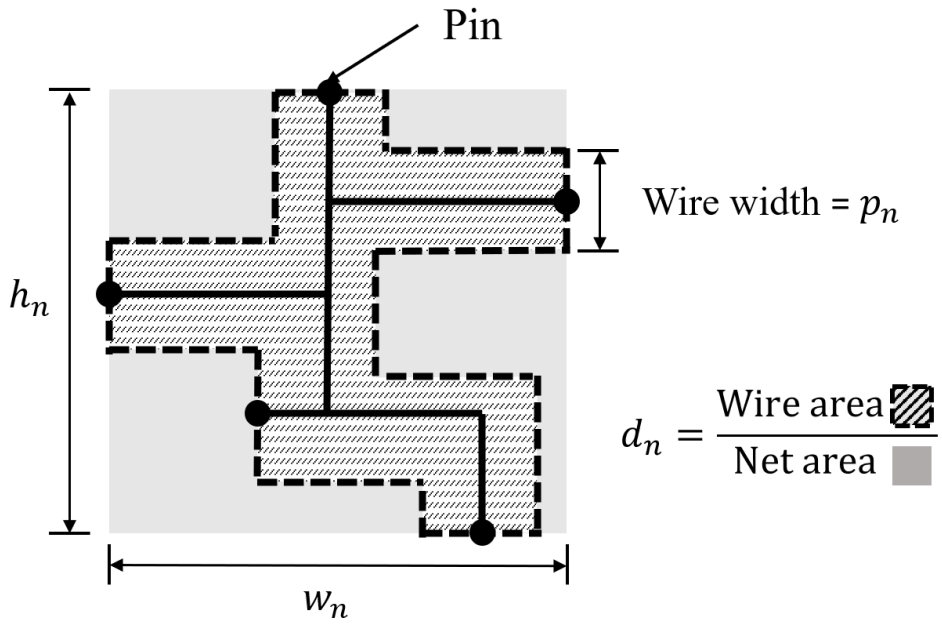


Figure 2.4: A five-pin net and the estimation of RUDY algorithm.

and height being w and h respectively. $R(x,y;x_{ll},y_{ll},w,h)$ represents whether coordinate (x,y) is in the given rectangle, and can be computed as follows:

$$R(x,y;x_{ll},y_{ll},w,h) = \begin{cases} 1 & \text{if } 0 \le x - x_{ll} \le w \text{ and } 0 \le y - y_{ll} \le h \\ 0 & \text{else} \end{cases} \tag{2.9}$$

Finally, the estimation of the wire density of all N nets at (x,y) using RUDY is calculated by the rectangle functions of all nets, weighted by the wire density d_n of each net:

$$RUDY(x,y) = \sum_{n=1}^{N} d_n \cdot R(x,y;x_n,y_n,w_n,h_n) \tag{2.10}$$

RUDY is widely used in placers and routers in the past few decades for its fast speed and low computational resource requirement. However, it is not accurate because it assumes that all the wires of a net are inside its enclosing rectangle, which is true only when the wire takes the shortest path between two pins. It is

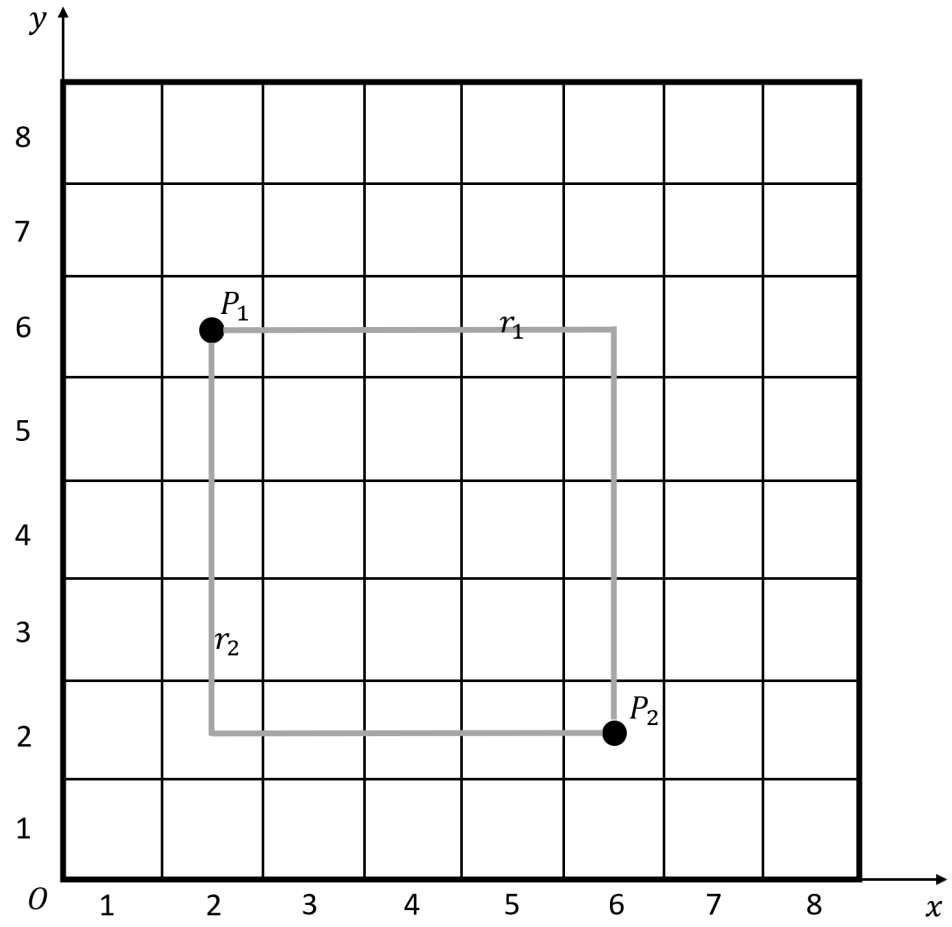


Figure 2.5: The coordinate system of probabilistic-based estimation.

common that detours need to be taken because of the blockage of other pins and wires. Such detours may go outside the enclosing rectangle and therefore influence the accuracy of RUDY.

2.2.2 Probabilistic-Based Estimation

Lou *et al.* [46] proposed a probabilistic-based global routing congestion estimation algorithm that predicts the number of wires in each G-cell. Given a design with $M \times N$ G-cells, a coordinate system can be built so that each tile has its unique coordinate. The coordinate of the lower left G-cell is $(1,1)$, and the coordinate of

the upper right G-cell is (M,N). For example, Figure 2.5 shows a design with 8×8 G-cells. There are two pins, P_1 and P_2, in the design. The coordinates of G-cells that contains P_1 and P_2 are $(2,6)$ and $(6,2)$, respectively.

Different from RUDY which determines the exact path of a net, probabilistic-based estimation first break each net into multiple two-pin nets, and route each two-pin nets sequentially. For each two-pin net, a pattern routing is performed, where multiple possible routes connecting the two pins can be chosen from. The estimator does not choose a specific route but assign equal probability to each route. Suppose net n has k_n possible routes, the probability of each route is $\frac{1}{k_n}$. For example, as shown in Figure 2.5, there are two possible L-shape routes, r_1 and r_2, connecting P_1 and P_2. Therefore, the probability of each of r_1 and r_2 is $\frac{1}{2}$.

Suppose there are N routes going through a G-cell, of which the coordinate is (x,y), The estimation of the number of wires in a G-Cell at coordinate (x,y), which is denoted by $U(x,y)$, can be computed by the sum of probabilities of all N routes that go through G-cell (x,y).

$$U(x,y) = \sum_{n=1}^{N} \frac{1}{k_n} \tag{2.11}$$

where $\frac{1}{k_n}$ is the possibility of net n.

The drawback of probabilistic-based estimation is that the pattern routing limits the accuracy of estimation. In pattern routing, the possible routes of a net are derived from several preset patterns. Although J. Westra *et al.* [68] considered more patterns than [46], it is still insufficient to describe all possible routes of a net in pattern routing.

2.2.3 ML-Based Estimation

Recently, efforts have been put into estimating routing congestion using supervised ML. In supervised learning, a model is trained using a labeled dataset, attempting to predict the routing congestion accurately. The labeling of the dataset requires to route the placed netlists in the dataset using a router. Once the training is finished, the model is able to predict the congestion of unknown placed netlists without routing. Usually, a feature extraction algorithm is required to collect useful infor-

mation called “features”, such as the locations of pins, design rules, *etc.*, from a placed netlist. The features are mapped into tensors as the input of the prediction model.

Qi *et al.* [54, 74] modeled the congestion estimation problem into a regression problem and use Multivariate Adaptive Regression Splines (MARS), a commonly-used ML model, to solve it. In this approach, the layout of placement and global routing is first divided into multiple tiles, on which a feature extraction algorithm is applied to convert each tile into a vector. The training data is fed into MARS, and the routing congestion prediction result, which describes the detailed routing congestion level of a tile, is obtained from MARS output. The reduction of wire-shorts and detailed routing runtime can be achieved by applying this model to global routing. The disadvantage of MARS is that the input vectors only describe the features of a single tile, and the global information is not included. J-Net [41] is a customized CNN that predicts the congestion of detailed routing. It abstracts the pins and macros density in placement results into image data, which is taken as the input of J-Net (an extension of U-Net [57] architecture). The network output is a heatmap representing the locations where detailed routing congestion may occur. The image input keeps the global information and the spatial relationship of different regions in the design. PROS [16] takes advantages of FCN [45] to predict routing congestion from global placement results. One advantage of FCN is that it does not have fully-connected layers, where the image input is disarrayed and flattened into vectors, resulting in the loss of spatial information. Alawieh *et al.* [4] transfer the routing congestion problem in large-scale Field-programmable Gate Arrays (FPGAs) to an image-to-image problem, and then uses a conditional Generative Adversarial Network (GAN) [19, 49] to solve it. This thesis is inspired by MEDUSA [73], a FCN that predicts the wire congestion of global routing based on a placed netlist. MEDUSA maps the placed netlist into a multi-channel image and converts the routing congestion into an image generation problem. The prediction result is an image that describe the number of wires in each region. The advantage of MEDUSA is that the congestion prediction is applied to a global router, and successfully improved the performance of global routing.

Although the Neural Network (NN)-based approaches usually take global information into consideration, most previous works are 2D, which implies they do

not predict on which metal layer the congestion is located. 2D prediction is easier than 3D prediction, as the prediction location is less precise. However, it would guide the router better if the prediction tells the 3D locations of congestion.

2.3 Wire-short Prediction

In addition to forecasting routing congestion, it is also important to predict locations of wire-shorts so that routability optimization engines can be applied to fix them. Given a physical design, wire-short prediction predicts the number and locations of wire-shorts before detailed routing. Recent studies have paid attention to wire-short prediction due to the successful application of ML in EDA. Some wire-short predictors require global routing results, while others perform prediction directly from a placed netlist.

Some works, *e.g.,* [62, 63], adopt Multilayer Perceptron (MLP) to predict the locations of wire-shorts from a placed netlist. The layout of a design is divided into many tiles, to which a feature extraction algorithm is applied to collect the information of tiles, such as number of pins, area of blockages, *etc.*, and convert it to a vector for each tile. The vector is the input of a MLP, which produces a binary output that describes the existence of wire-shorts in that tile. The output can be either "0", which corresponds to the non-existence of wire-shorts, or "1", which indicates the existence of the wire-shorts in the target tile. Therefore, the wire-short prediction is cast as a binary classification problem. Such prediction is fast due to the small size of MLP. However, it lacks global information as each vector only describes a single tile. RouteNet [69] is the first work to employ a CNN for wire-short prediction. A feature extraction algorithm is also applied, but the outputs of feature extraction are images that describe the placement and global routing information, which will be taken as the input of CNNs. RouteNet consists of two CNNs, the first CNN takes a placement as input and predicts the number of wire-shorts in a design, while the second CNN takes the global routing result as input and predicts the locations of wire-shorts. One disadvantage of RouteNet is that the global routing needs to be done to obtain the locations of wire-shorts, significantly increasing the time needed for wire-short prediction. Moreover, it is also helpful to move the wire-short prediction to an earlier stage (after placement)

as global routing is not needed for prediction.

2.4 Summary of Previous Routability Prediction Approaches

Many efforts have been put into routing congestion prediction. RUDY predicts the density of wires on the chip layout. Probabilistic-based approaches predict the number of wires in each G-cell. However, both RUDY and probabilistic-based approaches are based on assumptions that are not accurate: RUDY assumes that all wires take the shortest path, while probabilistic-based approaches assume all wires have certain patterns.

Recently, ML techniques are applied in routing congestion estimation, and progress has been achieved. However, a common drawback of most previous efforts on congestion estimation is that their predictions are 2D.

In addition, wire-short prediction attracts lots of attention recently. However, trial global routing fails to predict wire-shorts accurately due to the complexity of design rules. Some MLP-based approaches are fast but not accurate as the global information is not considered. While some CNN-based approaches require global routing result as an input, which requires extra runtime.

In light of the above-mentioned problems, this thesis investigates the feasibility of ML-based routing congestion and wire-short prediction methods that are fast and accurate compared to previous approaches, such as trial global routing, RUDY, and probabilistic-based methods.

Chapter 3

Routability-driven Global Routing with 3D Congestion Estimation Using a Customized Neural Network

3.1 Introduction

Placement and Routing (P&R) is a key stage of the physical design of ICs. The computational complexity of P&R keeps growing exponentially with the continuous scaling up of designs and scaling down of technology nodes. By virtue of the inherent problem complexity, current P&R methodologies tend to be partitioned in several sequential stages applied iteratively, with each stage based on advanced optimization algorithms. For routing, the common methodology is to break down the task by having a stage of global routing followed by a detail routing. Our work presented here is focused on global routing algorithm improvements.

Global routing algorithms take a given potential design placement topology as input and attempt to find a general routing solution given a design's netlist and a course grid of potential routing channels with given wire (track) capacities. The main bottleneck that global routing algorithms face is caused by routing congestion

which occurs when a routing solution attempts to route too many nets (overflow) in a region only able to accommodate fewer nets. Apart from congestion, other so-called DRVs turn out to nullify potential global routing solutions as well. Academic routers such as NCTU-gr [44] and NTHU-Route [10] have used heuristic algorithms [26, 27, 34, 43, 70] to predict or estimate congestion or other DRVs prior or during the global routing process. On the other hand, so-called *probabilistic* congestion estimation algorithms have been proposed to bring greater improvements, mostly in runtime reduction. Examples include the works reported in RUDY [59] and CU-GR[42]. These approaches suffer from limitations of scalability in that they cannot automatically account for the consistently growing number of design rules and constraints associated with ever changing technology nodes.

In light of the above mentioned challenges, more recently, some research efforts have turned to machine learning-based approaches to form a basis for congestion or DRVs estimation algorithms. RouteNet [69] uses a CNN for DRV prediction. Other similar works include those reported in [30, 41, 72]. Given that modern ICs implemented in advanced technology nodes are comprised of a large number (typically > 10) of metal layers, global routing algorithms able to handle 3D routing and multiple metal layers with vias forming bridges between layers, would be able to find global optimal routing solutions faster. However, a common denominator of these congestion estimation methods is that they address 2D routing, which assumes that all wires are on the same metal layer. Our work is focused on addressing this issue. We propose a 3D global routing congestion estimation methodology based on a CNN customized for the task at hand. We refer to our novel CNN as GlobalNet. Figure 3.1 shows the workflow of GlobalNet. After placement stage, GlobalNet predicts the number of wires and vias in each region. The results, called congestion heatmaps, are used to improve the performance of our proposed global router: UBC-GR. A key novel feature of GlobalNet is that it not only identifies a state of congestion, but also provides a measure of congestion severity. In order to demonstrate the overall potential impact of using GlobalNet to improve global routing algorithms, we replaced the CU-GR [42] optimization cost function by a cost function informed by GlobalNet and refer to the new global routing algorithm as UBC-GR. We obtained promising results on benchmark circuits and report them here.

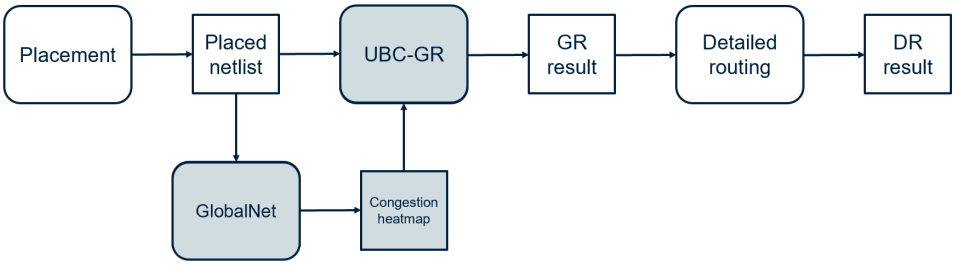


Figure 3.1: Workflow of GlobalNet.

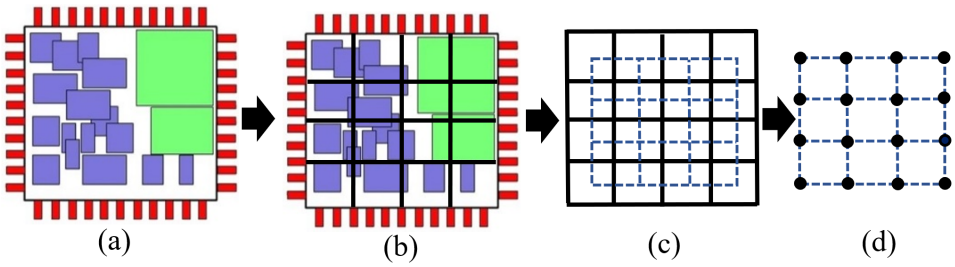


Figure 3.2: An example of transition from physical design to grid graph.

The remainder of this chapter is organized as follows. Section 3.2 presents background and preliminaries for describing our proposed global routing and congestion estimation methodology. Section 3.3 describes our proposed global routing methodology in detail. This is followed by Section 3.4 where we report experimental results. Finally, Section 3.5 concludes.

3.2 Preliminary

The usual method used to perform 3D global routing is to break up the IC into multiple planes (2D layers) and to subdivide each of the planes into a grid of regions denoted as G-cells that result from running virtual horizontal and vertical lines across the entire surface. Fig. 3.2 shows how the virtual horizontal and vertical lines (black lines) form a grid graph: (a) shows a 2D IC design; the black lines in (b) are virtual horizontal and vertical lines that partition the design; each tile in Fig. 3.2 (b) is a G-cell.

Following the creation of such grids for each layer, a graph $G(V,E)$ can be

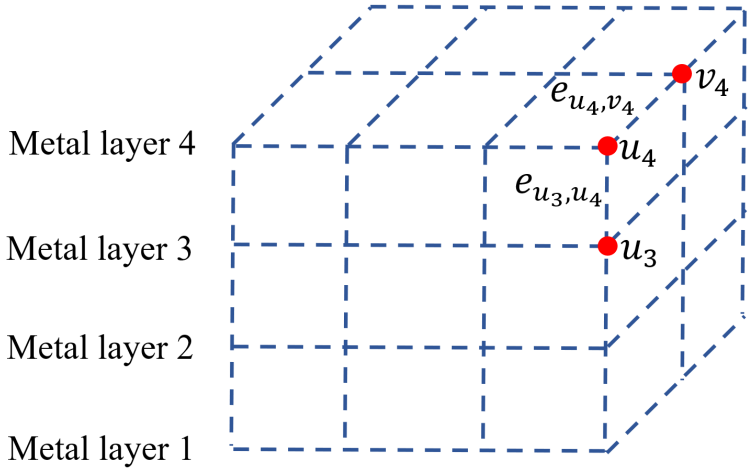


Figure 3.3: Grid graph of a 3D physical design with 4 layers.

constructed where each of the G-cells can be assumed to form a vertex $(v \in V)$ and these vertices can be assumed to be connected through edges $(e \in E)$. Fig. 3.2 (d) shows a grid graph of a 2D design, the black points are vertices and the blue dashed lines are edges interconnecting them. In 3D designs, we refer to v_l as a vertex on the lth metal layer. The edges interconnecting vertices on the same layer are referred to as *wire edges* while edges connecting vertices on different adjacent layers are referred to as *via edges*. Fig. 3.3 shows a grid graph of a 3D design with 4 layers, where u_4 and v_4 are two adjacent vertices on layer 4, and u_3 is a vertex adjacent to u_4 but on layer 3. Edge e_{u_4,v_4} is a wire edge interconnecting u_4 and v_4, and edge e_{u_3,u_4} is a via edge interconnecting u_3 and u_4.

We use the following four attributes of a grid graph: *capacity*, *demand*, *resource*, and *overflow*. A brief definition is given next:

- *Capacity:* Let e_{u_l,v_l} be the wire edge between two adjacent vertices u_l and v_l on the lth layer; and $e_{u_l,u_{l+1}}$ be the via edge between two vertices u_l and u_{l+1} with the same 2D coordinates but on the lth and $l+1$th layer respectively. The wire capacity of e_{u_l,v_l} denoted by $c_w(u_l,v_l)$ is the maximum number of nets that can utilize this edge. Similarly, the via capacity of $e_{u_l,u_{l+1}}$ denoted by $c_v(u_l,u_{l+1})$ is the maximum number of vias that can go through this edge.

- *Demand:* The wire demand, denoted by $d_w(u_l, v_l)$, is the number of wires that are routed through wire edge e_{u_l,v_l}, while the via demand, denoted by $d_v(u_l, u_{l+1})$, is the number of vias that utilize via edge $e_{u_l,u_{l+1}}$.

- *Resource:* The resource of a wire edge e_{u_l,v_l}, denoted by $r_w(u_l, v_l)$, is:

$$r_w(u_l, v_l) = c_w(u_l, v_l) - d_w(u_l, v_l) \tag{3.1}$$

  The resource of a via edge $e_{u_l,u_{l+1}}$ denoted by $r_v(u_l, u_{l+1})$ is similar to that of wire edge, except that the variable pair consists of adjacent vertices from different layers:

$$r_v(u_l, u_{l+1}) = c_v(u_l, u_{l+1}) - d_v(u_l, v_{l+1}) \tag{3.2}$$

- *Overflow:* The overflow is used as one of the metrics to evaluate the routing results. The overflow of a wire edge e_{u_l,v_l} denoted by $o_w(u_l, v_l)$ is computed as follows:

$$o_w(u_l, v_l) = \begin{cases} d_w(u_l, v_l) & \\ \quad -c_w(u_l, v_l) & \text{if } d_w(u_l, v_l) > c_w(u_l, v_l) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

  The overflow of a via edge $e_{u_l,u_{l+1}}$, denoted by $o_v(u_l, u_{l+1})$ is computed by:

$$o_v(u_l, u_{l+1}) = \begin{cases} d_v(u_l, u_{l+1}) & \text{if } d_v(u_l, u_{l+1}) > \\ \quad -c_v(u_l, u_{l+1}) & \quad c_v(u_l, u_{l+1}) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

3.3 Proposed Global Routing Methodology

Fig. 3.4 shows our proposed methodology based on GlobalNet, UBC-GR, and a design feature extraction algorithm that forms hyper-images from the placement information. The output of the GlobalNet block is a congestion heatmap, which

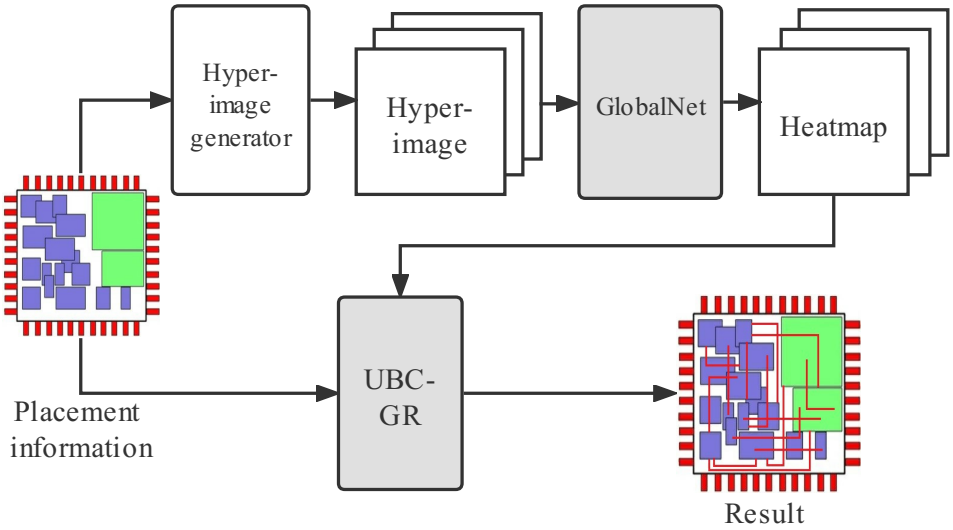


Figure 3.4: Flowchart for our GR methodology based on GlobalNet and UBC-GR.

describes the estimated wire and via demand of each edge of the graph G. The congestion heatmap together with the placement information are used as inputs to UBC-GR.

Different features are used to describe each G-cell of a design. We refer to *pins* as inputs and outputs of standard cells (such as gates and flip-flops) within a design. A *net* is defined as a set of connections between pins that should be electrically connected. If all pins that a net connects are in the same G-cell, this net is referred to as a *local net*. If a net connects at least two pins that are located in different G-cells, it is referred to as a *global net*. As shown in Fig. 3.5, the blue line represents a local net while the orange line represents a global net.

We use the following feature definitions:

- *NP:* number of pins in a G-cell.
- *NNP:* number of neighborhood pins, the average of the number of pins in a G-cell and its 8 surrounding G-cells.
- *ND:* net density, which is equal to the area of wires in a tile. . It is computed by RUDY algorithm [59] as described in Section 2.2.1.

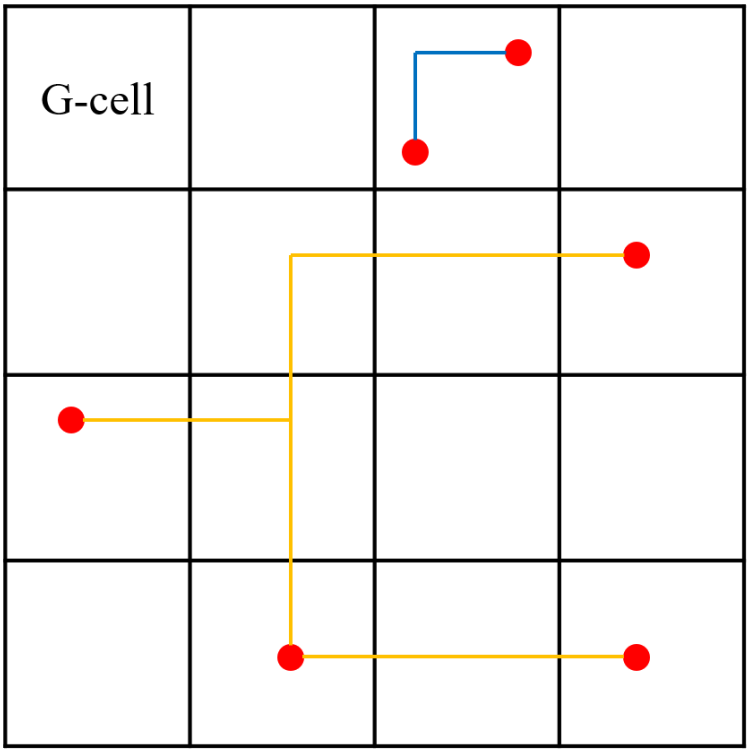


Figure 3.5: An example of pin and net: the red points are pins; the blue line is a local net; and the orange line is a global net.

- *NLN:* number of local nets. A local net is a net that have all its pins in a G-cell. An example of a local net is shown in Figure 3.5.
- *NGN:* number of global nets. A global net is a net that has at least one pin in a G-cell and at least one pin outside that G-cell. An example of a global net is shown in Figure 3.5.
- *WC:* wire capacity, which refers to the maximum number of wires that can be routed through a wire edge without causing an overflow.
- *VC:* via capacity, which refers to the maximum number of vias that can be used in a via edge without causing an overflow.

Unlike standard images which have three channels reserved for the colors red,

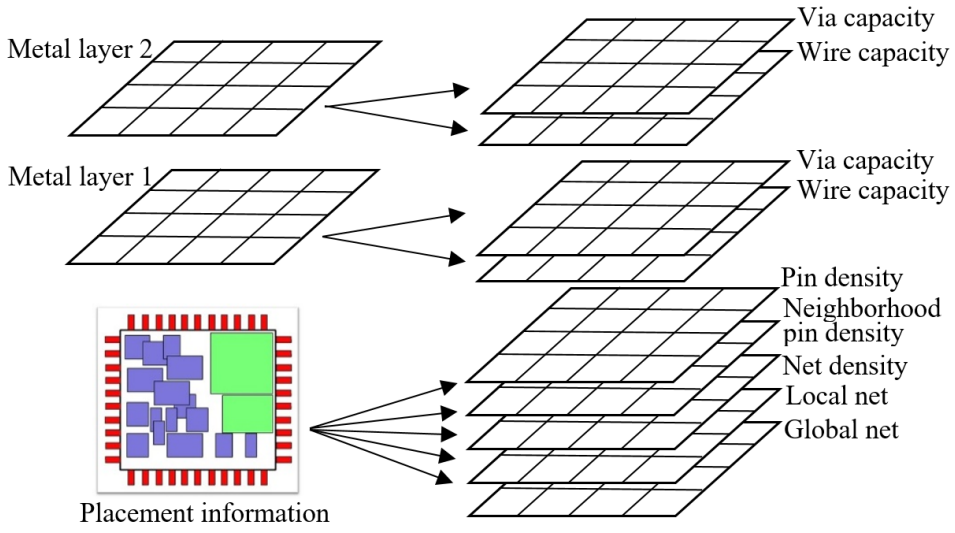


Figure 3.6: An example of how features are transformed from a design (left) to a Hyper-image (right).

green, and blue, we propose *hyper-images* which have no upper limit in terms of the number of channels. Fig. 4.5 shows how a design is transformed to a hyper-image. Each channel of a hyper-image corresponds to a feature, and each pixel in a hyper-image represents the value of a feature of a vertex (G-cell) or an edge. Therefore, the resolution of the hyper-image is the same as the resolution of the graph. Features such as wire capacity and via capacity are categorized as *layer features*. Each metal layer has its own layer features. For example, a design with nine layers has nine wire capacity channels with one for each layer. Other features, *e.g.,* pin density, net density, and neighborhood pin density are called *design features*. Each design has one specific channel for each design feature regardless of the number of metal layers.

From 3.4, a design's hyper-image is fed into GlobalNet which produces a congestion heatmap (also in the form of a hyper-image), with the same resolution as the graph of the design. The congestion heatmap has two kinds of information stored in separate channels for each layer: wire congestion heatmap; and via congestion heatmap. The value of each pixel in a wire/via congestion heatmap represents the predicted number of wires/vias demand of the corresponding edge.

3.3.1 GlobalNet

The network architecture of the GlobalNet is shown in Table 3.1. GlobalNet has 11 layers. The input window size of GlobalNet is 64x64xn, where n is the number of channels of the hyper-image. Inputs are passed down to 4 convolution-and-pooling-layer combinations. After the last pooling layer, the hyper-image resolution is downscaled to 4x4 with 256 channels. Another 4 convolution layers is added at the end to converge the 256 channels back to m, the number of channels used for storing congestion information in our case $m = 16$, because there are 8 channels for estimated wire demand and 8 channels for estimated via demand.

The output of GlobalNet forms a congestion heatmap that describes the estimated demand of wires and vias of each edge. Differing from previous works that produce a binary DRV output, GlobalNet produces continuous values, allowing for treating congestion as a regression problem.

The GlobalNet consists of a set of CNNs, which have specific requirements for input and output sizes. The input size of GlobalNet is 64x64 and its output size is 4x4. However, the sizes of physical designs vary from each other, and the sizes of hyper-images and congestion heatmaps also vary. Therefore, there is no guarantee that the length or width of the hyper-image of a random design can be evenly divided by integers 64 and 4. This causes a problem that prevents GlobalNet from predicting all regions of design because the remaining part of the hyper-image cannot be fed into GlobalNet. To overcome this problem, GlobalNet adopts a strategy which we refer to as "windowing", which gives the model the flexibility when dealing with designs of different layout sizes. A windowing example is shown in Fig. 3.7, where each tile represents a pixel in a channel of a hyper-image. As an example, assume the sizes of inputs and outputs are 6x6 and 2x2, respectively. The prediction begins from the top left corner of the design, where the 2x2 blue output window is shown, and a corresponding 6x6 orange input window is formed. If the input window goes out of the boundary, it will still be created with extra padding tiles and the values in those additional tiles are always 0. This is known as zero padding. GlobalNet takes the pixels from the input window and predicts the heatmap values in the output window. Once the prediction of one input window has gone through the model and the output values are obtained, GlobalNet slides

the input and output window by a certain step to a new position until the entire congestion heatmap is covered by output windows. In other words, the windowing technique continues until the predicted congestion values are stored in every pixel of the output congestion heatmap. If the step of sliding is smaller than the output window size, some pixels in the output congestion heatmap may be computed multiple times. GlobalNet takes the average of these repeated computed values as the final output.

Table 3.1: Neural Network Architecture of GlobalNet

input (64 x 64 x n hyper-image)	
conv5 - 32	output size: 64 x 64 x 32
maxpool	output size: 32 x 32 x 32
conv5 - 64	output size: 32 x 32 x 64
maxpool	output size: 16 x 16 x 64
conv5 - 128	output size: 16 x 16 x 128
maxpool	output size: 8 x 8 x 128
conv5 - 256	output size: 8 x 8 x 256
maxpool	output size: 4 x 4 x 256
conv5 - 128	output size: 4 x 4 x 128
conv5 - 64	output size: 4 x 4 x 64
conv5 - 32	output size: 4 x 4 x 32
conv5 - m	output size: 4 x 4 x m

3.3.2 UBC-GR

CU-GR [42] is a state-of-the-art open-source global router which outperformed all other routers in ICCAD19's global routing contest [21]. CU-GR uses a cost function in initial routing stage to drive the search of a global routing solution for a given design. A path of a net is formed by a set P of wire edges and vias edges, and the total cost of the path, $cost(P)$, will be the sum of the costs of all those edges, i.e., as follow:

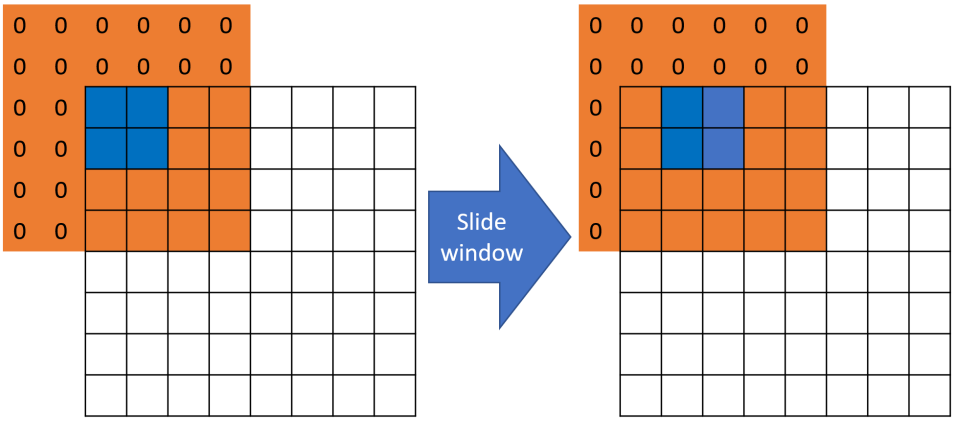


Figure 3.7: An example of windowing, the blue tiles is the output window and the orange tiles is the corresponding input window with zero padding.

$$cost(P) = \sum_{e_{u_l,v_l} \in P} cost_w(u_l, v_l) + \sum_{e_{u_l,u_{l+1}} \in P} cost_v(u_l, u_{l+1}) \tag{3.5}$$

where $cost_w(u_l, v_l)$ is the wire cost of a wire edge e_{u_l,v_l}, $cost_v(u_l, u_{l+1})$ is the via cost of a via edge $e_{u_l,u_{l+1}}$. The wire edge cost is comprised of two parts - wire length cost and expected overflow cost, and is computed as follows:

$$cost_w(u_l, v_l) = wl(u_l, v_l) + eo(u_l, v_l) \times lg(u_l, v_l) \tag{3.6}$$

where $wl(u_l, v_l)$ is the wire length cost, and $eo(u_l, v_l)$ is the expected overflow cost. $eo(u_l, v_l)$ is expressed as:

$$eo(u_l, v_l) = wl(u_l, v_l) \times \frac{d_w(u_l, v_l)}{c_w(u_l, v_l)} \times uoc \tag{3.7}$$

where uoc is a constant that represents a unit overflow cost, and $c_w(u_l, v_l)$ is the wire capacity of the edge between a vertex u_l on lth layer and a vertex v_l on lth layer.

Note that $eo(u_l, v_l)$ is linearly proportional to $d_w(u_l, v_l)$. However, in reality, the relationship between $eo(u_l, v_l)$ and $d_w(u_l, v_l)$ is usually not linear. When the re-

sources are still abundant, there is almost no congestion cost, but the cost increases rapidly as the resources are used up. Therefore, a logistic function, $lg(u_l, v_l)$, is introduced to make the $eo(u_l, v_l)$ non-linear. As illustrated in Figure 3.8, the x-axis is the resource $r_w(u_l, v_l)$. Before multiplying by logistic function $lg(u_l, v_l)$, the estimated overflow $eo(u_l, v_l)$ is linear. Multiplying by $lg(u_l, v_l)$ makes the result non-linear: there is almost no congestion cost when resource is sufficient, but the cost will increase rapidly as the resources are being used up and will keep increasing almost linearly after all the resources are used ($r_w(u_l, v_l) < 0$). This is because when the resources are used up, the expected overflow cost will dominate the congestion cost. The logistic function $lg(u_l, v_l)$ can be computed as follows:

$$lg(u_l, v_l) = (1.0 + exp(\alpha \times r_w(u_l, v_l)))^{-1} \tag{3.8}$$

where α is an adjustable parameter that determines the global router's sensitivity to overflow.

Given two vertical G-cells u_l and u_{l+1} in metal layer l and $l+1$, the via cost of $e_{u_l, u_{l+1}}$ in CU-GR is computed as follows:

$$cost_v(u_l, u_{l+1}) = uvc \times (1 + lg(u_l, u_{l+1})) \tag{3.9}$$

where uvc is a constant that represents a unit via cost. The logistic function $lg(u_l, u_{l+1})$ is the same as in Eq. 3.8, introducing non-linearity to via overflow.

The estimated overflow of a wire edge, $eo(u_l, v_l)$, is based on its actual demand, $d_w(u_l, v_l)$, which is determined by the number of routed nets. Therefore $eo(u_l, v_l)$ cannot account for future overflow arising from yet unrouted nets. This makes CU-GR "shorted-sighted": it always chooses the shortest path that does not cause overflow in current situation but does not consider the potential congestion that may exist in the future. UBC-GR aims to improve the routing performance by taking the future congestion into consideration. The congestion prediction results can be regarded as an estimation of the future congestion. Therefore, UBC-GR addresses this issue by modifying the cost function $cost_w(u_l, v_l)$. The cost function of UBC-GR is designed such that the areas with high congestion prediction results have a higher cost. Therefore, instead of simply choosing the shortest path,

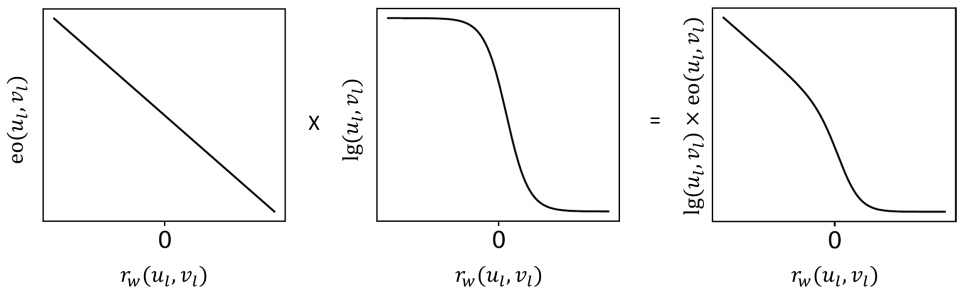


Figure 3.8: A demonstration of logistic function.

UBC-GR can take detours to avoid routing through the congested areas so that the probability of causing overflows is reduced. In detail, the proposed wire cost function $cost'_w(u_l, v_l)$ can be computed as follows:

$$\begin{aligned} cost'_w(u_l, v_l) =& wl(u_l, v_l) + (eo(u_l, v_l) \\ &+ \beta \times \frac{ed_w(u_l, v_l)}{c_w(u_l, v_l)}) \times lg(u_l, v_l) \end{aligned} \tag{3.10}$$

where $wl(u_l, v_l)$, $eo(u_l, v_l)$ and $lg(u_l, v_l)$ are respectively wire length cost, estimated overflow, and logistic function adopted from CU-GR, *uoc* is constant representing the unit overflow cost and $ed_w(u_l, v_l)$ is the estimated demand of wire of edge e_{u_l, v_l}, coming from the proposed GlobalNet. β is is an adjustable parameter that determines the global router's sensitivity to estimated demand.

Also in UBC-GR, the via cost $cost'_v(u_l, u_{l+1})$ of via edge $e'_{u_l, u_{l+1}}$ is computed as follows:

$$\begin{aligned} cost'_v(u_l, u_{l+1}) =& (uvc + \beta \frac{ed_v(u_l, u_{l+1})}{c_v(u_l, u_{l+1})}) \\ &\times (1 + lg(u_l, u_{l+1})) \end{aligned} \tag{3.11}$$

where u_l and u_{l+1} are two vertical G-cells in two different contiguous metal layers. $ed_v(u_l, u_{l+1})$ is the estimated via demand of edge $e_{u_l, u_{l+1}}$ coming from GlobalNet, β is the same parameter used in Eq. 3.10, and *uvc* is the unit via cost. The logistic function $lg(u_l, u_{l+1})$ is the same as Eq. 3.8, introducing non-linearity to via overflow calculation.

3.4 Experimental Results

To compare our proposed algorithm with CU-GR [42], the same benchmarks ISPD 2018 [18] and ISPD 2019 [21] were used. Only benchmarks with 9 metal layers were used while others (benchmark "9t4" and "9t5") were deprecated since a design with a different number of layers requires a structural change to GlobalNet. All benchmarks have the same preferred routing direction assignments for layers. Therefore, GlobalNet is able to handle the layer directions in the experiments.

We implemented GlobalNet using Python and the TensorFlow framework. The proposed UBC-GR was developed based on the open-sourced CU-GR, which was implemented in C++ with the boost library [1]. We used Rsyn [22] to parse LEF/DEF [8] formats. The original CU-GR was used to route all benchmarks as a training set, and the number of wires and vias was recorded to generate congestion heatmaps. The congestion heatmaps generated by CU-GR are the ground truths of the congestion prediction and are used to train and test GlobalNet. Due to the scarcity of benchmark data, we used only one benchmark as a test set and all other benchmarks as a training set. The advantage of such approach was that every benchmark could be tested by re-splitting training and test data, and a better prediction could be obtained because a larger training set was used.

3.4.1 Evaluation Metrics

PCC and NMSE are used to evaluate the accuracy of GlobalNet quantitatively. PCC is a measure of linear correlation between two sets of data. Given a pair of random variables (X,Y), where X and Y are two sets of numbers containing N numbers: $X = (x_1, x_2, \cdots, x_N)$, $Y = (y_1, y_2, \cdots, y_N)$. The formula for PCC is:

$$PCC_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{3.12}$$

where $\text{cov}(X,Y)$ is the covariance of X and Y, σ_x is the standard deviation of X, σ_Y is the standard deviation of Y.

The NMSE of X and Y can be computed as follows:

$$NMSE_{X,Y} = \frac{N\sum_{i=1}^{N}(x_i - y_i)^2}{\sum_{i=1}^{N}(x_i)\sum_{i=1}^{N}(y_i)} \tag{3.13}$$

Table 3.2: PCC and NMSE between the Predicted Congestion Heatmaps (GlobalNet) and the Ground truths (CU-GR)

Design	PCC	NMSE
8t1	0.8779	0.0328
8t2	0.9027	0.0444
8t3	0.9117	0.0302
8t4	0.8385	0.0349
8t5	0.8396	0.0358
8t6	0.8435	0.0532
8t7	0.8490	0.0334
8t8	0.8419	0.0379
8t9	0.8264	0.0547
8t10	0.8278	0.0561
Avg. of 8t	0.8559	0.0413
9t1	0.7998	0.0367
9t2	0.8140	0.0395
9t3	0.8546	0.0345
9t6	0.8208	0.0311
9t7	0.8557	0.0309
9t8	0.8287	0.0419
9t9	0.8298	0.0461
9t10	0.8451	0.0390
Avg. of 9t	0.8311	0.0375
Avg. of All	0.8449	0.0396

Table 3.3: Runtime of GlobalNet (unit: second)

Design	Runtime
8t1	10.21
8t2	7.93
8t3	49.75
8t4	132.30
8t5	152.71
8t6	136.24
8t7	176.43
8t8	310.62
8t9	324.74
8t10	316.32
9t1	8.78
9t2	25.06
9t3	6.48
9t6	257.41
9t7	366.27
9t8	298.14
9t9	687.09
9t10	679.42
Avg.	219.22

Table 3.4: Impact of Different Features on Congestion Estimation using PCC

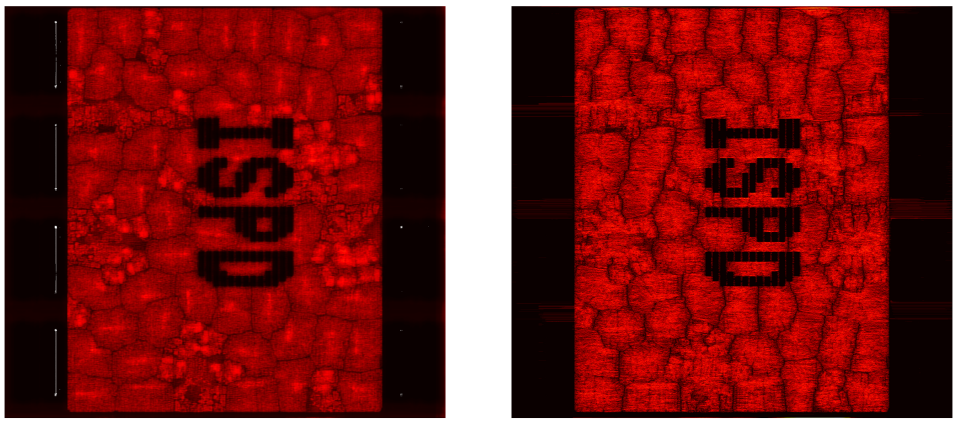
Features	PCC	Norm.
NP + NNP + ND + LN + GN + WC + VC	0.8377	1
NP + NNP + LN + GN + WC + VC	0.8405	**1.0033**
NP + NNP + ND + WC + VC	0.8118	0.9691
NP + ND + LN + GN + WC + VC	0.8331	0.9945

PCC describes the correlations of two variables, and NMSE describes the difference of two variables. Therefore, PCC and NMSE together can be used to describe the similarity of two variables X and Y. In our experiments, PCC and NMSE are used to evaluate the similarity of the GlobalNet results and the CU-GR results. Because the CU-GR results are considered as the ground truth, the similarity indicts the prediction accuracy of GlobalNet.

3.4.2 GlobalNet Performance

To the best of our knowledge, this work is the first to do 3D congestion estimation on ISPD benchmarks. Therefore, no baseline algorithms could be found. First, global routing is performed on all benchmarks using CU-GR. And the ground truths of congestion heatmaps are obtained from the CU-GR routing results. Then, GlobalNet is used to get the predicted congestion heatmaps. The heatmaps are flattened into vectors to compute the PCC and NMSE of the predicted congestion heatmap compared to the ground truth.

Table 3.2 shows the PCC and normalized mean squared error (NMSE) of all benchmarks. The high average PCC between the estimated heatmaps and the actual heatmaps of all designs implies that the predicted values were highly correlated to the actual values. Also, a low NMSE [7] implies that the predicted values were close to the actual values. Figure 3.9 shows an example of the prediction result of GlobalNet. On the left is the predicted congestion heatmap of one of the designs in ISPD 2019 benchmarks ("9t9"), on the right is the ground truth. The brightness of each pixel represent the wire usage of a G-Cell on metal layer 1. The two images



Predicted congestion heatmap

Ground truth of congestion heatmap (routed by CU-GR)

Figure 3.9: Congestion prediction result of design "9t9".

have similar overall brightness and patterns, which indicates the high accuracy of GlobalNet.

We investigated the impact of each feature on overall performance by testing different combinations of features and compared their PCC. Table 3.4 is a comparison between different feature combinations. NP, NNP, ND, LN, GN, WC and VC are abbreviations for different features, standing for number of pins, number of neighborhood pins, net density, number of local nets, number of global nets, wire capacity and via capacity respectively. Among all the feature combinations, the one with NP, NNP, LN, GN, WC, and VC had the highest accuracy. This was also the combination used in Table 3.2. The experiment showed that the combination of features may influence overall accuracy. LN and GN are the most important because the PCC decreased the most by taking them out of consideration. ND had some negative impact on the PCC, which implies that having more features does not always imply a better performance.

Moreover, the runtime of GlobalNet is recorded. The runtime includes the generation of hyper-image and the runtime of the neural network. As shown in Table 3.3, GlobalNet can predict most of the benchmarks in 5 minutes, and the average runtime of all benchmarks is 219.22 seconds.

3.4.3 UBC-GR Performance

Three metrics, wire length, number of vias, and number of overflows, were used to evaluate the routing results. A total score, which was the weighed sum of these three metrics, was also used. All the metrics are such that lower values imply better, more desirable results. Table 3.5, 3.6, 3.7, and 3.8 compares the wire length, number of vias, number of overflows and total score of UBC-GR and CU-GR. As shown in Table 3.7, the results of CU-GR were already well optimized with only three benchmarks having overflows. However, compared with CU-GR, UBC-GR further decreased the number of global routing overflow by 15.0%. In addition, the design 9t9 which was unroutable by CU-GR was successfully routed by UBC-GR. Table 3.6 and 3.5 shows that the number of vias were decreased by 3% while wire length is maintained at the same level. Normally, decreasing overflow would increase the wire length and number of vias as the wires need to take detours to avoid congestion. However, in our case, the fact that the wire length did not increase implies that our algorithm successfully predicted the potential congested regions and UBC-GR leveraged this information.

3.5 Conclusion

We proposed GlobalNet, a global routing congestion estimation algorithm based on a customized CNN. To the best of our knowledge, GlobalNet is the first 3D global routing congestion predictor. Based on the CU-GR, we developed UBC-GR that can utilize the 3D congestion information. Experimental results showed an incremental improvement in global routing performance by producing one more routable design and reducing overall number of overflows, total wirelength, and number of vias when GlobalNet and UBC-GR were used in conjunction. Future research directions include the extraction of more routing features, such as the design rule violations (DRVs) in detailed routing, to improve detailed routing.

Table 3.5: Comparison of Global Routing Wire Length (CU-GR vs UBC-GR)

Design	Wire Length	
	CU-GR	UBC-GR
8t1	410620	411227
8t2	7609440	7655090
8t3	8546570	8568780
8t4	25993900	25989600
8t5	27005100	27013400
8t6	34866000	34817300
8t7	63803100	63760500
8t8	64366000	64214200
8t9	53205300	53093400
8t10	66789400	66608300
Avg. of 8t	35259543	35213180
Norm.	1.0000	**0.9987**
9t1	610418	612818
9t2	24170600	24168900
9t3	780120	780180
9t6	64568800	64532700
9t7	118679000	118670000
9t8	181876000	181868000
9t9	274158000	274100000
9t10	270234000	270242000
Avg. of 9t	116884617	116871825
Norm.	1.0000	**0.9999**
Avg. of All	71537354	71505911
Norm.	1.0000	**0.9996**

Table 3.6: Comparison of Number of Vias (CU-GR vs UBC-GR)

Design	# Vias	
	CU-GR	UBC-GR
8t1	27186	26670
8t2	299852	291663
8t3	300763	293077
8t4	628530	618962
8t5	855806	807816
8t6	1294330	1221680
8t7	2110090	1995630
8t8	2175140	2035800
8t9	2179710	2034680
8t10	2311180	2167670
Avg. of 8t	1218259	1149365
Norm.	1.0000	**0.9434**
9t1	33555	32788
9t2	698909	685503
9t3	51158	50748
9t6	1757370	1728260
9t7	3125290	3098860
9t8	5749010	5618190
9t9	9598410	9374090
9t10	8054350	7966460
Avg. of 9t	3633507	3569362
Norm.	1.0000	**0.9823**
Avg. of All	2291702	2224919
Norm.	1.0000	**0.9709(3%↓)**

Table 3.7: Comparison of Number of Global Routing Overflows (CU-GR vs UBC-GR)

Design	# Overflows	
	CU-GR	UBC-GR
8t1	0	0
8t2	0	0
8t3	0	0
8t4	0	0
8t5	0	0
8t6	0	0
8t7	0	0
8t8	0	0
8t9	0	0
8t10	0	0
Avg. of 8t	0	0
Norm.	0	0
9t1	0	0
9t2	528	454
9t3	0	0
9t6	0	0
9t7	0	0
9t8	0	0
9t9	99	0
9t10	620	611
Avg. of 9t	156	133
Norm.	1.0000	**0.8551**
Avg. of All	69.19	59
Norm.	1.0000	**0.8551(15%↓)**

Table 3.8: Comparison of Global Routing Score (CU-GR vs UBC-GR)

Design	Score	
	CU-GR	UBC-GR
8t1	314054	312294
8t2	5004130	4994200
8t3	5476340	5456700
8t4	15511100	15470700
8t5	16925800	16738000
8t6	22610300	22295300
8t7	40341900	39862800
8t8	40883600	40250300
8t9	35321500	34685400
8t10	42639400	41974800
Avg. of 8t	22502812	22204049
Norm.	1.0000	**0.9867**
9t1	439429	437561
9t2	15144700	15053500
9t3	594692	593082
9t6	39313900	39179400
9t7	71840400	71730700
9t8	113934000	113407000
9t9	175522000	174546000
9t10	167644000	167292000
Avg. of 9t	73054140	72779905
Norm.	1.0000	**0.9962**
Avg. of All	44970069	44682208
Norm.	1.0000	**0.9936**

Chapter 4

VioNet: An Iterative Detailed Routing Wire-short Violation Predictor Based on a Convolutional Neural Network

4.1 Introduction

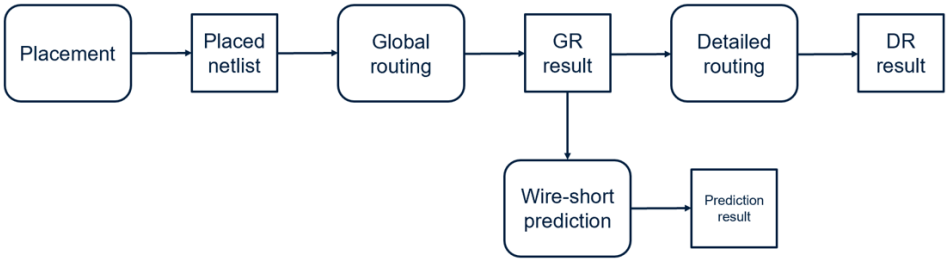
Detailed routing is a challenging stage of the System-on-Chip (SoC) physical design process. The design rule violations (DRVs) that generally occur in the detailed routing process cause the process to require considerable additional time to complete. Detecting and preventing DRVs early in the detailed routing process has been the goal of several research efforts in physical design automation. In this chapter, we describe VioNet, a machine learning framework that predicts detailed routing DRVs directly from a placed netlist.

As the semiconductor process technology advances, the density of cells and metal wires per unit area on a chip increases significantly, and the complexity of physical design grows exponentially due to the increased quantity of pins and limited routing resource for wires. New technology constraints and design rules add to the difficulty of physical design. Detailed routing complexity keeps increasing as

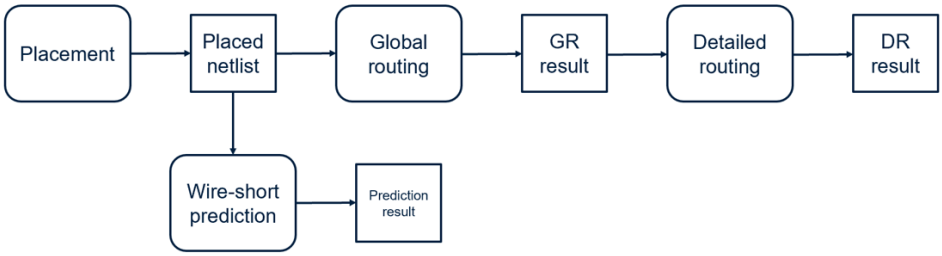
a result. In addition, the risk of routing failure rises due to the greater likelihood of violations that make the design invalid, especially wire-short, which, based on our investigation on ISPD 2019 dataset, is the most common DRV. For the above mentioned reasons, detecting and preventing detailed routing wire-shorts has become a critical issue in physical design as this directly affects the efficiency of detailed routers.

VioNet is a Deep Neural Network (DNN) [39] that predicts the locations of wire-shorts from a placed netlist. Figure 4.1 (a) and (b) show the workflows of two previous wire-short prediction approaches. As shown in Figure 4.1 (a), some previous works [30, 69] utilize the global routing results to guide the wire-short predictor. Intuitively, wire-shorts are more likely to happen in the congested areas after global routing stage. Therefore, global routing results are correlated to wire-shorts and can improve the accuracy of wire-short prediction. However, the disadvantage of such approach is that it has to wait for the global routing to be finished. As shown in Figure 4.1 (b), some other previous works [16, 29, 35, 62, 63] does wire-short prediction using the placement information. Such approach is faster as global routing is not needed. But the prediction may be inaccurate because it does not have access to the global routing results. Figure 4.1 (c) shows the workflow of proposed VioNet. Unlike approaches that predict wire-shorts from global routing results, our proposed VioNet uses a global routing congestion estimation to replace the global router so that the long runtime of global routing can be saved and the wire-short prediction can be moved to an earlier stage. Compared to approaches that does prediction from a placed netlist, VioNet utilizes a congestion heatmap, which is an estimation of global routing results, to improve the routing accuracy.

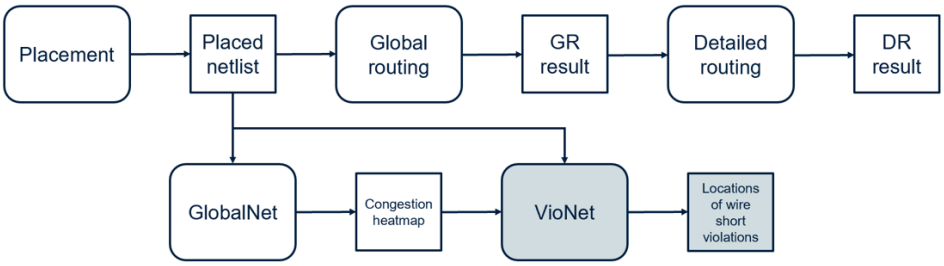
In VioNet, a placed netlist is first partitioned into multiple G-cells. A feature extraction algorithm is applied to convert the placed netlist into image data, so that the spatial information of a placed netlist can be preserved. However, the imbalanced dataset adds to the difficulty of wire-short prediction. The number of G-cells is typically very large but few of them contain wire-shorts [11, 25]. The ratio of the number of G-cells without wire-shorts to the number of G-cells with wire-shorts can be $100:1$ or even higher. Such mismatch of numbers of positives and negatives makes it difficult to achieve a balance between being too sensitive (predicting too many false positives) and too conservative (predicting too many false negatives).



(a) Wire-short prediction based on global routing results



(b) Wire-short prediction based on a placed netlist



(c) VioNet: wire-short prediction based on global routing congestion estimation

Figure 4.1: A comparison of three wire-short prediction approaches.

This problem can be alleviated by lowering the resolution of prediction, as the probability of a tile containing wire-shorts could be increased by making each tile corresponds to a larger die area. However, in such case, the quality of the prediction would be compromised because the locations of predicted wire-shorts would become less precise. Also, the size of training data would decrease because there would be fewer number of tiles in each design. We adopt an iterative strategy [71] to solve the data imbalance problem. First, a low-resolution wire-short prediction is performed, to balance the ratio of number of tiles with wire-shorts (positive) and tiles without wire-shorts (negative). Second, a high-resolution prediction is applied on tiles that have previously been flagged as positive. The negative tiles in low-resolution remain negative and are deprecated so that the high-resolution data prediction can be more balanced by reducing the number of negative tiles.

The rest of this chapter is organized as follows. Section 4.2 briefly introduces the architecture of CNN. Section 4.3 describes VioNet: wire-short predictor in detail. This is followed by Section 4.4 where we report experimental results. Finally, Section 4.6 concludes.

4.2 Convolutional Neural Network (CNN)

CNNs are specialized types of multi-layer artificial neural networks. As shown in Figure 4.2. A typical CNN usually has convolutional layers, pooling layers, and fully connected layers. The input and output of convolutional layers and pooling layers are tensors. The convolutional layers use convolution operation, while the pooling layers reduce the dimensions of data. The tensors are flattened into vectors before being sent to the fully connected layers, which is the same as a traditional multi-layer perceptron neural network (MLP). CNNs are specifically designed to process pixel data and are used in image recognition and processing [37].

Recently, research works [28, 65] have applied CNNs to global routing congestion estimation and detailed routing wire-short prediction. For such applications, a physical design is first partitioned into multiple tiles with the same size using a grid. A feature extraction algorithm is then applied to convert the design to an image, where each pixel represents a feature of a tile. A router is used to find routing solutions as ground truths to train the CNN. The CNN, once trained, is then able to

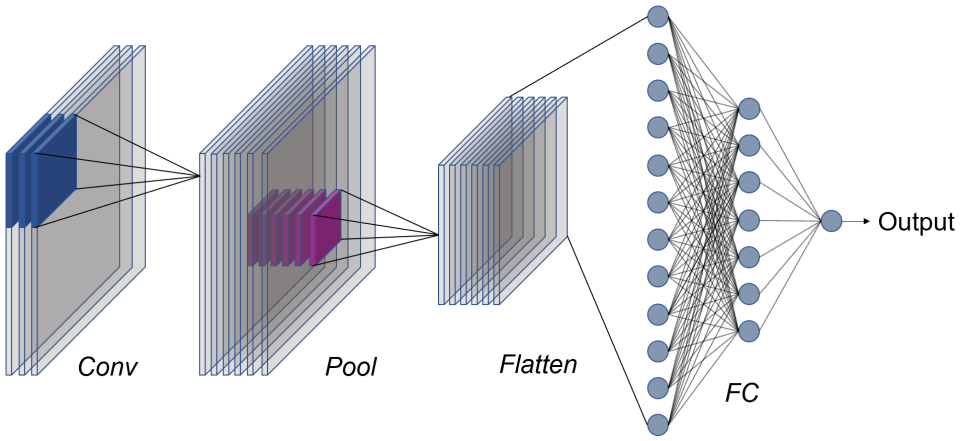


Figure 4.2: Basic structure of a convolutional neural network (CNN).

predict the congestions or wire-shorts without a router.

4.3 Methodology

4.3.1 Problem Formulation

We specifically aim at finding the areas in the physical design that are prone to the occurrence of wire-shorts of detailed routing before routing takes place. The design layout is first divided into $W \times L$ rectangle tiles using a grid. A feature extraction algorithm is used to map the tiles into an image of features, which we call a "feature map".

The input and output of VioNet can be expressed as follows:

Input: $(X^{(i)}, y^{(i)})$, where $X^{(i)} \in \mathbb{R}^{w \times l \times F}$, $y^{(i) \in \{0,1\}}$.

Output: $\hat{y}^{(i)} \in \{0,1\}$.

Each $(X^{(i)}, y^{(i)})$ pair is an instance of the dataset and i is the index of the instance. $X^{(i)}$ is a tensor, which is obtained by applying an input window containing $w \times l, (w < W, l < L)$ tiles on the feature map. F is the number of channels of the feature map. As shown in Figure 4.3, the grey rectangle represents a feature map, while the white block represents an input window. $X^{(i)}$ is the input of the CNN. $y^{(i)}$ is the ground truth to train the CNN, also known as the target output of CNN. $\hat{y}^{(i)}$ is

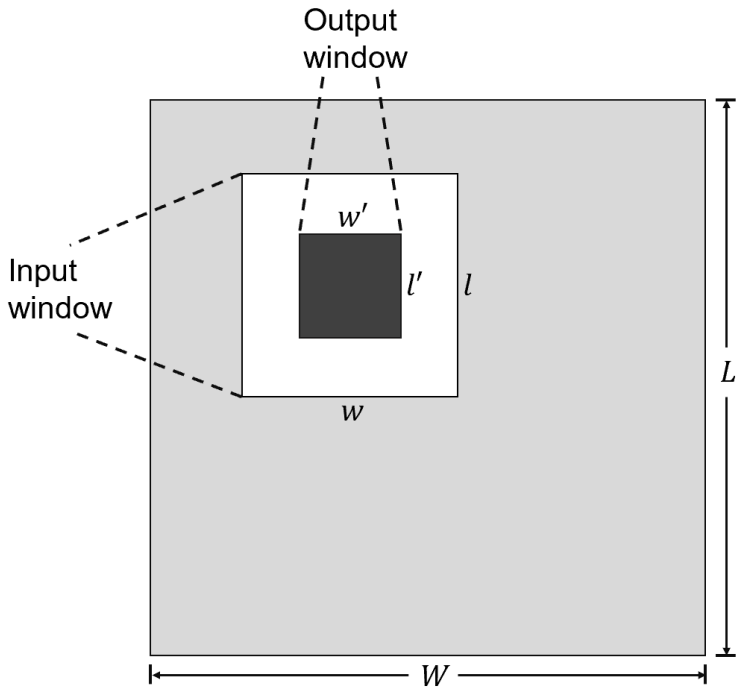


Figure 4.3: An example of an input-output pair for model learning.

the output of CNN. It is the prediction result of $X^{(i)}$, which describe the occurrence of wire-short in the output window containing $w' \times l', (w' < w, l' < l)$ tiles. As shown in Figure 4.3, the black block in the centre of the input window represents an output window. $\hat{y}^{(i)} = 0$ indicates that the output window does not have wire-shorts, and we label such sample as a negative instance, while $y^{(i)} = 1$ indicates the presence of wire-shorts in the output window, and we label such sample a positive instance. The input window is larger than the output window so that the CNN can take the influence of tiles surrounding the output window into consideration.

After a prediction is done, the input and output windows are moved to another location. This procedure is repeated until the output window has covered the entire design.

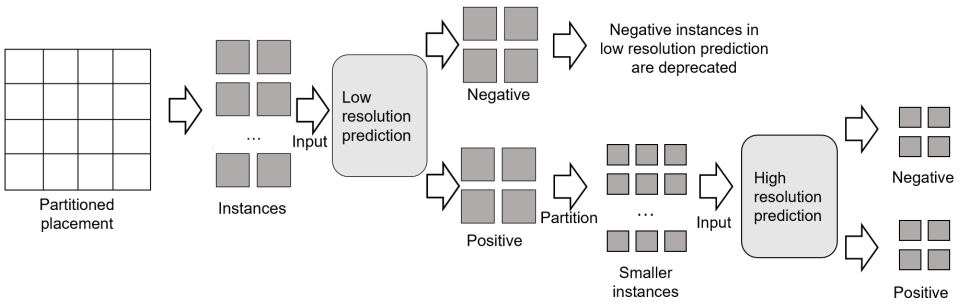


Figure 4.4: Flowchart of iterative prediction.

4.3.2 GlobalNet

VioNet is inspired by GlobalNet [52], a Fully Convolutional Network (FCN) based global routing congestion predictor to estimate the wire and via usage of each tile from a placement netlist. The result of GlobalNet is used as one of the features for VioNet. GlobalNet estimation allows VioNet to be aware of the congested areas in global routing. Compared with other approaches that perform wire-short predictions from global routing solutions, the long runtime of global routing can be saved. Another advantage of moving wire-short to an earlier stage is that the prediction result can be used to improve global routing.

4.3.3 Feature Extraction

We developed a feature extraction algorithm to convert the placement and global routing congestion information into feature maps. As shown in Figure 4.5, on the left is a design layout partitioned into $W \times L$ tiles, on the right is its corresponding feature map. The feature map is essentially a tensor with F channels, where F is also the number of features. Each channel describes a feature of a design. Each channel contains $W \times L$ elements so that each element represents the value of a feature within a given tile. The following features are extracted:

- *NP:* number of pins in a tile.
- *NNP:* number of neighborhood pins, the average of the number of pins in a tile and its 8 surrounding tiles.

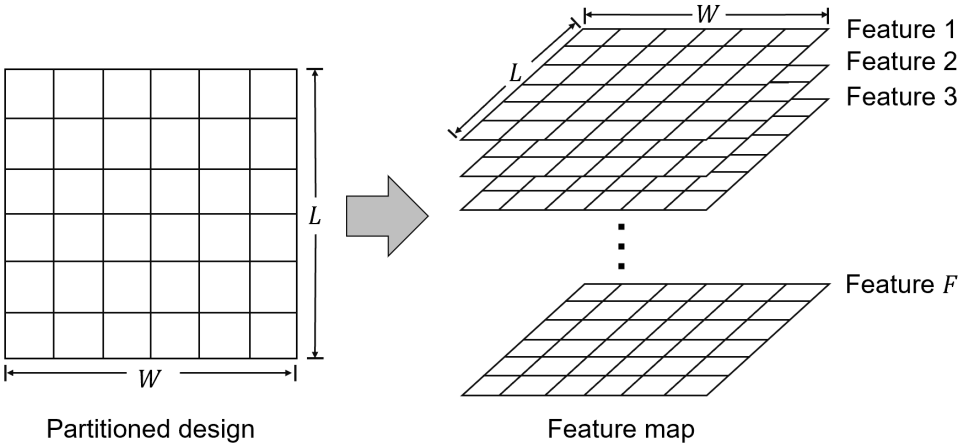


Figure 4.5: Description of feature extraction.

- *ND:* net density, which is equal to the area of wires in a tile. It is computed by RUDY algorithm [59].
- *NLN:* number of local nets. A local net is a net that have all its pins in a tile.
- *NGN:* number of global nets. A global net is a net that have at least one pin in a tile and at least one pin outside that tile.
- *NT:* number of wire tracks in a tile.
- *EU:* estimated usage of wires and vias in a tile computed by GlobalNet [52].

4.3.4 Iterative Prediction

When a design layout is partitioned into multiple tiles, those that contain wire-shorts are typically relatively small in quantity compared with those that do not. Two classes are defined: tiles containing wire-shorts are "positive instances"; and the tiles that do not contain wire-shorts are "negative instances". We also define the ratio of number of positive instances and number of negative instances as "positive to negative ratio (PNR)". The PNR is very low in the ISPD 2019 dataset [21] used in this work. For example, the average PNR is $1 : 342$ if designs are partitioned by the same tile size that the global routing uses. A low PNR can result in a poor

prediction performance because most standard machine learning algorithms, such as CNNs, assume or expect balanced class distributions. Therefore, when these networks are presented with imbalanced datasets, they fail to properly represent the distributive characteristics of the data and result in producing unfavorable accuracy across the classes of the data [25]. This implies that a naive application of a model may focus on learning the characteristics of the abundant observations (negative instances) only, neglecting the examples from the minority class (positive instances).

The PNR can be adjusted to become more balanced by lowering the prediction resolution. The larger area on which each prediction is performed, the higher the possibility that the area contains wire-shorts. For example, for the ISPD 2019 dataset, if each output contains 16×16 tiles instead of a single one, the PNR reduces to $1 : 1.78$ from $1 : 342$. However, if the resolution is too low, the locations of predicted wire-shorts would be too vague to provide useful information for routers. Therefore, a trade-off exists in predicting the wire-short: a high resolution implies a low accuracy, while a lower resolution produces a better accuracy but loses the exact locations of predicted wire-shorts.

Here, we propose an iterative solution, as shown in Figure 4.4, to solve the data imbalance problem without sacrificing the overall prediction resolution. First, the placement information is divided into multiple large tiles and a low-resolution prediction is done to identify the large approximate locations of wire-shorts. The positive instances are then further partitioned into small tiles and sent to a high-resolution prediction so that the detailed location of the wire-shorts can be predicted. In the first low-resolution prediction step, the total number of tiles is smaller and the possibility of each tile containing wire-shorts is larger, resulting in a more balanced PNR. The negative instances from the low-resolution prediction will not be further passed to the high-resolution prediction. Therefore, the number of negative instances in high-resolution prediction is reduced compared with non-iterative approach, and the number of positive instances stays the same, resulting in a more balanced PNR.

4.3.5 Neural Network Structure

VioNet splits the wire-short prediction into two predictions with different resolutions. Therefore, two different CNNs need to be used. The difference between the two predictions is that the size of output window of low-resolution prediction $w'_l \times l'_l$ is larger than that of the high-resolution prediction $w'_h \times l'_h$ ($w'_l > w'_h, l'_l > l'_h$). Different w'_l and l'_l were tested in our experiments to determine the optimal resolution setups.

Both low-resolution and high-resolution prediction adopt the classical structure shown in Table 4.1. Because the labels of these two CNNs are different, they have to be trained separately.

Table 4.1: Neural Network Architecture of VioNet

input (64 x 64 x 7 hyper-image)	
conv5 - 16	output size: 64 x 64 x 16
maxpool	output size: 32 x 32 x 16
conv5 - 32	output size: 32 x 32 x 32
maxpool	output size: 16 x 16 x 32
conv5 - 64	output size: 16 x 16 x 64
maxpool	output size: 8 x 8 x 64
conv5 - 64	output size: 8 x 8 x 64
maxpool	output size: 4 x 4 x 64
FC500	
FC500	
sigmoid	

4.4 Experimental Results

4.4.1 Experimental Setup

The ISPD 2019 dataset [21] was used in our experiments. The ISPD 2019 dataset is one of the latest open sourced routing benchmarks in academia, compared with other benchmarks that were widely used in wire-short prediction, such as ISPD

2015 [17] and ISPD 2018 [18]. The ISPD 2019 dataset is more challenging because it has more complex designs with larger number of nets and pins. Two designs ('9t4' and '9t5') were deprecated because they have different number of layers, thus can not be processed by GlobalNet. The ISPD 2019 dataset includes physical design placements and global routing reports so that both global routers and detailed routers can use this benchmark set. On average, designs have 368897 nets and 1629403 cell pins. The largest design has 895253 nets and 3957499 pins.

We implemented the VioNet using Python and the TensorFlow framework. Rsyn [22] was used to parse LEF/DEF [8] formats. An open sourced global router, CU-GR [42] was used to route the designs to get the real number of wires and vias so that GlobalNet can be trained. We also adopt Dr. CU [12, 13], an open sourced detailed router, to perform detailed routing on the ISPD 2019 dataset. The detailed routing results, including the number and locations of wire-shorts were recorded to label the training data for the training of VioNet. All the instances in the designs were randomly divided into a training set and a test set. 80% of the instances were chosen to form the training set, while the remaining 20% were assigned to the test set. The size of a tile is the same as the size of a G-cell, which is the tile partitioned in global routing stage.

To the best of our knowledge, VioNet is the first work that performs wire-short prediction on ISPD 2019 dataset. The experiments were conducted on a machine with 2.20GHz CPU and an Nvidia GeForce GTX 1080 Ti graphics card.

4.4.2 Performance Metrics

In wire-short prediction, compared to the influence of false negatives, which may result in an invalid design because of the omitted wire-shorts, the influence of false positives is trivial: they usually cause an increase in wire length. Therefore, we care more about the model's ability to predict positive instances. The overall accuracy, which is the percentage of number of true predictions in number of all predictions, is not an effective measure to describe the model's ability to predict the minority class (positive) when the dataset is imbalanced, since it does not distinguish the prediction accuracy of positive instances and the prediction accuracy of negative instances[23]. For example, a model that predicts all instances in the

test set, regardless of the true value, to be the majority instances (negative), will have a high classification accuracy. But such a result is not useful because none of the minority instances (positive) are predicted.

Before introducing the performance metrics, the following terminologies need to be defined:

- **True Positive (TP):** an instance predicted to be positive when the ground truth is positive.
- **False Positive (FP):** an instance predicted to be positive when the ground truth is negative.
- **True Negative (TN):** an instance predicted to be negative when the ground truth is negative.
- **False Negative (FN):** an instance predicted to be negative when the ground truth is positive.

Table 4.2 summarizes the definition of TP, FP, TN and FN.

Table 4.2: An Explanation of TP, TN, FP and FN.

		Prediction Result	
		Positive	Negative
Ground Truth	Positive	TP	FN
	Negative	FP	TN

The following metrics, which are widely used to evaluate the performance of imbalanced classification, are used to evaluate the performance of VioNet:

1) True Positive Rate (TPR): refers to the probability of a positive test, conditioned on truly being positive. TPR can be computed as follows:

$$TPR = \frac{TP}{TP+FN} \tag{4.1}$$

2) True Negative Rate (TNR): refers to the probability of a negative test, con-

ditioned on truly being negative. TPR can be computed as follows:

$$TNR = \frac{TN}{TN + FP} \tag{4.2}$$

3) Accuracy (AC): refers to the percentage of accurate predictions. Accuracy is computed as follows:

$$AC = \frac{TN + TP}{TN + TP + FN + FP} \tag{4.3}$$

4.4.3 Result of the Low-resolution Prediction Stage

At the low-resolution prediction stage, as mentioned in Section 4.3.4, the lower resolution yields a more balanced dataset, and an easier prediction. However, the low-resolution prediction only reports the approximate locations of wire-shorts, and the areas predicted to be positive in low-resolution prediction are sent to high-resolution prediction to determine the more exact locations of the wire-shorts. Therefore, a relatively small output window at the low-resolution prediction stage provides more precise information and may improve the accuracy of the high-resolution prediction.

To investigate the influence of output window size on prediction accuracy, three experiments with different output window sizes were performed. The size of the output window at the low-resolution prediction stage was configured in three respective settings: Setting 1 with $w'_{l1} \times l'_{l1} = 64 \times 64$; Setting 2 with $w'_{l2} \times l'_{l2} = 32 \times 32$; and Setting 3 with $w'_{l3} \times l'_{l3} = 16 \times 16$. Table 4.3, 4.4, 4.5 shows the test results of three settings. As shown in Table 4.3, Setting 1 had the largest output dimension, which implies the lowest resolution and the most balanced dataset. Therefore, the performance was the best among all three experiments in terms of AC, TPR, and TNR. All three experiments using three different settings had a TPR of over 95%, although Setting 3 had a lower AC and TNR due to its relatively small output dimension. The high performance of the low-resolution prediction implies that the wire-short prediction can be very accurate when the PNR is close to 1. However, the good AC of low-resolution prediction is not useful since the locations of predicted wire-shorts are too vague in that case. Thus, a high-resolution

prediction stage needs to be executed based on the low-resolution prediction result by concatenating a secondary CNN to the first one (Figure 4.4).

Table 4.3: Results with low-resolution stage of Setting 1 ($w'_{l1} \times l'_{l1} = 64 \times 64$).

	TP	TN	FP	FN	AC	TPR	TNR
9t1	1	0	0	0	1.00	1.00	-
9t2	10	3	0	1	0.93	0.91	1.00
9t3	3	0	0	0	1.00	1.00	-
9t6	22	37	0	1	0.98	0.96	1.00
9t7	49	21	0	1	0.99	0.98	1.00
9t8	49	21	0	1	0.99	0.98	1.00
9t9	68	25	0	0	1.00	1.00	1.00
9t10	69	26	3	1	0.96	0.99	0.90
Avg.	271	133	3	5	0.98	0.98	0.98

4.4.4 Result of the High-resolution Prediction Stage

Three experiments were also performed with the high-resolution prediction stage. Each of them used as input the corresponding output from the three low-resolution prediction experiments discussed in Section 4.4.3. The window size of the three experiments at the high-resolution prediction stage were the same: $w'_h \times l'_h = 4 \times 4$. We also implemented a CNN based wire-short prediction without iteration by applying a high-resolution prediction directly without low-resolution prediction as the baseline algorithm. The number of TP, TN, FP, FNs were recorded. AC, TPR and TNR were used to evaluate the performances of these prediction results. Only the positive instances of low-resolution prediction are sent to high-resolution prediction. The negative instances in the low-resolution prediction stage are counted as negative predictions in the high-resolution prediction stage. Therefore, the AC, TPR and TNR shown in Table 4.6, 4.7, and 4.8 are the final results of the iterative

Table 4.4: Results with low-resolution stage of Setting 2 ($w'_{l2} \times l'_{l2} = 32 \times 32$).

	TP	TN	FP	FN	AC	TPR	TNR
9t1	1	0	0	1	0.50	0.50	-
9t2	31	24	0	1	0.98	0.97	1.00
9t3	1	5	1	0	0.86	1.00	0.83
9t6	72	96	2	3	0.97	0.96	0.98
9t7	94	100	5	1	0.97	0.99	0.95
9t8	49	21	0	1	0.99	0.98	1.00
9t9	158	108	18	4	0.92	0.98	0.86
9t10	247	104	12	6	0.95	0.98	0.90
Avg.	653	458	38	17	0.95	0.97	0.92

Table 4.5: Results with low-resolution stage of Setting 3 ($w'_{l3} \times l'_{l3} = 16 \times 16$).

	TP	TN	FP	FN	AC	TPR	TNR
9t1	2	5	2	0	0.78	1.00	0.71
9t2	77	86	6	0	0.96	1.00	0.93
9t3	0	12	2	0	0.86	-	0.86
9t6	216	390	16	5	0.97	0.98	0.96
9t7	240	474	127	6	0.84	0.98	0.79
9t8	301	466	384	11	0.66	0.96	0.55
9t9	589	487	412	8	0.72	0.99	0.54
9t10	555	524	376	34	0.72	0.94	0.58
Avg.	1980	2444	1325	64	0.76	0.97	0.65

prediction (low-resolution prediction followed by high-resolution prediction).

As shown in Table 4.6, 4.7, and 4.8, although Setting 1 and 2 had better TPRs in the low-resolution prediction stage compared with Setting 3, the performance of these three settings from the high-resolution prediction stage are similar in terms of AC, TPR and TNR. The reason is that the low-resolution prediction of Setting 3 had a smaller w'_l and l'_l. This implies that the locations of identified wire-shorts are more precise than the other two in the low-resolution prediction stage, which compensates for the low AC of Setting 3. Table 4.10, 4.11, and 4.12 show the detailed prediction results including the number of TP, TN, FP, and FN. As shown in the tables, when the number of instances increases, a reduction of AC, TPR, and TNR can be observed in Setting 2, Setting 3, and non-iterative prediction. This implies that the accuracy of prediction can not be guaranteed in large designs. Setting 1, however, had the most balanced performance for both small and large designs. Therefore, Setting 1 is recommended although all three settings had similar average accuracy.

4.4.5 Comparison with the Non-iterative Prediction

The non-iterative prediction is implemented by applying high-resolution prediction directly, without low-resolution prediction. Table 4.9 shows that all three Settings of iterative approaches outperformed the non-iterative approach in terms of AC, TPR and TNR. Compared with Table 4.8, the AC, TPR and TNR were increased by 11%, 7%, and 11% respectively, when the iterative approach is applied. 4.13 shows the detailed results of non-iterative prediction including the number of TP, TN, FP, and FN.

4.5 Runtime

The runtime of Dr. CU and the proposed VioNet are recorded. The runtime of the proposed VioNet consists of two part: the runtime of the low-resolution prediction and the runtime of the high-resolution prediction. The runtime of GlobalNet is not included. Results in Table 4.14 show that the average runtime of VioNet is 92 times faster than that of Dr. CU on ISPD 2019 benchmarks. The proposed VioNet finished the two benchmarks, 9t9 and 9t10, in less than 3 minutes, while Dr. CU

requires almost 4 hours. For the simple benchmarks like 9t1 and 9t3, VioNet can finish them in a few seconds. Compared with detailed routing that sometimes takes hours, the time needed for VioNet is trivial but the results generated from VioNet include important information about detailed routing (the locations of wire-shorts).

4.6 Conclusion

We developed VioNet, a CNN based algorithm that can predict the locations of detailed routing wire-shorts based on a placed netlist. One advantage of VioNet is that a global routing congestion estimation is used to replace a global router, so that the global routing results can be obtained to improve the accuracy of VioNet in a short time. An iterative strategy was also applied to solve the data imbalance problem. Experimental results showed that compared to the non-iterative approach, the iterative strategy increased the prediction accuracy, true positive rate, and true negative rate of VioNet by 11%, 7%, and 11% respectively. Experimental results showed that VioNet can predict the wire-shorts fast and accurately: on average, 74% of the wire-shorts were predicted, and the runtime of VioNet was 92 times faster than that of Dr. CU.

Table 4.6: AC, TPR, and TNR of iterative prediction with Setting 1 ($w'_{l1} \times l'_{l1} = 64 \times 64$)

Design	AC	TPR	TNR
9t1	0.65	0.89	0.63
9t2	0.54	0.90	0.49
9t3	0.98	0.00	1.00
9t6	0.85	0.96	0.85
9t7	0.86	0.77	0.86
9t8	0.75	0.59	0.76
9t9	0.56	0.77	0.55
9t10	0.69	0.62	0.69
Avg.	0.71	0.73	0.71

Table 4.7: AC, TPR, and TNR of iterative prediction with Setting 2 ($w'_{l2} \times l'_{l2} = 32 \times 32$).

Design	AC	TPR	TNR
9t1	0.89	0.29	0.93
9t2	0.61	0.99	0.58
9t3	0.98	1.00	0.98
9t6	0.79	0.99	0.79
9t7	0.83	0.70	0.83
9t8	0.85	0.40	0.86
9t9	0.60	0.72	0.59
9t10	0.60	0.74	0.59
Avg.	0.71	0.74	0.70

Table 4.8: AC, TPR, and TNR of iterative prediction with Setting 3 ($w'_{l3} \times l'_{l3} = 16 \times 16$).

Design	AC	TPR	TNR
9t1	0.70	1.00	0.68
9t2	0.64	0.99	0.61
9t3	0.93	-	0.93
9t6	0.81	0.97	0.80
9t7	0.86	0.60	0.87
9t8	0.90	0.32	0.92
9t9	0.69	0.61	0.70
9t10	0.52	0.90	0.51
Avg.	0.73	0.72	0.73

Table 4.9: AC, TPR, and TNR of non-iterative prediction.

Design	AC	TPR	TNR
9t1	0.72	0.60	0.73
9t2	0.61	0.73	0.60
9t3	0.93	0.50	0.94
9t6	0.68	0.63	0.68
9t7	0.67	0.54	0.67
9t8	0.68	0.65	0.69
9t9	0.55	0.65	0.55
9t10	0.57	0.67	0.57
Avg.	0.62	0.65	0.62

Table 4.10: Detailed prediction results of iterative prediction with Setting 1 ($w'_{l1} \times l'_{l1} = 64 \times 64$)

Design	# Instances	# Positive	# Negative	TP	TN	FP	FN
9t1	256	19	237	17	150	87	2
9t2	3584	429	3155	386	1540	1615	43
9t3	768	13	755	0	755	0	13
9t6	8704	319	8385	305	7106	1279	14
9t7	15360	224	15136	173	13048	2088	51
9t8	18176	459	17717	271	13450	4267	188
9t9	23808	990	22818	761	12568	10250	229
9t10	25344	992	24352	613	16896	7456	379
Avg.	96000	3445	92555	2526	65513	27042	919

Table 4.11: Detailed prediction results of iterative prediction with Setting 2 ($w'_{l2} \times l'_{l2} = 32 \times 32$).

Design	# Instances	# Positive	# Negative	TP	TN	FP	FN
9t1	128	7	121	2	112	9	5
9t2	3584	322	3262	319	1878	1384	3
9t3	448	1	447	1	439	8	0
9t6	11072	479	10593	472	8324	2269	7
9t7	12800	312	12488	217	10426	2062	95
9t8	18432	468	17964	188	15523	2441	280
9t9	23424	1075	22349	779	13213	9136	296
9t10	23616	1003	22613	740	13344	9269	263
Avg.	93504	3667	89837	2718	63259	26578	949

Table 4.12: Detailed prediction results of iterative prediction with Setting 3 ($w'_{l3} \times l'_{l3} = 16 \times 16$).

Design	# Instances	# Positive	# Negative	TP	TN	FP	FN
9t1	144	9	135	9	92	43	0
9t2	2704	226	2478	224	1512	966	2
9t3	224	0	224	0	208	16	0
9t6	10032	437	9595	424	7707	1888	13
9t7	13552	312	13240	186	11523	1717	126
9t8	18592	479	18113	155	16648	1465	324
9t9	23936	1011	22925	618	15942	6983	393
9t10	23824	971	22853	870	11554	11299	101
Avg.	93008	3445	89563	2486	65186	24377	959

Table 4.13: Detailed prediction results of non-iterative prediction.

Design	# Instances	# Positive	# Negative	TP	TN	FP	FN
9t1	118	5	113	3	82	31	2
9t2	2916	284	2632	208	1578	1054	76
9t3	173	2	171	1	160	11	1
9t6	10160	660	9500	413	6481	3019	247
9t7	13247	392	12855	212	8672	4183	180
9t8	17156	426	16730	277	11468	5262	149
9t9	24014	1055	22959	686	12603	10356	369
9t10	24075	1050	23025	705	13070	9955	345
Avg.	91859	3874	87985	2505	54114	33871	1369

Table 4.14: The runtime comparison of VioNet and Dr. CU (unit: second).

Design	Dr. CU	VioNet
9t1	152	5
9t2	1644	16
9t3	61	3
9t6	2853	60
9t7	6822	80
9t8	9532	55
9t9	13353	155
9t10	13983	154
Norm.	91.66667	1

Chapter 5

Conclusion

5.1 Contributions

This thesis demonstrated the design and implementation of two ML-based frameworks that can improve the routability of global routing and detailed routing for ASICs. The main contributions are as follows:

1. Design and development of GlobalNet, a FCN framework that predicts the 3D routing congestion of global routing. A feature extraction algorithm is used to convert a design to image data, which is taken as the input of the proposed NN. The output of GlobalNet is a congestion heatmap that describes the wire and via usage of the design, from which the overflow can be derived.

   (a) Experiments on ISPD 2018 and ISPD 2019 benchmarks showed that GlobalNet achieved a high average PCC (0.84) and a low NMSE (0.04), which indicated the high accuracy of prediction.

   (b) The impacts of different features were investigated. Experiments were done with different feature combinations. The optimal feature setup was found. Our findings also suggested that using more features does not always mean a better prediction accuracy.

   (c) Development of UBC-GR, which is a global router improved from an open-sourced global router, CU-GR. UBC-GR utilized the congestion

estimation result from GlobalNet to avoid potential overflows. Two benchmarks in ISPD 2019 that were unroutable to CU-GR were successfully routed by UBC-GR. Moreover, a 3% of reduction in the number of vias and a 15% of reduction in the number of overflows were achieved in the ISPD dataset.

2. Investigation of the feasibility of wire-short prediction using CNN. Designed and implemented VioNet, a customized CNN that predicts the number and locations of detailed routing wire-shorts from a placed netlist.

   (a) We used the global routing congestion estimation result from GlobalNet to improve the performance of wire-short prediction. The advantage was that the wire-short predictor could have access to the global routing information while the long runtime of global routing could be avoided, and the wire-short prediction could be moved to an earlier stage.

   (b) A top-down iterative approach was adopted to predict the locations of wire-shorts. A low-resolution prediction was done first to determine the approximate locations of wire-shorts. Then, a high-resolution prediction was performed on the regions that were predicted to contain wire-shorts in the previous low-resolution prediction stage.

   (c) Experiments on ISPD 2019 benchmarks showed that, on average, 74% of the wire short violations were predicted. The top-down iterative strategy also improved the prediction accuracy, true positive rate, and true negative rate were improved by 11%, 7%, and 11%, respectively, compared with the non-iterative approach.

5.2 Future Work

Comparison with Previous Works

Due to the novelty of this work, there is no previous work on 3D global routing congestion estimation or detailed routing wire-short prediction using the ISPD dataset.

But it is still meaningful to compare this work to other works that do 2D prediction on other dataset. In the future, some other neural networks, like MEDUSA [73], RouteNet [69], CongestionNet [35], will be implemented on the ISPD dataset for comparison with the proposed GlobalNet and VioNet.

Further Investigation on the Impact of Congestion Estimation on Routing

The proposed UBC-GR successfully reduced the number of overflows in global routing stage. Some further analyze of the experimental results and investigations on the mechanism behind this improvement may be helpful for future research. For example, the prediction result of GlobalNet is a multi-channel congestion heatmap. But do the predictions on different metal layers have different accuracy? Does prediction of each metal layer have the same contribution to the improvements of routing performance? The lower metal layers usually contain more wires. Intuitively, the prediction of lower metal layers should be more important than that of the higher layers, but such assumption lacks support from experimental data. Some more experiments need to be done to compare the prediction accuracy (PCC and NMSE), and the routing improvement caused by the prediction of each metal layer.

Detailed Routing with Routing Prediction

This thesis proved the feasibility of using congestion estimation to improve the performance of a global router. However, the contributions that congestion estimation and wire-short prediction could make to a detailed router are still unclear.

Placement with Routing Prediction

The experimental results showed that many of the designs in ISPD 2018 and ISPD 2019 benchmarks were unroutable to CU-GR, UBC-GR, and Dr. CU. It would be helpful if the placer could be improved by using the congestion estimation or wire-short prediction results.

Unsupervised Learning based Routing Prediction

This work suffered from the same problem as most previous works on ML in EDA did, which is the lack of training data. Only 18 benchmarks from ISPD 2018

and ISPD 2019 were available. Unsupervised ML can solve this problem as no training data is needed. Some efforts have been put into Reinforcement Learning (RL) based placement and routing. However, the most significant disadvantage that constrains the application of RL is its lengthy runtime. RL would become more practical if techniques that can speed up the training of RL could be proposed.

Bibliography

[1] Boost geometry library. https://www.boost.org/doc/.

[2] I. 20189Contest. Ispd 2019 contest on initial detailed routing., 2019. URL http://www.ispd.cc/contests/19/.

[3] P. Agnew and M. Kelly. The vms algorithm. *IBM System Products Div. Lab., Endicott, NY, Tech Rep. TR01*, 1338, 1970.

[4] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. A. Iyer, and D. Z. Pan. High-definition routing congestion prediction for large-scale fpgas. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 26–31. IEEE, 2020.

[5] Apple. Apple unveils m1 ultra, the world's most powerful chip for a personal computer, 2022. URL https://www.apple.com/newsroom/2022/03/apple-unveils-m1-ultra-the-worlds-most-powerful-chip-for-a-personal-computer/.

[6] X. Bai, Z. Zhu, P. Li, J. Chen, T. Lan, X. Li, J. Yu, W. Zhu, and Y.-W. Chang. Timing-aware fill insertions with design-rule and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[7] A. Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[8] Cadence. Lef/def 5.8 language reference, 2016. URL http://coriolis.lip6.fr/doc/lefdef/lefdefref/LEFSyntax.html.

[9] P. Case, M. Correia, W. Gianopulos, W. Heller, H. Ofek, T. Raymond, R. Simek, and C. Stieglitz. Design automation in ibm. *IBM Journal of Research and Development*, 25(5):631–646, 1981.

[10] Y. Chang, Y. Lee, and T. Wang. Nthu-route 2.0: A fast and stable global router. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 338–343, Nov 2008.

[11] N. V. Chawla, N. Japkowicz, and A. Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.

[12] G. Chen, C.-W. Pui, H. Li, J. Chen, B. Jiang, and E. F. Young. Detailed routing by sparse grid graph and minimum-area-captured path search. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 754–760, 2019.

[13] G. Chen, C.-W. Pui, H. Li, and E. F. Young. Dr. cu: Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(9):1902–1915, 2019.

[14] H.-M. Chen, M. D. F. Wong, H. Zhou, F.-Y. Young, H. H. Yang, and N. Sherwani. *Integrated Floorplanning and Interconnect Planning*, pages 1–18. Springer US, Boston, MA, 2001. ISBN 978-1-4757-3415-7. doi:10.1007/978-1-4757-3415-7_1. URL https://doi.org/10.1007/978-1-4757-3415-7_1.

[15] H.-Y. Chen and Y.-W. Chang. Global and detailed routing. In *Electronic Design Automation*, pages 687–749. Elsevier, 2009.

[16] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Young. Pros: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2020.

[17] I. . Contest. Ispd 2015 blockage-aware detailed routing-driven placement contest., 2015. URL http://www.ispd.cc/contests/15/web/benchmark.html.

[18] I. . Contest. Ispd 2018 contest on initial detailed routing., 2018. URL http://www.ispd.cc/contests/18/.

[19] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[20] W. A. Dees and P. G. Karger. Automated rip-up and reroute techniques. In *19th Design Automation Conference*, pages 432–439. IEEE, 1982.

[21] S. Dolgov, A. Volkov, L. Wang, and B. Xu. 2019 cad contest: Lef/def based global routing. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–4, 2019. doi:10.1109/ICCAD45719.2019.8942107.

[22] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis. Rsyn: An extensible physical synthesis framework. In *Proceedings of the 2017 ACM on International Symposium on Physical Design*, pages 33–40, 2017.

[23] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[24] P. Garrou, J. J.-Q. Lu, and P. Ramm. Three-dimensional integration. *Handbook of Wafer Bonding*, 2012.

[25] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[26] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, and E. F. Y. Young. Ripple: An effective routability-driven placer by iterative cell movement. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 74–79, Nov 2011.

[27] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit designs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 80–84, Nov 2011.

[28] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, et al. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 26(5):1–46, 2021.

[29] Y.-H. Huang, Z. Xie, G.-Q. Fang, T.-C. Yu, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. Routability-driven macro placement with embedded cnn-based prediction model. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 180–185. IEEE, 2019.

[30] W.-T. Hung, J.-Y. Huang, Y.-C. Chou, C.-H. Tsai, and M. Chao. Transforming global routing report into drc violation map with convolutional neural network. In *Proceedings of the 2020 International*

Symposium on Physical Design, ISPD '20, page 57–64, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370912. doi:10.1145/3372780.3375557.

[31] F. K. Hwang. On steiner minimal trees with rectilinear distance. *SIAM journal on Applied Mathematics*, 30(1):104–114, 1976.

[32] D. Jansen et al. *The electronic design automation handbook*. Springer, 2003.

[33] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI physical design: from graph partitioning to timing closure*. Springer Nature, 2022.

[34] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov. A SimPLR method for routability-driven placement. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 67–73, 2011. ISBN 978-1-4577-1398-9.

[35] R. Kirby, S. Godil, R. Roy, and B. Catanzaro. Congestionnet: Routing congestion prediction using deep graph neural networks. In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 217–222. IEEE, 2019.

[36] C. Kison, O. M. Awad, M. Fyrbiak, and C. Paar. Security implications of intentional capacitive crosstalk. *IEEE Transactions on Information Forensics and Security*, 14(12):3246–3258, 2019.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[38] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer. *Electronic design automation for IC implementation, circuit design, and process technology: circuit design, and process technology*. CRC Press, 2016.

[39] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[40] H. Li, G. Chen, B. Jiang, J. Chen, and E. F. Young. Dr. cu 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2019.

[41] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In *Proceedings of the 2020 International Symposium*

on Physical Design, ISPD '20, page 135–142, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370912. doi:10.1145/3372780.3375560.

[42] J. Liu, C.-W. Pui, F. Wang, and E. F. Young. Cugr: Detailed-routability-driven 3d global routing with probabilistic resource model. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[43] W. Liu, Y. Li, and C. Koh. A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 713–719, Nov 2012.

[44] W. Liu, W. Kao, Y. Li, and K. Chao. Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(5):709–722, May 2013. ISSN 0278-0070. doi:10.1109/TCAD.2012.2235124.

[45] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[46] J. Lou, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. In *Proceedings of the 2001 International Symposium on Physical Design*, ISPD '01, page 112–117, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133472. doi:10.1145/369691.369749. URL https://doi.org/10.1145/369691.369749.

[47] W. Maly, C. Ouyang, S. Ghosh, and S. Maturi. Detection of an antenna effect in vlsi designs. In *Proceedings. 1996 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 86–94. IEEE, 1996.

[48] E. J. McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956.

[49] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[50] G. Moore. Moore's law. *Electronics Magazine*, 38(8):114, 1965.

[51] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[52] Y. Pan, Z. Zhou, and A. Ivanov. Routability-driven global routing with 3d congestion estimation using a customized neural network. In *2022 International Symposium on Quality Electronic Design (ISQED)*, 2022.

[53] G. Parasch and R. L. Price. Development and application of a designer oriented cyclic simulator. In *Proceedings of the 13th Design Automation Conference*, pages 48–53, 1976.

[54] Z. Qi, Y. Cai, and Q. Zhou. Accurate prediction of detailed routing congestion using supervised data learning. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 97–103. IEEE, 2014.

[55] P. Ramm, A. Klumpp, J. Weber, and M. Taklo. 3d system-on-chip technologies for more than moore systems. *Microsystem technologies*, 16 (7):1051–1055, 2010.

[56] B. Ray, A. R. Tripathy, P. Samal, M. Das, and P. Mallik. Half-perimeter wirelength model for vlsi analytical placement. In *2014 International Conference on Information Technology*, pages 287–292, 2014. doi:10.1109/ICIT.2014.61.

[57] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[58] R. B. Singh, A. S. Baghel, and A. Agarwal. A review on vlsi floorplanning optimization using metaheuristic algorithms. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 4198–4202. IEEE, 2016.

[59] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Design, Automation Test in Europe*, pages 1–6, April 2007. doi:10.1109/DATE.2007.364463.

[60] S. A. Szygenda. Tegas2—anatomy of a general purpose test generation and simulation system for digital logic. In *Proceedings of the 9th Design Automation Workshop*, pages 116–127, 1972.

[61] T. G. Szymanski. Dogleg channel routing is np-complete. *IEEE transactions on computer-aided design of integrated circuits and systems*, 4(1):31–41, 1985.

[62] A. F. Tabrizi, L. Rakai, N. K. Darav, I. Bustany, L. Behjat, S. Xu, and A. Kennings. A machine learning framework to identify detailed routing short violations from a placed netlist. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[63] A. F. Tabrizi, N. K. Darav, L. Rakai, I. Bustany, A. Kennings, and L. Behjat. Eh? predictor: A deep learning framework to identify detailed routing short violations from a placed netlist. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(6):1177–1190, 2019.

[64] G. Taylor and J. Ousterhout. Magic's incremental design-rule checker. In *21st Design Automation Conference Proceedings*, pages 160–165, 1984. doi:10.1109/DAC.1984.1585790.

[65] L.-C. Wang. Experience of data analytics in eda and test—principles, promises, and challenges. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(6):885–898, 2016.

[66] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng. *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann, 2009.

[67] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller, and S. S. Sapatnekar. Glare: Global and local wiring aware routability evaluation. In *DAC Design Automation Conference 2012*, pages 768–773, 2012.

[68] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic congestion prediction. In *Proceedings of the 2004 International Symposium on Physical Design*, ISPD '04, page 204–209, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138172. doi:10.1145/981066.981110. URL https://doi.org/10.1145/981066.981110.

[69] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[70] Y. Xu, Y. Zhang, and C. Chu. Fastroute 4.0: Global router with efficient via minimization. In *Asia and South Pacific Design Automation Conference*, pages 576–581, 2009. ISBN 978-1-4244-2748-2.

[71] X. Yang, R. Kastner, and M. Sarrafzadeh. Congestion estimation during top-down placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(1):72–80, 2002.

[72] W. Zeng, A. Davoodi, and Y. H. Hu. Design rule violation hotspot prediction based on neural network ensembles. *arXiv preprint arXiv:1811.04151*, 2018.

[73] Z. Zhou. *Machine learning based techniques for routing interconnects in very large scale integrated (VLSI) circuits*. PhD thesis, University of British Columbia, 2022. URL https://open.library.ubc.ca/collections/ubctheses/24/items/1.0413022.

[74] Z. Zhou, Z. Zhu, J. Chen, Y. Ma, B. Yu, T.-Y. Ho, G. Lemieux, and A. Ivanov. Congestion-aware global routing using deep convolutional generative adversarial networks. In *2019 ACM/IEEE 1st Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6. IEEE, 2019.