Deep Learning based Refinement for Package Substrate Routing

Peng-Tai Huang†, Tsubasa Koyama†, Keng-Tuan Chang§, Chih-Yi Huang§, Chen-Chao Wang§, Tsung-Yi Ho†‡
Department of Computer Science, National Tsing Hua University, Taiwan†
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong‡
Product Design Division, Advanced Semiconductor Engineering (ASE), Inc., Taiwan§
tedhuang0730@gmail.com; a0918050152@gmail.com; gordon_chang@aseglobal.com; joey_huang@aseglobal.com;
alexcc_wang@aseglobal.com; tyho@cs.nthu.edu.tw

***Abstract*—Heterogeneous integration packaging has become increasingly important due to recent rapid technological advancements. In these designs, substrate routing is a critical factor in terms of time to market. While there are some existing works and automatic routing tools available to help designers solve routing problems, they often result in poor performance due to the complex constraints and specifications of industrial designs. Manual revision of these results is time-consuming and can take weeks. In this work, we propose a deep learning approach to improving the area distribution and reducing detours in the auto-routing results of industrial Flip-Chip Ball Grid Array (FCBGA) substrate designs, with the goal of reducing the time needed for manual modification. Experimental results show that our proposed methods can effectively refine both detours and area distribution in auto-routing results, producing results that are similar to manual routing. We also successfully reduce the modification time compared to manual one.**

***Index Terms*—package substrate routing, routing refinement, deep neural network**

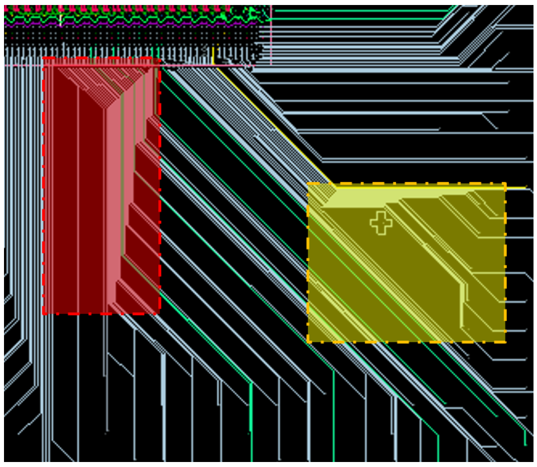
Fig. 1. Visualization of dense routing areas and detours in auto-routing result, which are highlighted in red and yellow squares, respectively.

I. Introduction

In recent years, technological progress has advanced rapidly, leading to increased demand for high-bandwidth and high-performance applications such as cloud computing and GPU chips. This has spurred innovation in the advanced Integrated Chip (IC) packaging domain, including the development of heterogeneous integration packaging like Fan-Out Chip on Substrate (FOCoS) [1] and Wafer Level Integrated Fan-Out (WLInFO) [2] technologies, which integrate individually manufactured components to form a System-in-Package (SiP). Among various package designs, Flip-Chip packaging has become particularly popular due to its high number of I/O pins, good electrical performance, and ability to facilitate direct thermal dispassion.

Substrate routing is a crucial stage in these package designs and can be divided into two subproblems: escape routing and area routing. During escape routing, the nets are routed from the pins inside the chip to the chip boundary. Then, area routing is used to connect these boundary points to the target pins. Some recent works, such as [3], [4], and [5], have proposed concurrent routing methodologies that address both escape and area routing to bridge the gap between these two subproblems. [6] proposed a ring routing framework for Flip-Chip packaging that takes into account symmetry and shielding constraints to enhance the package design cycle.

Additionally, previous works have employed Artificial Intelligence (AI) techniques such as Deep Learning (DL) and Reinforcement Learning (RL) to achieve better routing results. These techniques have demonstrated significant improvements in efficiency, accuracy, and adaptability in modern complex chip designs, which have become increasingly important. For instance, in [7], Monte Carlo Tree Search (MCTS) combined with Deep Reinforcement Learning (DRL) guided rollout policy was used to address circuit routing problems.

Despite the availability of previous works and automatic routing tools that aid designers in solving routing problems quickly, they often result in poor performance due to the complex constraints and specifications of real designs. To meet design specifications, designers may need to spend several weeks modifying the initial results, which can be a time-consuming process. Examples of modifications that designers may need to make are shown in Fig. 1, with dense routing areas and detours highlighted in red and yellow squares, respectively. Dense routing areas can result in shorts and crosstalk between adjacent nets, leading to poor electrical performance during substrate fabrication. Detours are only allowed for length matching of differential pairs. Reducing the time required for manual modification is, therefore, an important issue in the entire design cycle.

There are already some previous works that focus on refining initial routing results. [8] developed a diffusion-based topological router (D-Router) that reduces routing congestion from the initial substrate routing by iteratively spreading out the routing area through a simulated diffusion process. [9] proposed a novel framework that combines supervised machine learning and the technique of rip-up and reroute to repair defects in substrate routing.

However, most previous works such as [1], [2], [3], [4], [5], [6], [7], and [9] typically employ grid-based algorithms, which may be inefficient or time-consuming for large industrial cases. While the router proposed in [8] can produce a good area distribution in a shorter runtime, its purpose is slightly different from the focus of this work. Their router reroutes the initial routing results before the diffusion process, so the initial routing results are not directly refined. In [9], the proposed rerouting framework can lead to a manual-like design that reduces detours and slightly improves routing congestion, but the processes are done locally within fixed boxes and cannot interact with adjacent boxes to solve problems globally.

In this work, we focus on optimizing the detours and area distribution of industrial Flip-Chip Ball Grid Array (FCBGA) substrate routing from the perspective of refinement, with the goal of reducing manual modification time. Instead of constructing a new routing framework to solve the routing problems from scratch, we propose several gridless methods to refine auto-routing results, which can handle large industrial cases. In our proposed methods, we also apply ResNet50V2 [10], a Deep Neural Network (DNN) model, to bring our refined results closer to those of manual routing.

The rest of the paper is organized as follows. Section II introduces the entire refinement flow. Section III shows the experimental results. Finally, Section IV concludes this paper.

II. Methodologies

In this section, an overview of the overall flow will be provided first, followed by a description of the methods used in each stage. Fig. 2 shows the overall flow of our proposed refinement process. Given the auto-routing information, we first find the net order based on it. DNN guided routing method will then be used to refine detours as a pre-processing step for the subsequent phases. During the detour optimization stage, refinement process will be repeated until the number of detours converges. After refining the detours, net translation method will be used to distribute the routing area smoothly in the area optimization phase. The global net translation step performs rough routing area refinement by iteratively expanding the distance between nets to redistribute the entire routing area. The remaining dense areas will then be detected and further refined in the sparser direction. Finally, meaningless bends will be removed for better results.

A. Find Net Order

The net order, which is used to make our subsequent refinements smoother, will be found first. Since we are focusing on refining the auto-routing result, the net order does not need to be calculated in as complex a manner as in previous works

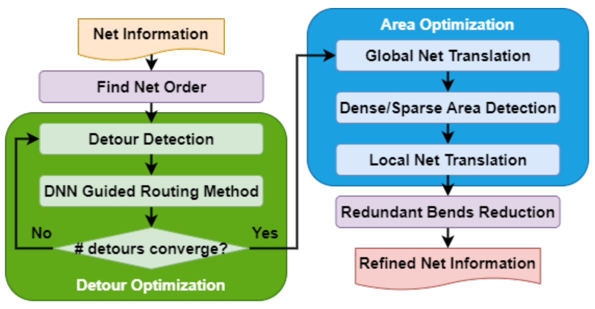

Fig. 2. Overall flow of proposed refinement.

and can be found directly from the initial routing results. Based on the characteristic of FCBGA routing, where the start points are inside and the end points are outside in each net, we use a movable frame to determine the net order. If we consider $N_K^i(x,y)$, where K is the set of nets and i is the turning points and edge points of each net, the middle points can be calculated as follows:

$$\left(\frac{min(N_K^i x) + max(N_K^i x)}{2}, \frac{min(N_K^i y) + max(N_K^i y)}{2}\right), \tag{1}$$

which denoted as $N_K^m(x,y)$. Then, the initial frame can be constructed using these midpoints, and the right bound of the frame can be calculated as follows:

$$\forall\, k \in K \wedge N_k^m x > 0 \Rightarrow \frac{1}{|k|}\sum_{p}^{p\in k} N_p^m x, \tag{2}$$

and the other bounds can also be calculated by modifying (2).

After constructing the initial frame, we check if it intersects all nets. If there are some nets do not intersect, the frame will shrink until it intersects all the nets. Once the frame is set, the net order can be obtained in either a clockwise or counterclockwise order by sorting the intersections.

B. Detour Detection

Detours in routing often occur during sharp turns or adjacent to other detoured segments. Due to the 135-degree routing restriction, these detours must contain more than three consecutive segments with at least two turns in the same direction. Detours can be easily detected by checking the vector and length of each segment as there must be short segments in every three consecutive detoured segments, and the vectors of the first and third segments must be perpendicular. We will first highlight every three detoured segments, then merge all consecutive detoured segments.

C. DNN Guided Routing Method

To adapt to large-scale cases and leverage the characteristics of 135-degree routing, we propose a parallelogram gridless routing method. Fig. 3a shows the overall flow of our DNN guided routing method. In this method, we use DNN to obtain manual-like results and the divide-and-conquer technique to reduce computational complexity. For each consecutive detoured segments, the model predicts its routing direction, taking as input the start point (blue dot), end point (yellow

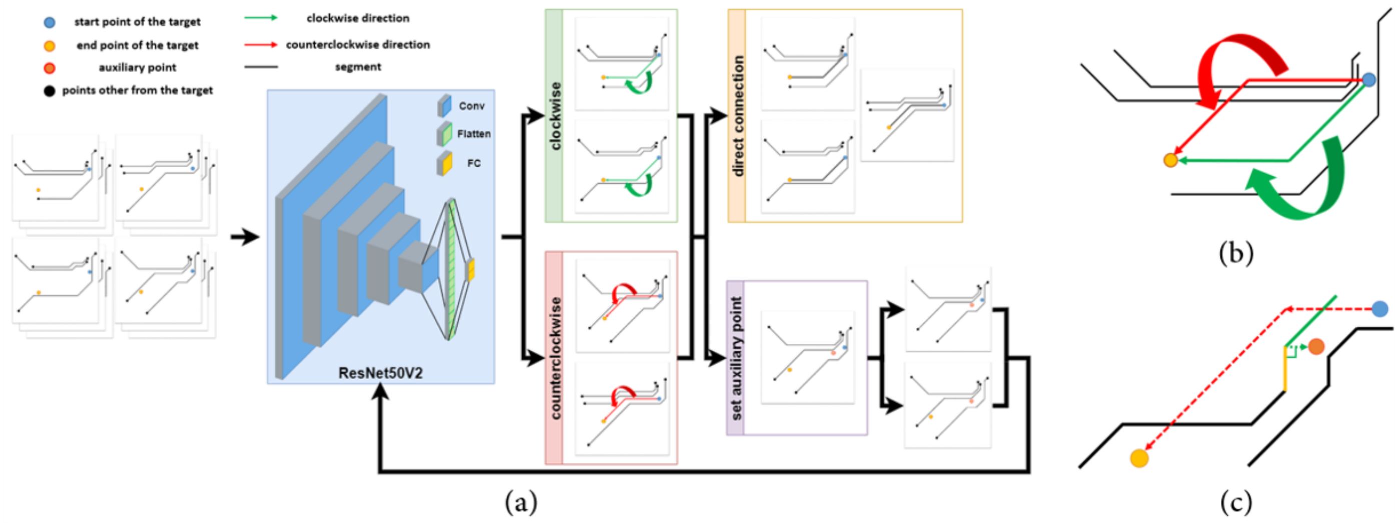

Fig. 3. (a) The overall flow of DNN guided routing method. (b) Two directions from the start point (blue dot) to the end point (yellow dot), clockwise (green lines) and counterclockwise (red lines). (c) Visualization of setting the auxiliary point (red dot) which is generated near the bottom of the first intersecting segment (green segment) and perpendicular to the next segment along the crossed one (yellow line), and within the same region as the start point (blue dot).

dot), and nearby nets (black lines). As shown in Fig. 3b, the routing direction can be only clockwise (green lines) or counterclockwise (red lines). After prediction, intersections with other nets in the predicted direction will be detected, and if there are none, we route directly from the start to the end point (orange container). Otherwise, an auxiliary point will be generated (purple container), dividing the routing problem into two subproblems, from the start point to the auxiliary point and from the auxiliary point to the end point. Fig. 3c illustrates the process of setting the auxiliary point (red dot), which will be set near the bottom of the first intersecting segment (green line) and perpendicular to the next segment of the intersected one (yellow line). It is always set in the region where the start point (blue dot) is located, as its purpose is to help routing flow smoothly and avoid net crossings. The process continues until the entire problem or subproblems are completed in a direct connection. Finally, the entire rerouted consecutive segments will be merged to complete the detour optimization stage.

In this work, we use ResNet50V2 as our DNN model. We extract images from manual routes so that the model can learn the direction that designers prefer. Each image contains the start and end points of every two consecutive segments and their adjacent nets, and one of the routing directions is the label. Clockwise or counterclockwise labels from the start point to the end point will be determined based on the manual routing results. Data with more than 10,000 images will be split into 80% for training, 10% for validation, and the rest for testing. With a model accuracy of up to 90%, our routing method takes into account manual routing information to a great extent.

D. Net Translation Method

The net translation will be performed for each net in the selected group, sequentially on every three consecutive segments starting from the first segment to the last segment. By analyzing the possible situations that net translation might encounter, we found that only two patterns of three consecutive segments need to be considered, which we named the trapezoid and parallelogram based on their translation shapes, as shown in the left half of Fig. 4. We first determine which pattern will be used in the current phase and whether we can directly construct the desired shape. If it is possible, the desired shape will be constructed directly and the proposed net translation method will be performed. The upper part of each shape in Fig. 4 shows an example of their direct construction and net translation, with red lines and orange arrows indicating new segments and translation directions, respectively. If direct construction is not possible, a break point (red dot) will be set according to the nearby segments and split one of the three consecutive segments to smooth the translation process. In the case of the parallelogram pattern, the break point will be moved by a certain value to avoid net overlap when forming the shape. During each translation, the specification violation will be checked and if it violates, the movement will be canceled.

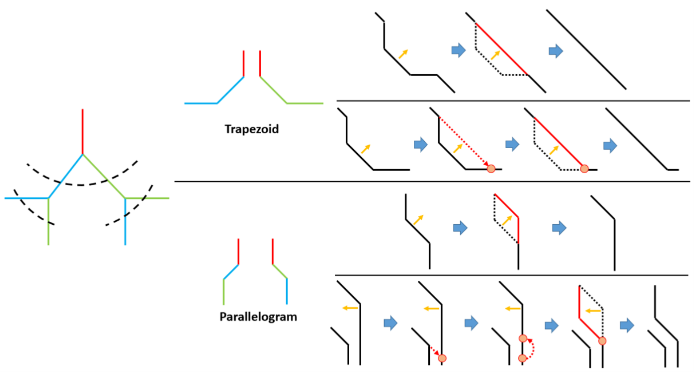

Fig. 4. Two patterns of nets with left and right turns from the red line with vertical routing direction are highlighted in green and blue, respectively. The right half shows some net translation examples.

TABLE I
ROUTING RESULT COMPARISON

	Benchmark			**Wirelength (μm)**		**# Segments**		**# Dense Areas**	
	# Nets	**Chip size**	**Substrate size**	**Auto-route**	**Ours**	**Auto-route**	**Ours**	**Auto-route**	**Ours**
Case1	308	$11165 * 3462$	$29600 * 29600$	3599015	3576650	1993	1028	1459	559
Case2	233	$12709 * 11167$	$50268 * 53002$	2450300	2417559	1443	450	923	631
Case3	296	$22318 * 27116$	$65180 * 65380$	6140656	6034468	3646	1414	2873	2208
Case4	195	$9833 * 8478$	$24610 * 24610$	1962700	1930996	1819	835	964	543
Comparisons				1.00	0.99	1.00	0.42	1.00	0.60

E. Dense/Sparse Area Detection

After the global net translation phase, we detect the remaining dense areas and their nearby largest sparse areas. To detect dense areas, the original design will be divided into several routing blocks first. We then check if the value of each block exceeds a defined threshold to detect whether the partial area is dense or not. Sparse areas are then detected in two directions perpendicular to the routing direction of each continuous dense area. The censored line expands until it hits other dense areas or design edges. After detecting the sparse areas in these two directions, the refinable area will be calculated and the larger one will be chosen as the detection result for this continuous dense area.

F. Redundant Bends Reduction

In the refinement process, we use break points to smooth the translation of dense areas. However, these break points may result in redundant bends in the routing details. To eliminate these meaningless bends, in the final stage of our proposed refinement flow, we apply the parallelogram approach of the net translation method to convert them into straight lines.

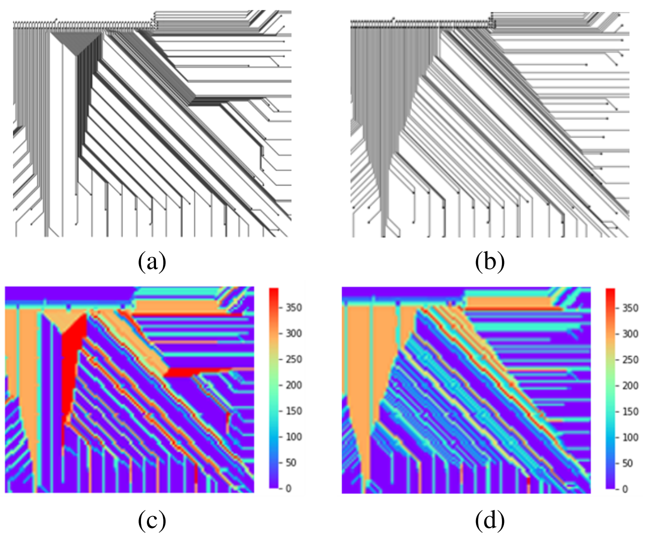

Fig. 5. Comparison between the partial results of auto-routing (a) and after our refinement (b) of Case1. Heatmaps, (c) and (d), are provided to compare the routing area density more easily.

III. EXPERIMENTAL RESULTS

The proposed refinements are applied to four industrial cases provided by Advanced Semiconductor Engineering (ASE) Group for evaluation. TABLE I shows the information for these four cases and compares the results of our refinements with the auto-routing results. The wirelength is not increased during our refinement, and the number of segments and dense areas can be used to evaluate the effectiveness of our refinements in reducing detours and improving area distribution, respectively. Fewer segments suggest fewer detours and redundant bends, while fewer dense areas indicate a more evenly distributed routing area. Our proposed methods can improve detours and area distribution by an average of 58% and 40%, respectively. Partially refined results for Case 1, including heatmaps for comparing routing area densities, are shown in Fig. 5. The proposed refinements have significantly reduced the time required to complete the routing process, which used to take weeks to complete manually but can now be done in seconds.

IV. CONCLUSIONS

In this paper, we proposed a deep learning approach to refine substrate routing, aiming to improve upon existing auto-routing results rather than constructing an entire routing framework from the ground up. Our methods, which are designed to handle large-scale industrial cases, are all gridless, making them faster and more flexible than grid-based algorithms. In addition to improving effectiveness, we also took into account industrial requirements by using DNN to anticipate the routing features preferred by designers. The experimental results showed that our methods were able to effectively refine detours and area distribution without increasing the wirelength. We were also successful in reducing the time needed for modification compared to the manual one.

REFERENCES

[1] Lai et al., “A Comparative Study of 2.5D and Fan-out Chip on Substrate: Chip First and Chip Last,” IEEE ECTC, 2020.
[2] Tseng et al., “InFO (Wafer Level Integrated Fan-Out) Technology,” IEEE ECTC, 2016.
[3] Chang et al., “Irregular Bumps Design Planning for Modern Ball Grid Array Packages,” IEEE ECTC, 2020.
[4] Lin et al., “A Complete PCB Routing Methodology with Concurrent Hierarchical Routing,” ACM/IEEE DAC, 2021.
[5] Lin et al., “A Unified Printed Circuit Board Routing Algorithm With Complicated Constraints and Differential Pairs,” IEEE ASP-DAC, 2021.
[6] Chi et al., “Practical Substrate Design Considering Symmetrical and Shielding Routes,” IEEE DATE, 2022.
[7] He et al., “Circuit Routing Using Monte Carlo Tree Search and Deep Reinforcement Learning,” IEEE VLSI DAT, 2022.
[8] Liu et al., “Effective congestion reduction for IC package substrate routing,” ACM TODAES, 2010.
[9] Yeh et al., “Substrate Signal Routing Solution Exploration for High-Density Packages with Machine Learning,” IEEE VLSI DAT, 2022.
[10] He et al., “Identity Mappings in Deep Residual Networks,” ECCV, 2016.