TRouter: Thermal-Driven PCB Routing via Nonlocal Crisscross Attention Networks

Tinghuan Chen, *Member, IEEE*, Silu Xiong, Huan He, and Bei Yu, *Senior Member, IEEE*

***Abstract*—In this article, we propose TRouter, a thermal-driven printed circuit board (PCB) routing framework via a machine-learning model. The model is designed to capture the long-range spatial information from the PCB layout and predict thermal distribution. The information contains pads, vias, components and wire segments. A gradient in each grid cell obtained from the backpropagation is integrated into a full-board routing algorithm to guide thermal-aware wire detour and via punching. To achieve a significant speedup, we construct a conflict graph according to whether overlapping among convex hulls of nets. A greedy-based method is adopted to remove nonroot nodes from all nodes. Then, a task graph is constructed to improve the parallelism. We conduct experiments on open-source benchmarks to illustrate our TRouter can achieve significant speedup and lower-temperature designs, compared with a state-of-the-art PCB routing algorithm.**

***Index Terms*—EDA, machine learning (ML), physical synthesis, printed circuit board (PCB), routing, thermal optimization.**

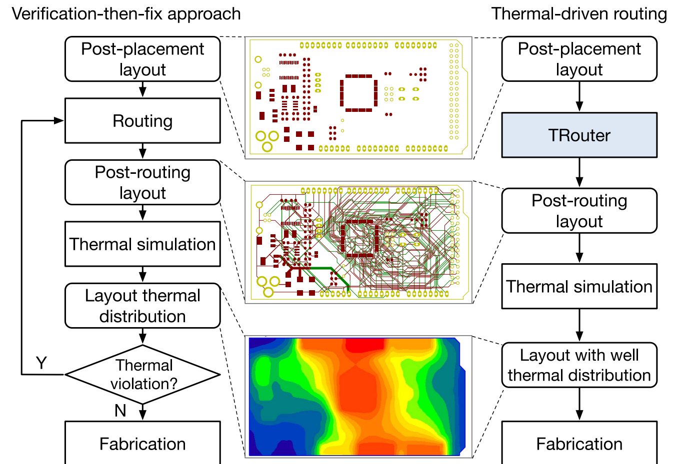

Fig. 1. Verification-then-fix routing approach and our thermal-driven routing flow.

I. Introduction

PRINTED circuit boards (PCBs) are an important part of electronic equipment, connecting different components by a complex array of circuits. PCB failure is a major concern in the industry since it is used in many important electronics, such as wearables, satellites, medical devices, and airplanes [1]. It is crucial that failures are quickly identified and the appropriate measures are taken. In order to keep electronics working smoothly, any industrial company is looking to better address PCB failure and even prevent it from occurring.

High temperature is one of the reasons for PCB failure, since it can lead to malfunctions and permanent damage [2]. Routing is one of the crucial steps impacting PCB thermal distribution [3]. Because the routing (vias and metal wires) locations and densities influence heat dissipation. For example, according to design experience, placing vias nearby components can facilitate heat dissipation [4].

Manuscript received 18 July 2022; revised 15 November 2022 and 3 January 2023; accepted 22 January 2023. Date of publication 10 February 2023; date of current version 20 September 2023. This work was supported in part by Huawei and in part by the Research Grants Council of Hong Kong SAR under Project CUHK14209420. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Bei Yu.)*

Tinghuan Chen was with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR. He is now with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China.

Silu Xiong and Huan He are with Huawei, Hangzhou 310051, China.

Bei Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2023.3243544

To achieve a low-temperature PCB design, the verification-then-fix approaches are used in the design flow, as shown in Fig. 1. In the design flow, many advanced routing algorithms are adopted to automatically route all nets. These algorithms are classified into escape routing and area routing [5]. They assume that the PCB designs contain only ball grid array (BGA) packages. The escape routing routes the I/O pads or solder bumps on a die or package to the lines that can escape to the area surrounding the die [6]. The area routing connects the previously escaped routes of I/O pads and is usually subject to an upper/lower bound of routed length for each connection [7]. The escape routing can be further categorized into the ordered escape and the unordered escape. The former routes the connections with specific ordering on the boundary, while the latter needs not. Based on escape and area routings, some excellent works were proposed to route all nets under complicated constraints [8], [9]. However, they may be inappropriate for dealing with irregular pads. Recently, a unified routing framework was proposed to handle such designs with the routing of P/G nets, signal nets, and irregular pads together [10].

After the PCB layout design is finished, some models are used to obtain the thermal distribution. The thermal simulation provides interactive feedback for reliability during the design process. The most popular thermal models can be classified into two categories: 1) computational fluid dynamics (CFDs) [11] and 2) compact thermal models [3], [12]. CFD

1937-4151 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

requires assuming the amount of heat generated by different components, pads, vias, and segments. PCB is divided into smaller parts. A grid construction implements space discretization. The thermal distribution is described by partial differential equations, which are handled to obtain the temperature value at each grid cell [11]. In compact thermal models, heat transfer is analogous to electrical phenomena in an RC circuit. Each node in the circuit is regarded as a block, whose heat dissipation is regarded as a current source connected to the node [3], [12]. However, the traditional verification-then-fix approaches with these models are ill-equipped when faced with ever-growing thermal limit violations. As shown in Fig. 1, if there are one or more thermal violations, the designers will return to the previous routing step to fix them. Thus, traditional approaches have low efficiency in achieving a low-temperature PCB design.

To improve the design and verification efficiency, a straightforward strategy is to integrate a thermal model into the routing stage. In [4] and [13], based on the compact thermal models [3], the maximum temperature is minimized to the through-the-silicon via number at each tile. Then, wire and via locations are determined by minimizing wirelength. In [14], dummy nonsignal vias and wires are inserted to reduce temperature. However, they occupy extra routing resources, which may cause routing failure. In addition, dummy nonsignal vias will bring extra manufacturing costs [4]. Pathak and Lim [15], [16] proposed that the temperature is minimized by relocating vias. However, these thermal-driven routing frameworks cannot provide a thermal-aware guide for wire detour. In other words, not enough freedoms in both wire detour and via punching are provided to optimize thermal performance in previous works [4], [13], [14], [15], [16]. The primary reason behind the freedom issue is that existing thermal models [3], [11], [12] are difficult to provide a guide for thermal-aware wire detour and via punching at the routing stage.

Recently, machine-learning (ML) methods have been developed in the computer-aided design (CAD) field and have already achieved significant success [17], [18]. ML transforms the conventional analytical modeling and optimization problem into a data-to-data mapping problem to provide an efficient and accurate performance evaluation at several design stages [19], [20], [21], [22]. As an ML model, convolutional neural networks (CNNs) can extract and abstract features layer by layer from image-based data to outperform traditional shallow ML models in handling difficult tasks [23], [24]. In the CAD field, CNNs were leveraged to detect layout manufacturability and reliability violations [25], [26], [27], [28], [29], [30]. CNNs were developed to predict the congestion heatmap. Then, the model was further used to avoid unnecessary searches and accelerate the overall routing process [31]. CNNs were proposed to provide a routing guide by mimicking the sophisticated manual layout approaches [32]. Intuitively, the PCB layout can be naturally represented as image-based data. However, the routing cannot be simply guided by the predicted thermal distribution, since the high-temperature areas cannot be straightforwardly set as a keepout or large cost area. Instead, their neighborhoods and thermal dissipation ability should be considered. To our best knowledge, none of the prior art directly handles the performance- or reliability-driven routing problem via ML models.

In this article, we propose TRouter, a thermal-driven PCB routing framework via an ML model. We make the following contributions.

1) An ML model is designed to capture the long-range spatial information from the PCB layout and predict thermal distribution. The long-range spatial information contains pads, vias, components and wire segments.
2) Since the gradient value represents the thermal sensitivity, it is integrated into a full-board routing algorithm to guide thermal-aware wire detour and via punching. Compared with previous works [4], [13], [14], [15], [16], our thermal-aware routing guide can provide more freedom to optimize thermal performance and reduce temperature.
3) In order to achieve a significant speedup, according to whether overlapping among convex hulls of nets, we construct a conflict graph. A greedy-based method is adopted to remove nonroot nodes from all nodes. Then, a task graph is constructed to improve the parallelism.
4) We conduct experiments on open-source benchmarks to illustrate our TRouter can achieve significant speedup and lower-temperature designs, compared with a state-of-the-art PCB routing algorithm.

The remainder of this article is organized as follows. In Section II, we provide a problem formulation about thermal-driven PCB routing. In Section III, we provide an overall flow. In Section IV, we present a thermal-driven routing guide generation, including feature extraction, network structure, and gradient value generation. In Section V, we use the gradient values to sufficiently honor the thermal-driven routing guide in detailed routing, where a task graph is constructed with net convex hulls to improve the parallelism. Section VI presents experimental results with comparison and discussion, followed by the conclusion in Section VII.

II. Problem Formulation

This section introduces some terminologies and related problem formulation. As illustrated in Fig. 2, a two-layer PCB layout contains components with solder mask-defined (SMD) pads and/or through-hole pads. They are defined as follows.

Definition 1 (Component): Any basic discrete device or physical entity in an electronic system is used to affect electrons or their associated fields.

Definition 2 (SMD Pad): A pad is in the top layer or bottom layer.

Definition 3 (Through-Hole Pad): A pad is punched through all routing layers.

Note that as a PCB routing problem, all components layer assignments and positions have been given in the post-placement layout. To perform routing, a netlist is used to define connections among different component pads. The netlist is defined as follows.

Definition 4 (Netlist): A description of the connectivity among through-hole pads and/or SMD pads.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

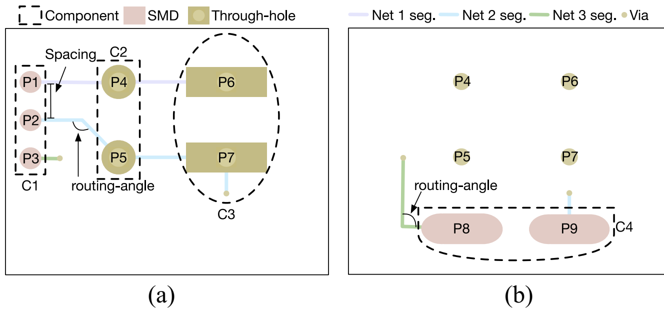

Fig. 2. Two-layer PCB layout and design rules. (a) Top layer. (b) Bottom layer. The components C1–C3 are in the top layer. C4 is in the bottom layer. The SMD pads P1–P3 are in the top layer. P8 and P9 are in the bottom layer. The through-hole pads P4–P7 are punched from the top layer to the bottom layer.

In order to achieve good manufacturability, routing results need to satisfy design rules. The typical rules contain noncrossing, spacing, and routing-angle constraints. Noncrossing means no crossing happens among different nets on the same layer. Fig. 2 shows spacing and routing-angle rules. A specific spacing distance between wires and/or vias on the same layer is defined in the spacing constraint. Only 90° or 135° angles between two connected segments are allowed.

According to the definitions, we define the thermal-driven PCB routing problem as follows.

Problem 1 (Thermal-Driven PCB Routing): Given a netlist, design rules and a post-placement layout containing a set of SMD pads, a set of through-hole pads, and a set of components, connect all the nets so that there is no design rule violation, and the total wirelength, via number and temperature are minimized.

III. Overview

To handle Problem 1, we propose TRouter, a thermal-driven PCB routing framework via an ML model. Our TRouter mainly contains thermal-driven routing guide generation and guided detailed routing. The thermal-driven routing guide generation method is adopted to obtain a guide by minimizing the temperature. And the guided detailed routing method is adopted to route all the nets while honoring the guide.

The overview is shown in Fig. 3. The steps related to thermal-driven routing guide generation (guided detailed routing) are highlighted in purple (green). We devise our TRouter based on the grid-based model since it has a simple data structure, typically constructs routing models quickly, and allows fast neighbor identification and one-step move [33]. Initially, the routing obstacles are captured by rasterizing arbitrary shape obstacles and pads to the grid for path search. The layout patterns are captured by rasterizing the arbitrary shapes of components and pads to the grid. A thermal-driven routing guide is generated by the backpropagation of a pretrained ML model. Meanwhile, we construct a conflict graph according to whether we are overlapping among convex hulls of nets. Each net is represented as one node. There is one edge between two nodes if the convex hulls of the two corresponding nets are overlapped. Then, a task graph is constructed by using the conflict graph and thermal-driven routing guide. According to the task graph, a batch of nets is iteratively routed. In each iteration, components, pads, deployed segments, and vias are captured by rasterizing them to the grid before routing. Then, the thermal-driven routing guide is updated at each grid cell. Each net is routed by honoring that guide. When all nets are routed, extra iterations with ripping-up and rerouting are performed to reduce the violation number. Finally, our TRouter outputs the routing result.

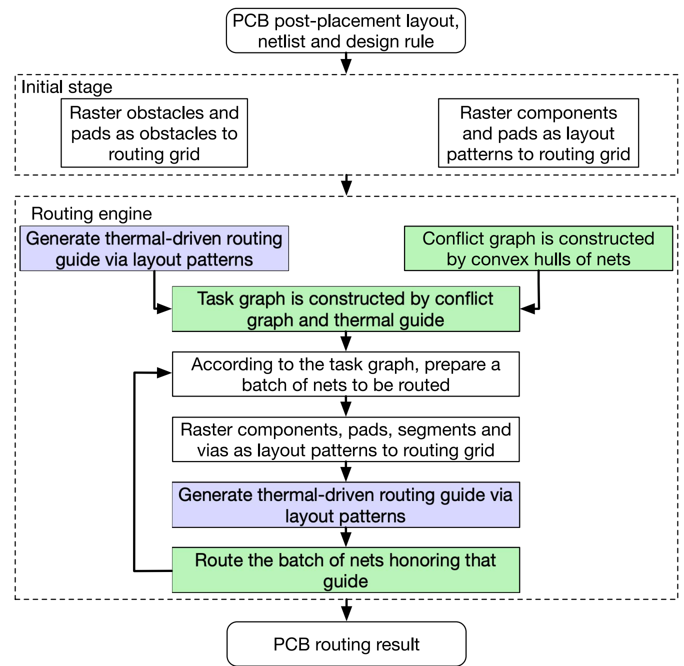

Fig. 3. Our TRouter overview.

IV. Thermal-Driven Routing Guide Generation

In this section, we present a thermal-driven routing guide generation. To provide a guide for thermal-aware wire detour and via punching at the routing stage, we design an ML model named nonlocal attention networks. The PCB layout can be naturally represented as image-based data. CNNs can extract and abstract features layer by layer from image-based data. Our model input is the PCB layout and the output is thermal distribution. Compared with traditional CNNs, our designed model can harvest long-range contextual information from full-PCB dependencies.

To bridge the gap between our model and the routing engine, we need to map the PCB layout into a grid. This mapping can avoid time consuming and tricky interactions with the CAD tool. The layout patterns contain pads, vias, components and wire segments. As a result, our model can provide a guide for thermal-aware wire detour and via punching at the routing stage. Unlike the previous work [32], where the inference is only performed once, our TRouter needs interaction between the routing engine and the ML model for routing each or several nets. Because the high-temperature areas cannot be straightforwardly set as a keepout or large cost area, our TRouter cannot be simply guided by the predicted thermal

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

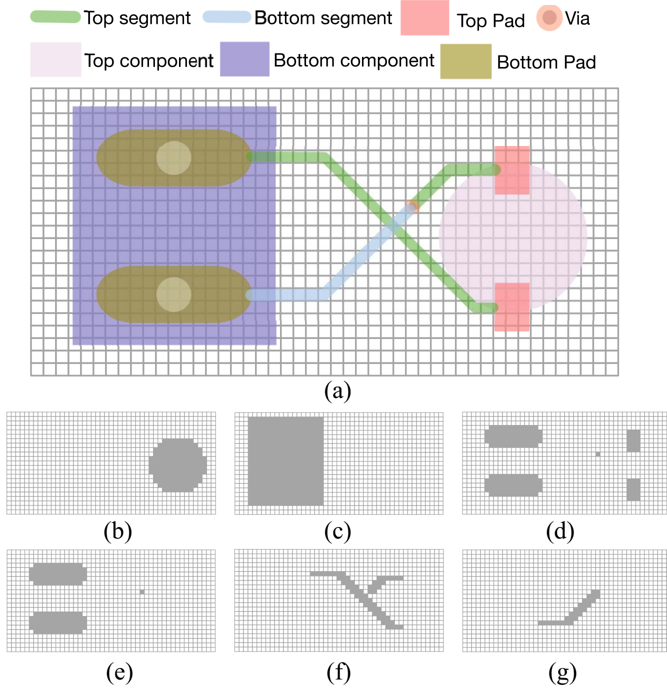

Fig. 4. Two-layer PCB layout and feature patterns: (a) routing grid; (b) top component pattern; (c) bottom component pattern; (d) top pad/via pattern; (e) bottom pad/via pattern; (f) top segment pattern; and (g) bottom segment pattern.

distribution. Instead, our TRouter adopts gradient values in each grid cell obtained from BP of our pretrained model as a thermal-driven routing guide.

Our thermal-driven routing guide generation consists of three tasks: 1) feature extraction from routing layout; 2) model structure and training; and 3) gradient value generation in the grid cell.

A. Feature Extraction From Routing Layout

To extract feature patterns from the layout and perform grid-based routing, we rasterize the whole layout as shown in Fig. 4(a). These arbitrary shapes along with the pads are rasterized to a 3-D routing grid as obstacles to facilitate later stages and derive a design-rule-violation-free routing solution. Besides, rasterizing can help extract feature patterns from the layout without the help of any CAD tools. We rasterize all components, pads, vias, and segments as features since they play an important role in thermal distribution. Besides, a guide for thermal-aware wire detour and via punching can be provided at the routing stage. Note that in this example as shown in Fig. 4(a), the PCB has two layers. Thus, based on the 3-D routing grid, we can directly extract these patterns as shown in Fig. 4(b)–(g). Each pattern can be naturally encoded as a binary 3-D tensor, where the value is 1 if the region is occupied (gray cell). Otherwise, the value is 0 (white cell). Moreover, one $|L|$-layer layout is encoded as a tensor with $2 + 2|L|$ channels. The tensor contains component pattern $\mathfrak{X}_{cp} \in \mathbb{R}^{2\times W\times H}$, pad/via pattern $\mathfrak{X}_{pv} \in \mathbb{R}^{|L|\times W\times H}$ and segment pattern $\mathfrak{X}_{sg} \in \mathbb{R}^{|L|\times W\times H}$. W and H denote the routing grid width and height, respectively.

B. Model Structure and Training

After the pattern features are extracted, an ML model will be built to predict the thermal distribution. In each PCB design, the layout has the same size as its thermal distribution heatmap, as shown in Fig. 5. The different PCB designs have different layout sizes W and H. However, the traditional CNNs with fully-connected (FC) layers cannot be used since the FC layer cannot take varying-size features as inputs. In order to support varying size input and predict the thermal distribution heatmap, a fully convolutional network (FCN) is constructed by replacing all FC layers with convolutional and deconvolutional layers [34]. The deconvolutional layers are implemented by interpolation methods to enlarge the feature size. Based on FCN, U-net is developed to perform feature fusion with low-level features by skip connections [35] for better prediction performance. However, the traditional U-net is limited to short-range spatial information since it adopts the traditional convolution of local receptive fields. The thermal distribution relies on long-range spatial information on PCB.

To honor the advantage of the feature fusion in U-net and capture long-range spatial information on PCB, we customize an ML model named nonlocal crisscross attention networks. This model uses U-net as our backbone with crisscross attention modules, as shown in Fig. 5, where our model configurations (i.e., channel number, convolution kernel size, and pooling size) are given. We adopt different configurations for the PCB layout with different layer numbers. One $|L|$-layer PCB layout is encoded as a tensor $[\mathfrak{X}_{cp}, \mathfrak{X}_{pv}, \mathfrak{X}_{sg}]$ with $W\times H$ size and $2+2|L|$ channels to input into our model. Our model output channel number relies on the thermal simulator. Here, we uniformly set it as 1.

As shown in Fig. 5, our model can be divided into three parts: 1) encoder; 2) crisscross attention; and 3) decoder. In the encoder part, 3×3 convolution, ReLU, and 2×2 MaxPooling are sequentially performed. Thanks to 3×3 convolution, the channel number is gradually enlarged from $2 + 2|L|$ to 512. Because of 2×2 MaxPooling, the feature size is gradually reduced from $W \times H$ to $W/8 \times H/8$. In the crisscross attention part, two modules assign different weights (attention map) to each part of the feature through crisscross directions and extract more critical, important, and long-range spatial information [36]. In the decoder part, deconvolution, concatenation, 3×3 convolution, and ReLU are sequentially performed so that the feature size and channel number gradually become $W \times H$ and 1, respectively.

The crisscross attention module, as shown in Fig. 6, is adopted to capture long-range spatial information on PCB meanwhile less computation and space overheads are introduced. More specifically, an attention map is obtained to aggregate features through crisscross directions. First, three 1×1 convolutions are adopted to extract features from input feature $\mathfrak{X}' \in \mathbb{R}^{C'\times W'\times H'}$ and obtain three feature maps $\mathfrak{F}^{(1)} \in \mathbb{R}^{C''\times W'\times H'}$, $\mathfrak{F}^{(2)} \in \mathbb{R}^{C''\times W'\times H'}$, and $\mathfrak{F}^{(3)} \in \mathbb{R}^{C'\times W'\times H'}$. C' and C'' are the channel numbers. According to our model as shown in Fig. 5, $W' = W/8$, $H' = H/8$, and $C' = 512$ are the input feature $\mathfrak{X}$ width, height, and channel number. C'' is empirically set as 256. To generate a crisscross attention map, we need to obtain a correlation tensor from $\mathfrak{F}^{(1)}$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

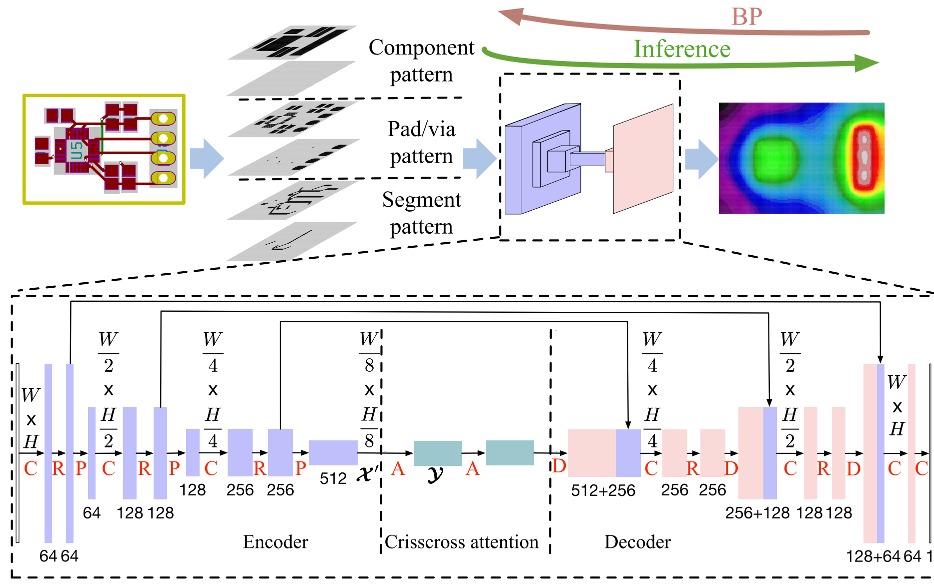

Fig. 5. Thermal distribution prediction model. Box represents the feature tensor. The number below the box represents the channel number. C: 3 × 3 convolution; R: ReLU; P: 2 × 2 MaxPooling; A: crisscross attention; and D: deconvolution.

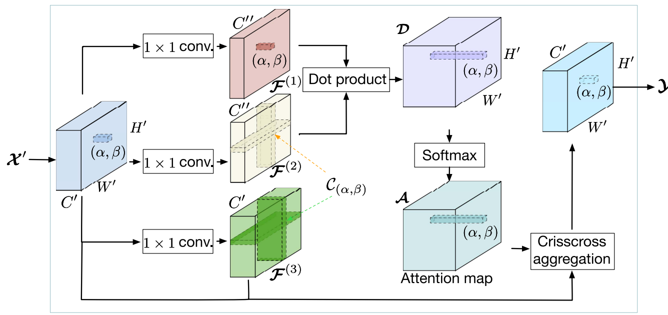

Fig. 6. Crisscross attention module. The module configurations: $W' = W/8$, $H' = H/8$, $C' = 512$, and $C'' = 256$.

and $\mathfrak{F}^{(2)}$. For convenience, for a spatial coordinate (α, β), where $\alpha = 1, 2, \ldots, W'$ and $\beta = 1, 2, \ldots, H'$, we define a crisscross set

$$\mathcal{C}_{(\alpha,\beta)} = \{(1, \beta), \ldots, (W', \beta), (\alpha, 1), \ldots (\alpha, \beta - 1), (\alpha, \beta + 1), \ldots, (\alpha, H')\} \quad (1)$$

whose each element column index is α or row index is β. Thus, this set has $W' + H' - 1$ elements, as shown in Fig. 6. Then, for the feature vector $\boldsymbol{f}^{(1)}_{\alpha,\beta} \in \mathbb{R}^{C''}$ (denoted by dark red) at the spatial coordinate (α, β) in $\mathfrak{F}^{(1)}$, we collect all feature vectors $\boldsymbol{f}^{(2)}_{\mathcal{C}_{(\alpha,\beta)}(j)} \in \mathbb{R}^{C''}$ (denoted by dark yellow) from $\mathfrak{F}^{(2)}$, where $\mathcal{C}_{(\alpha,\beta)}(j)$ denotes the jth element in the set $\mathcal{C}_{(\alpha,\beta)}$. In the correlation tensor $\mathcal{D} \in \mathbb{R}^{(W'+H'-1)\times W'\times H'}$, at the spatial coordinate (α, β), each channel element (denoted by dark purple) can be obtained by

$$d_{j,\alpha,\beta} = \left(\boldsymbol{f}^{(1)}_{\alpha,\beta}\right)^{\top} \cdot \boldsymbol{f}^{(2)}_{\mathcal{C}_{(\alpha,\beta)}(j)}. \quad (2)$$

The correlation tensor $\mathcal{D}$ can be used to evaluate the crisscross correlation between $\mathfrak{F}^{(1)}$ and $\mathfrak{F}^{(2)}$. Furthermore, the attention map $\mathcal{A} \in \mathbb{R}^{(W'+H'-1)\times W'\times H'}$ is obtained by softmax operation for $\mathcal{D}$ over the channel dimension. Finally, according to the attention map, the feature can be aggregated through crisscross directions as follows:

$$\boldsymbol{y}_{\alpha,\beta} = \sum_{j=1}^{|\mathcal{C}_{(\alpha,\beta)}|} a_{j,\alpha,\beta} \boldsymbol{f}^{(3)}_{\mathcal{C}_{(\alpha,\beta)}(j)} + \boldsymbol{x}'_{\alpha,\beta} \quad (3)$$

where $\boldsymbol{x}'_{\alpha,\beta}$ and $\boldsymbol{y}_{\alpha,\beta}$ denote a feature vector at the spatial coordinate (α, β) in $\mathcal{X}'$ and $\mathcal{Y}$, respectively. $a_{j,\alpha,\beta}$ is the element at the jth channel. (α, β) is the spatial coordinate in the attention map $\mathcal{A}$.

Note that by using one crisscross attention module, the long-range spatial information can be aggregated through crisscross directions. Moreover, two stacked crisscross attention modules can be used to aggregate long-range spatial information between any two positions. Thus, as shown in Fig. 5, in our crisscross attention part, two stacked crisscross attention modules are used to capture long-range spatial information on the PCB layout.

Formally, our model $\phi_{\boldsymbol{W}}$ takes component pattern tensor $\mathcal{X}_{cp}$, pad/via pattern tensor $\mathcal{X}_{pv}$ and segment pattern tensor $\mathcal{X}_{sg}$ as inputs. Then, it outputs the predicted thermal distribution $\hat{\boldsymbol{Y}} = \phi_{\boldsymbol{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg}) \in \mathbb{R}^{1\times W\times H}$, where $\boldsymbol{W}$ denotes the trainable model coefficients, including kernels in convolutions as shown in Figs. 5 and 6. $\boldsymbol{W}$ is determined at the training stage by minimizing the loss function as follows:

$$\mathcal{L}(\boldsymbol{W}) = \frac{||\boldsymbol{Y} - \phi_{\boldsymbol{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})||_2^2}{H \cdot W} \quad (4)$$

which is the mean-square error between the predicted thermal distribution and ground truth. The stochastic gradient descent method is used to train our model. $\boldsymbol{Y}$ denotes the ground-truth thermal distribution matrix, which is obtained from a thermal simulator.

In traditional analytical models, the thermal distribution is described by partial differential equations to the information on layout space discretization [11]. Our model learns the relationship between the thermal distribution and the information

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

on layout space discretization. Thus, it has the same functionality as the learning-based electromigration model [21], [30]. Compared with traditional analytical models [3], [11], [12], our model can provide a guide for thermal-aware wire detour and via punching at the routing stage.

C. Gradient Value Generation in Grid Cell

Now we have proposed an ML model $\phi_{\mathbf{W}}$ to predict thermal distribution. However, the grand challenge is how to use the pretrained model to guide routing. We propose to use gradient values obtained from BP as a cost at each routing grid cell to guide thermal-driven routing. The main idea behind the strategy is that the gradient values represent the thermal sensitivity for the routing pattern.

One of our routing objectives is to minimize the temperatures as follows:

$$\min_{\mathcal{X}_{pv},\mathcal{X}_{sg}} \quad \frac{1}{H \cdot W} ||\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})||_2^2. \tag{5}$$

We denote $\psi(\boldsymbol{x}_{pv}, \boldsymbol{x}_{sg}) \triangleq ||\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})||_2^2/(H \cdot W)$, where each input tensor $\mathcal{X}_{\cdot}$ is vectorized as $\boldsymbol{x}$. H and W are the routing grid height and width, as shown in Fig. 4. Then, the first-order Taylor expansion of $\psi(\boldsymbol{x}_{pv}, \boldsymbol{x}_{sg})$ can be expressed as follows:

$$\psi\left(\boldsymbol{x}_{pv}^{(i)} + \Delta\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)} + \Delta\boldsymbol{x}_{sg}^{(i)}\right) \approx \psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right) + \left(\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{pv}}\right)^{\top} \Delta\boldsymbol{x}_{pv}^{(i)} + \left(\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{sg}}\right)^{\top} \Delta\boldsymbol{x}_{sg}^{(i)} \tag{6}$$

where i denotes that the ith routing step. $\Delta\boldsymbol{x}_{pv}^{(i)}$ and $\Delta\boldsymbol{x}_{sg}^{(i)}$ mean incremental vias and segments for routing in the next step. Since

$$\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\phi_{\mathbf{W}}\left(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg}\right)} = \frac{2}{H \cdot W} \tag{7}$$

according to the chain rule, the relationships between $\partial\psi(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{pv}$ $(\partial\psi(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{sg})$ and $\partial\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})/\partial\boldsymbol{x}_{pv}$ $(\partial\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})/\partial\boldsymbol{x}_{sg})$ are shown as follows:

$$\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{pv}} = \frac{2}{H \cdot W} \cdot \frac{\partial\phi_{\mathbf{W}}\left(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg}\right)}{\partial\boldsymbol{x}_{pv}} \tag{8}$$

$$\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{sg}} = \frac{2}{H \cdot W} \cdot \frac{\partial\phi_{\mathbf{W}}\left(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg}\right)}{\partial\boldsymbol{x}_{sg}} \tag{9}$$

where in practice $\partial\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})/\partial\boldsymbol{x}_{pv}$ and $\partial\phi_{\mathbf{W}}(\mathcal{X}_{cp}, \mathcal{X}_{pv}, \mathcal{X}_{sg})/\partial\boldsymbol{x}_{sg}$ are calculated by BP of our pretrained model with Pytorch or C++ libtorch library [37].

According to our feature pattern definition, routing is the process where all unoccupied regions (values are 0 in pattern tensor) are chosen to detour wires or punch vias (values are 1 in pattern tensor). $\Delta\boldsymbol{x}_{sg}^{(i)}, \Delta\boldsymbol{x}_{pv}^{(i)} \in \{0, 1\}^{W \times H \times |L|}$. According to (6), the smaller gradient values $\partial\psi(\boldsymbol{x}_{cp}, \boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{pv}$ or $\partial\psi(\boldsymbol{x}_{cp}, \boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{sg}$ have negative effects on temperature increase in the routing step. As a result, before routing each or several nets, as shown in Fig. 7, the layout patterns are input into the pretrained model then BP is performed to obtain gradient value $r(\boldsymbol{c})$ in each grid cell, where $\boldsymbol{c}$ is the spatial coordinate in the layout pattern tensor.

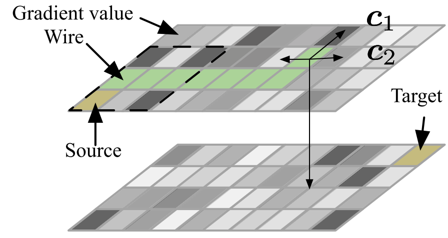

Fig. 7. Routing pattern and gradients. Gray represents the gradient value. The darker gray represents the larger gradient value; the lighter gray represents the smaller gradient value. The dotted rectangle represents the component. $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ denote the next spatial coordinates for routing.

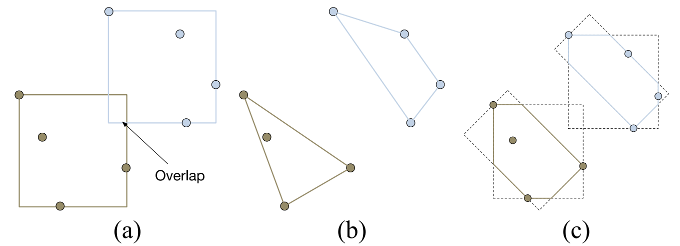

Fig. 8. (a) Net bounding box: the two nets cannot be routed simultaneously since there is an overlapping between the two bounding boxes. (b) Net convex hull in [38]. (c) Our net convex hull: the two nets can be routed simultaneously since there is no overlapping between the two convex hulls.

V. Guided Detailed Routing

In detailed routing, the net ordering and the path search algorithm play an important role in routing quality and thermal performance. To sufficiently honor the thermal-driven routing guide, we propose to use the gradient values in net ordering and the path search algorithm. In net ordering, according to whether overlapping among nets, we simultaneously route several nets for acceleration.

In traditional net ordering methods, the conflict graph is constructed by the net bounding box [39], as shown in Fig. 8(a). More specifically, if there is an overlap between two nets' bounding boxes, they cannot be routed simultaneously. However, this scheme will bring low parallelism. In [38], as shown in Fig. 8(b), a net convex hull is proposed to optimize congestion at the placement stage. However, the convex hull in [38] cannot be directly used to determine whether nets can be simultaneously routed. Since only 90° or 135° angles between two connected segments are allowed in the PCB design rule. Based on the routing-angle constraint, we propose a convex hull, which is the intersection area between the two bounding boxes in 0° and 135° directions, as shown in Fig. 8(c).

Our convex hull is constructed as shown in Fig. 9. First, each pad of the net is rotated 45° clockwise, as shown in Fig. 9(a). Next, the minimum and maximum horizontal and vertical coordinates are found among the centers of all rotated pads. Then, the bounding box is constructed to cover all rotated pads of the net by the minimum and maximum

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

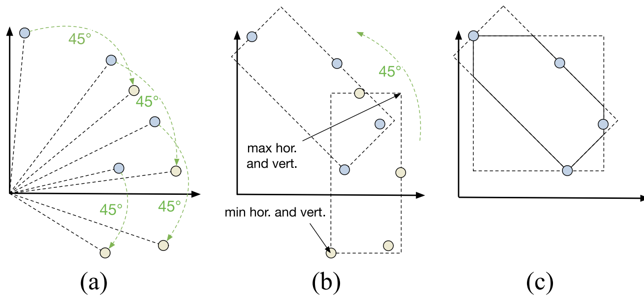

Fig. 9. Our convex hull construction. (a) Each pad of the net is rotated 45° degrees clockwise. (b) Minimum and maximum horizontal and vertical coordinates are found among the centers of all rotated pads. The bounding box is constructed to cover all rotated pads of the net by the minimum and maximum horizontal and vertical coordinates. And it is rotated 45° counterclockwise. (c) Rotated bounding box intersects with the bounding box in 0° to obtain our convex hull.

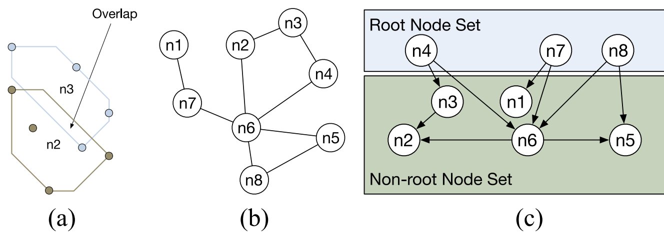

Fig. 10. (a) Overlapping and conflict relationship between nets $n2$ and $n3$. (b) Conflict graph. (c) Task graph.

horizontal and vertical coordinates. And it is rotated 45° counterclockwise, as shown in Fig. 9(b). Finally, as shown in Fig. 9(c), the rotated bounding box intersects with the bounding box in 0° to obtain our convex hull. Our convex hull construction is implemented by Boost C++ library [40]. It allows deploying two connected segments with 135° or 90° angles. According to the nets' convex hulls, the conflict graph is constructed.

Inspired by [39], we build a conflict graph, where each node represents a net n and each edge e represents the conflicted relationship between two nets (nodes). For example, as shown in Fig. 10(a) and (b), there is an edge between nodes $n2$ and $n3$ since their convex hulls have overlapping. In other words, nodes $n2$ and $n3$ have a conflicted relationship. We denote a conflict graph as $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the node (net) set and $\mathcal{E}$ is the edge (conflicted relationship) set.

According to this conflict graph, a task graph needs to be constructed to improve routing parallelism. The task graph construction can be divided into two steps: 1) root node set construction and 2) net order determination. The former constructs a root node set, where any two nodes have no conflicted relationship. The latter determines a net routing order starting from root nodes. According to this conflict graph, we extract one root node batch from this conflict graph. We split all nodes into two parts: 1) the root nodes and 2) the nonroot nodes. Different from the previous work [39], where the root nodes are incrementally obtained from all nodes, we adopt a greedy-based method to remove nonroot nodes from all nodes. This strategy can select more root nodes to improve the routing parallelism. The proposed greedy-based method is shown in

Algorithm 1 Greedy-Based Root Node Set Construction

Require: All nodes $\mathcal{N}$ and all conflicted relationships $\mathcal{E}$;
1: Initialize the root node set $\mathcal{R} \leftarrow \mathcal{N}$;
2: **for** each $e = (n_i, n_j) \in \mathcal{E}$ **do**
3: **if** $n_i \in \mathcal{R}$ and $n_j \in \mathcal{R}$ **then**
4: Remove n_i from $\mathcal{R}$;
5: **end if**
6: **end for**
7: **return** the root node set $\mathcal{R}$.

Algorithm 1. We input all nodes $\mathcal{N}$ and all conflicted relationships $\mathcal{E}$, i.e., all edges. At the initial stage, the root node set $\mathcal{R}$ is initialized by all nodes $\mathcal{N}$. Then, we traverse all conflicted relationships $e \in \mathcal{E}$. If the two nodes have a conflicted relationship and they are both in the root node set $\mathcal{R}$, one of them is removed. The algorithm outputs the root node set $\mathcal{R}$.

In addition to the root nodes, we customize the previous work [39] and define the net order by the two rules: if one node is in the root set and the other is not, the order is from the node in the root set to the other; if the two nodes are not in the root batch, the order is from the node with a larger variance of all gradient values within its convex hull to the other. The main idea behind the strategy is that more freedom is provided to route the net when the net is located in an area with a large discrepancy in thermal sensitivity. Because the variance of gradients represents the thermal sensitivity discrepancy. By using the two rules, net order can be determined. Fig. 10(c) shows a task graph construction example, where the smaller node ID represents the smaller gradient variance within the net convex hull.

A path search algorithm is performed to route each net. Our TRouter adopts the grid-based A* searching as a path search algorithm. Different from traditional routing, our routing scheme will honor the thermal-driven routing guide by the cost function defined as follows:

$$f(\boldsymbol{c}) = g(\boldsymbol{s}, \boldsymbol{c}) + h(\boldsymbol{c}, \boldsymbol{t}) + \lambda \cdot r(\boldsymbol{c}) \tag{10}$$

where $g(\boldsymbol{s}, \boldsymbol{c})$ is the cost from the source $\boldsymbol{s}$ to the current location $\boldsymbol{c}$ and $h(\boldsymbol{c}, \boldsymbol{t})$ is the estimated cost from the current location $\boldsymbol{c}$ to the target $\boldsymbol{t}$. The cost contains wirelength and via number. λ is a hyperparameter controlling the tradeoff between thermal performance and routing cost. Empirically, we set λ to make the magnitude of $\lambda \cdot r(\boldsymbol{c})$ comparable with the wirelength of each net. Note that the gradients $\partial\psi(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{pv}$ and $\partial\psi(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{sg}$ represent the thermal sensitivity for routing pattern. In other words, the smaller gradient values $\partial\psi(\boldsymbol{x}_{cp}, \boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{pv}$ or $\partial\psi(\boldsymbol{x}_{cp}, \boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{sg}$ have negative effects on temperature increase in the routing step. Thus, $r(\boldsymbol{c})$ is defined as follows to guide thermal-driven routing:

$$r(\boldsymbol{c}) = \begin{cases} \left(\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{pv}}\right)_{\boldsymbol{c}}, & \text{if punch a via at } \boldsymbol{c} \\ \left(\frac{\partial\psi\left(\boldsymbol{x}_{pv}^{(i)}, \boldsymbol{x}_{sg}^{(i)}\right)}{\partial\boldsymbol{x}_{sg}}\right)_{\boldsymbol{c}}, & \text{if deploy a segment at } \boldsymbol{c}. \end{cases} \tag{11}$$

The path search step is shown in Fig. 7. The cost is calculated at each neighborhood grid cell. Then, the cell with

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

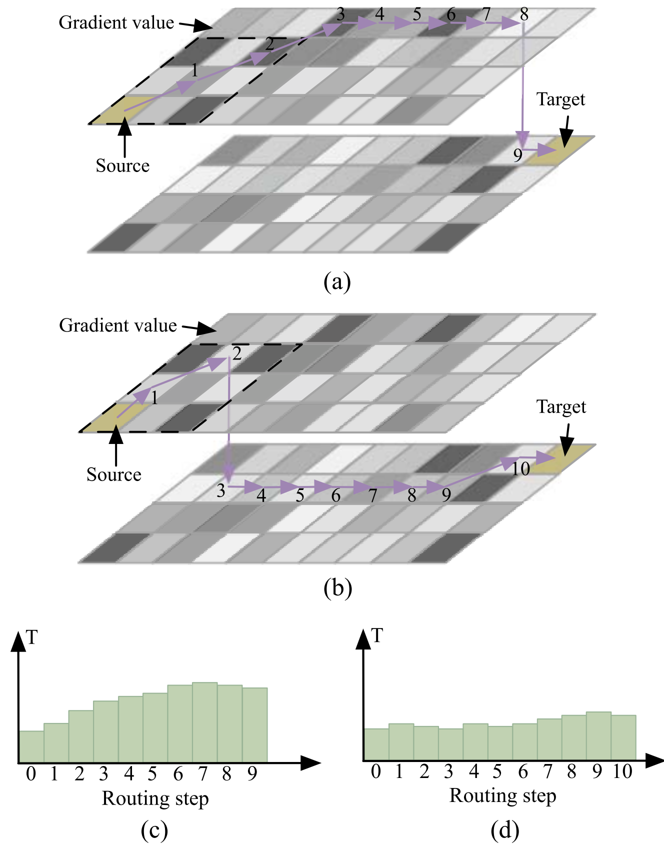

Fig. 11. Routing step without or with a thermal guide and its thermal variation. (a) Routing step without the thermal guide. (b) Routing step with the thermal guide. (c) Thermal variation to the routing step without the thermal guide. (d) Thermal variation to the routing step with the thermal guide. Gray represents the gradient value. The darker gray represents the larger gradient value. The lighter gray represents the smaller gradient value. The dotted rectangle represents the component. T denotes the temperature on PCB.

the minimal cost is selected to place the segment or via. We provide a simple example to illustrate our path search with the thermal-driven routing guide, as shown in Fig. 11. If our thermal model is not used to guide the path search, a shortest path is found to route from source to target, as shown in Fig. 11(a). However, it may bring a higher temperature on PCB if the path passes grid cells with a large gradient, as shown in Fig. 11(c). As shown in Fig. 11(b), if our thermal model guides the path search, the searched path can evade grid cells with a large gradient and tend to pass grid cells with a small gradient. As a result, our path search algorithm can achieve a low-temperature design, as shown in Fig. 11(d).

Based on our task graph and the path search algorithm, we achieve routing parallelism by using Taskflow [41]. Taskflow is a modern C++ tasking toolkit that automatically executes parallel tasks by using the task dependency graph. It can use our task graph as the task dependency graph to perform the tasks in maximum parallelism on a multithreaded CPU.

When all nets are routed, extra iterations with ripping-up and rerouting are performed to reduce the violation number. After the detailed routing is finished, the result is exported into KiCad and OrCAD DSN formats. By using our routing scheme, it is expected to achieve significant speedup and lower-temperature PCB designs.

TABLE I
BENCHMARKS CONSISTENT WITH [10]

Designs	$\mathcal{W} \times \mathcal{H}$ (mm)	$\lvert L\rvert$	$\lvert C\rvert$	$\lvert P\rvert$	$\lvert N\rvert$
PCB1	21×14	2	8	40	15
PCB2	51×23	2	18	77	34
PCB3	55×28	2	34	138	38
PCB4	23×60	2	28	140	52
PCB5	41×42	2	48	163	54
PCB8	57×87	2	36	188	70
PCB9	44×36	2	58	229	80
PCB10	102×54	2	57	319	99
PCB11	89×58	2	64	401	134
PCB12	58×60	4	58	233	35
PCB13	86×72	4	61	314	63

TABLE II
OUR NEW BENCHMARKS

Designs	$\mathcal{W} \times \mathcal{H}$ (mm)	$\lvert L\rvert$	$\lvert C\rvert$	$\lvert P\rvert$	$\lvert N\rvert$
D1	65×55	2	48	190	62
D2	51×23	2	46	207	69
D3	44×45	2	13	118	50
D4	94×63	4	276	1510	272

VI. Experimental Results

A. Implementation Details and Hardware Environment

We implemented our algorithm in the C++ programming language. Boost C++ library [40] is used to calculate the geometric computation within our proposed algorithm. Pytorch library is used to train our CNNs on a Linux machine with 12 cores and NVIDIA Tesla V100 GPU with 32-GB memory. C++ libtorch library is used to perform inference and BP [37]. Routing is performed on a Linux machine with 10 cores and NVIDIA TITAN XP GPU with 12-GB memory. In our TRouter, inference and BP are performed on GPU while other operations are performed on CPU. The routing grid and layout pattern resolutions are set as 0.1 mm. We set the hyperparameter λ as 100. The ripping-up and rerouting are performed five times to reduce the violation number.

B. Benchmark, Baseline Method, and Performance Evaluation

Our benchmarks are from PCBRouter [10] as shown in Table I. These benchmarks are downloaded from [42], where PCB6, PCB7, and PCB14 are missing or inconsistent with [10]. Besides, some designs are used as new benchmarks as shown in Table II, which have been released in https://github.com/plutochen1990/TRouterBM. $\mathcal{W} \times \mathcal{H}$ denotes the PCB width and height dimension. Since the routing grid and layout pattern resolutions are set as 0.1 mm, the relationships between routing grid width (height) and PCB width (height) are $W = 10\mathcal{W}$ ($H = 10\mathcal{H}$). $|L|$, $|C|$, $|P|$, and $|N|$ denote the numbers of PCB layers, components, pads, and nets, respectively.

A state-of-the-art PCBRouter [10], as a baseline, is performed on these benchmarks on the same CPU. Like our TRouter, PCBRouter performs the ripping-up and rerouting

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

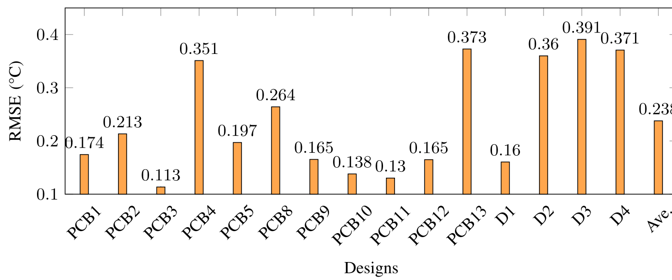

Fig. 12. Model prediction accuracy.

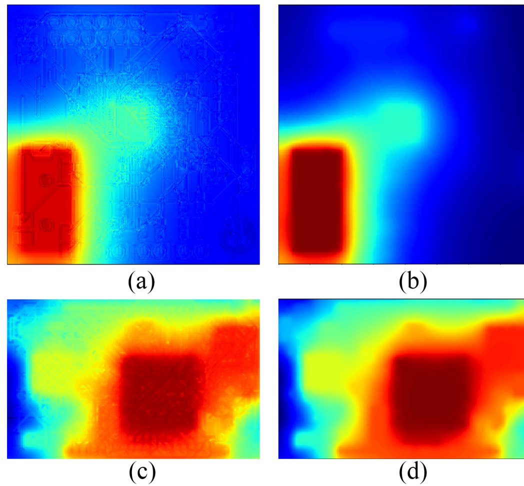

Fig. 13. Thermal distribution heatmap: (a) our model on PCB5; (b) HyperLynx [44] on PCB5; (c) our model on D4; and (d) HyperLynx [44] on D4.

five times to reduce the violation number. And the routing grid resolution is set as 0.1 mm. Other settings are the same as our TRouter. Our TRouter follows PCBRouter [10] to handle the differential pair routing. All differential pair nets can be identified and first routed by PCBRouter and our TRouter. In practice, there are few differential pair nets. Thus, our TRouter performances are not significantly impacted by following PCBRouter to handle the differential pair routing. A differential pair checker is implemented to verify the differential-pair constraint by checking their wirelength and segment directions. Other design rules, including noncrossing, spacing and routing-angle constraints, are verified by KiCad PCB [43].

Siemens HyperLynx software [44] is used to perform the thermal simulation. It simulates the PCB convection, conduction and radiation, and generates a temperature heatmap. Note that each component heat dissipation in our benchmarks is automatically taken from a component datasheet in HyperLynx. This way can mimic the worst conditions when the real heat dissipation from experience is unavailable [45]. In industry, the worst-case can be used as a metric to evaluate reliability and performance. For example, graph-based analysis (GBA), as the worst-case analysis, is widely used in static timing analysis [46], [47].

C. Model Accuracy and Runtime

According to our feature extraction and network structure, the input tensor channel number relies on the PCB layer number. Since there are two layer number types ($|L| = 2$ and $|L| = 4$) in our benchmarks, we need to configure our model as ModelL2 and ModelL4. In other words, as shown in Table III, ModelL2 is used to predict the $|L| = 2$ PCB thermal distribution and ModelL4 is adopted to predict the $|L| = 4$ PCB thermal distribution. ModelL2 input channel is $2|L| + 2 = 6$ and ModelL4 input channel is $2|L| + 2 = 10$. Other configurations are the same in ModelL2 and ModelL4.

In order to evaluate the model performance for each design, we train the model on other designs with the same layer number. For example, testing is performed on PCB1 with ModelL2 pretrained on all 2-layer designs except for PCB1. Likewise, testing is performed on D4 with ModelL4 pretrained on all 4-layer designs except for D4. The same settings are

TABLE III
RELATIONSHIP BETWEEN TWO MODEL CONFIGURATIONS AND DESIGNS

Model config.	#Input channel	Designs	$\|L\|$
ModelL2	6	PCB1, PCB2, PCB3, PCB4, PCB5, PCB8, PCB9, PCB10, PCB11, D1, D2, D3	2
ModelL4	10	PCB12, PCB13, D4	4

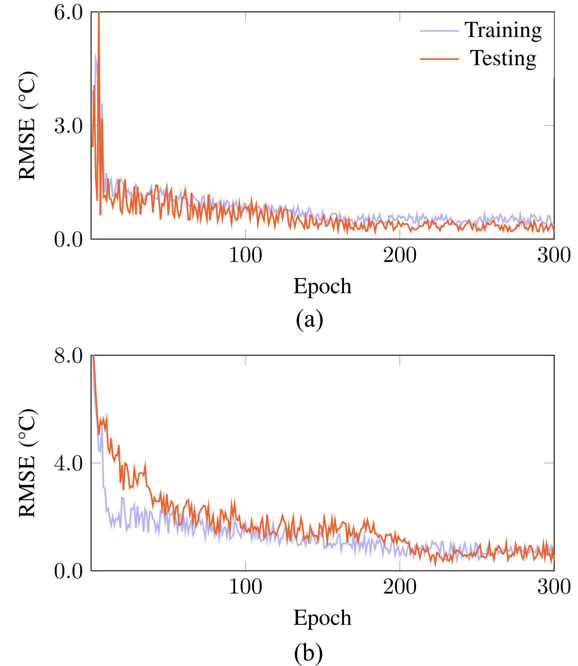

Fig. 14. Training behavior: (a) ModelL2 testing on PCB5 when other 2-layer PCBs are used to train and (b) ModelL4 testing on D4 when other 4-layer PCBs are used to train.

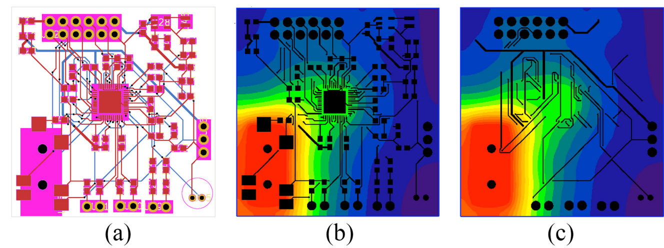

Fig. 15. PCB5 routing result and thermal distribution: (a) routing result and (b) thermal distribution.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

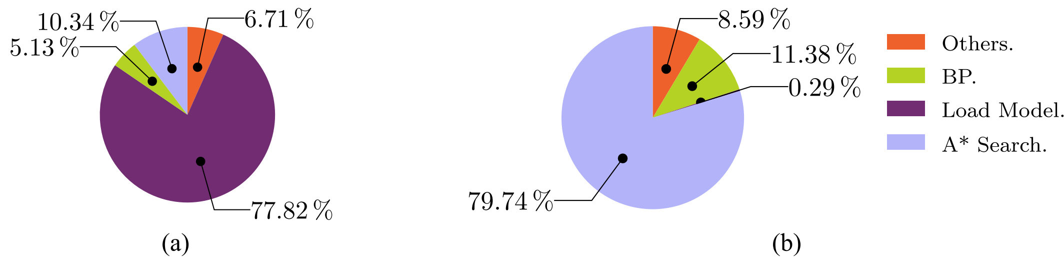

Fig. 16. Runtime profiling: (a) PCB1 and (b) D4.

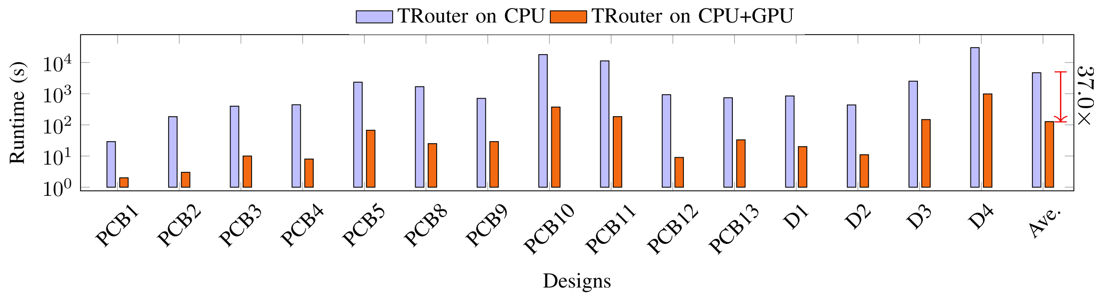

Fig. 17. Runtime of our TRouter on CPU and CPU+GPU.

used to verify our routing performances. HyperLynx can simulate thermal distribution on partially routed, or fully routed PCBs [44], [48]. Thus, in each design, the layout and its thermal distribution in each routing step are collected as the dataset. Moreover, each layout and its thermal distribution are rotated in 90°, 180°, and 270° for data augmentation. In each design, the thermal distribution is normalized by its maximum temperature value to improve our model generalization and learning ability. The training epoch number is 300. The batch size is 8. The learning rate is 10^{-5}. In order to eliminate the overfitting issue, we set the weight decay hyperparameter as 10^{-3}. The root mean-squared error (RMSE) is used to evaluate the error between our prediction and ground truth.

We show our model accuracy in Fig. 12, where RMSEs are shown in the original scale. Our model can achieve accurate estimations since the prediction error is 0.238 °C on average. Fig. 13 shows the thermal distribution heatmap on PCB5 and D4. Our predictions are closer to the ground truth. In order to verify the overfitting issue, we show training behaviors as shown in Fig. 14. PCB5 and D4 are used as testing cases and other 2-layer and 4-layer PCBs are used to train our model, respectively. We can see RMSEs of the training and testing datasets reduce with the epoch number increases. Thus, our model has no overfitting issue.

We compare our model runtime with the HyperLynx software [44]. Note that the HyperLynx software [44] only runs on CPU while our model can run on CPU or GPU. Thus, we report our model runtimes on CPU and GPU and the HyperLynx software [44] on CPU in Table IV. Compared with the HyperLynx software [44], our model on GPU can achieve 504.3× speedup. Compared with CPU, our model on GPU can achieve 5.88/0.03 = 196.0× speedup on average. Thus, this motivates us to deploy our model on GPU to perform inference and BP in our TRouter. In addition, compared with the HyperLynx software, our model can guide thermal-driven routing by thermal-aware wire detour and via punching.

TABLE IV
RUNTIME (S) OF THERMAL DISTRIBUTION PREDICTION

Design	HyperLynx [44]	Our model	
		CPU	GPU
PCB1	13.87	0.49 (28.3×)	0.01 (1387.0×)
PCB2	15.58	2.32 (6.7×)	0.01 (1558.0×)
PCB3	13.88	2.71 (5.1×)	0.02 (694.0×)
PCB4	15.18	2.65 (5.7×)	0.02 (759.0×)
PCB5	14.40	2.90 (5.0×)	0.02 (720.0×)
PCB8	15.00	8.99 (1.7×)	0.04 (375.0×)
PCB9	14.57	2.85 (5.1×)	0.02 (728.5×)
PCB10	14.81	10.63 (1.4×)	0.05 (296.2×)
PCB11	15.27	9.82 (1.6×)	0.04 (381.8×)
PCB12	14.74	3.62 (4.1×)	0.02 (737.0×)
PCB13	15.95	6.34 (2.5×)	0.03 (531.7×)
D1	14.84	6.65 (2.2×)	0.03 (494.7×)
D2	15.35	2.13 (7.2×)	0.01 (1535.0×)
D3	15.43	11.98 (1.3×)	0.07 (220.4×)
D4	18.10	14.10 (1.3×)	0.07 (258.6×)
Ave.	15.13	5.88 (2.6×)	0.03 (504.3×)

D. Routing Performance

We use our model to guide thermal-driven routing. BP is performed for every five nets. After the routing is finished, we input the PCB layout into the HyperLynx software [44] to perform the thermal simulation. As an example, we show the PCB5 routing result and thermal distribution heatmap as shown in Fig. 15. We can see all nets are successfully routed and there are no design rule violations. Moreover, some vias are automatically deployed nearby components to facilitate heat dissipation.

We compare our TRouter with the state-of-the-art PCBRouter [10]. Both PCBRouter [10] and our TRouter can

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

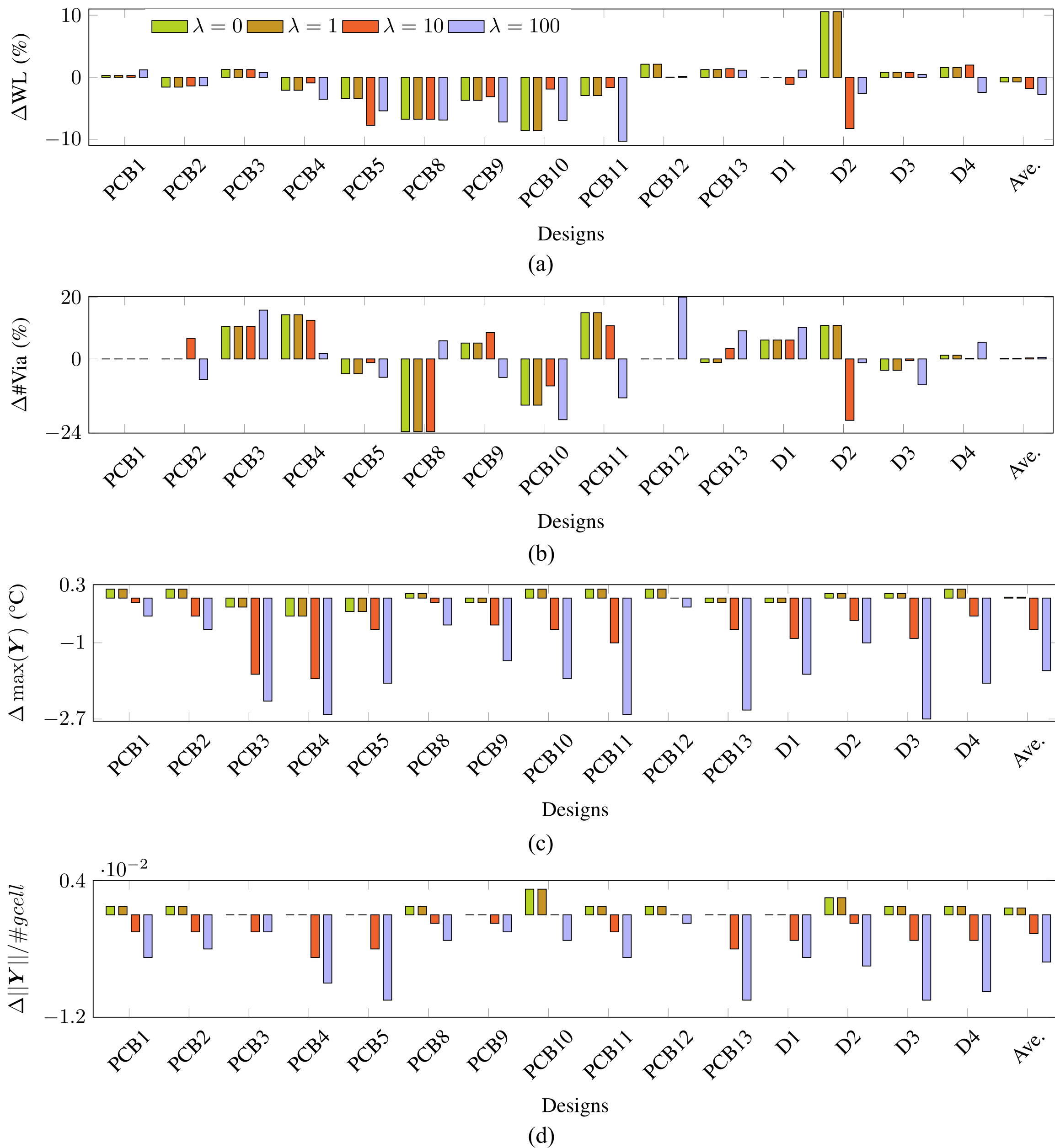

Fig. 18. (a) WL. (b) #Via. (c) max($\boldsymbol{Y}$). (d) $||\boldsymbol{Y}||$/#gcell increase with respect to λ.

achieve 0 design rule violation and 100% routability. The differential pairs with the same routed wirelength and segment directions are achieved by both PCBRouter and our TRouter. Since they are routed by the same method before other nets. Other performances, such as wirelength (WL), via number (#Via), runtime (RT), and thermal performance, are shown in Table V. max($\boldsymbol{Y}$) and $||\boldsymbol{Y}||/\#gcell$ are used to measure thermal performance, where $||\cdot||$ denotes ℓ_2 norm. We can see that our TRouter can beat PCBRouter [10] in both runtime and thermal performance. In particular, our TRouter can achieve 2.7 °C maximum temperature reduction at most since our proposed model is used to guide thermal-driven routing. The temperature has an influence on the lifetime of PCB and its electronic devices. Every 1 °C increase in the temperature reduces the electronic device lifetime by 26 000 h when the temperature range is from 50 °C to 75 °C [49], [50]. According to the HyperLynx Thermal datasheet [48], every 10 °C increase in temperature beyond 100 °C reduces the PCB lifetime by as much as 50%. Thus, our TRouter can significantly extend the lifetime of PCB and its electronic devices.

The typical routing performances are shown in Table V. Compared with PCBRouter [10], thanks to the thermal guide, our TRouter brings less wirelength (reduced by 2.791%) and vias (reduced by 3.398%) on average to reduce temperature. The primary reason behind the phenomenon is that our thermal guide can improve the heat dissipation efficiency for routing by thermal-aware wire detour and via punching. In addition, our TRouter can achieve 2.9× speedup on average because routing is performed in parallel with Taskflow [41] and our task graph.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

TABLE V
ROUTING RESULTS

Design	PCBRouter [10]					Our TRouter				
	WL (mm)	#Via	$\max(\boldsymbol{Y})$ (°C)	$\lVert\boldsymbol{Y}\rVert/\#gcell$	RT (s)	WL (mm)	#Via	$\max(\boldsymbol{Y})$ (°C)	$\lVert\boldsymbol{Y}\rVert/\#gcell$	RT (s)
PCB1	83.300	9	26.6	0.149	3	84.297 (↑1.197%)	9 (↓0.000%)	26.2 (↓0.4)	0.144 (↓0.005)	2 (1.5×)
PCB2	273.917	15	62.0	0.343	1	270.172 (↓1.367%)	14 (↓6.667%)	61.3 (↓0.7)	0.339 (↓0.004)	3 (0.3×)
PCB3	505.534	38	29.1	0.138.	35	509.442 (↑0.773%)	44 (↑15.789%)	26.8 (↓2.3)	0.136 (↓0.002)	10 (3.5×)
PCB4	754.216	56	58.9	0.327	7	727.519 (↓3.540%)	57 (↑1.786%)	56.3 (↓2.6)	0.319 (↓0.008)	8 (0.9×)
PCB5	980.395	84	36.3	0.165	50	927.358 (↓5.410%)	79 (↓5.952%)	34.4 (↓1.9)	0.155 (↓0.010)	67 (0.7×)
PCB8	1233.115	17	85.2	0.386	17	1148.134 (↓6.892%)	18 (↑5.882%)	84.6 (↓0.6)	0.383 (↓0.003)	25 (0.7×)
PCB9	1178.936	117	31.9	0.161	127	1094.128 (↓7.194%)	110 (↓5.983%)	30.5 (↓1.4)	0.159 (↓0.002)	29 (4.4×)
PCB10	5095.385	595	39.9	0.187	246	4740.767 (↓6.960%)	478 (↓19.664%)	38.1 (↓1.8)	0.184 (↓0.003)	373 (0.7×)
PCB11	4140.010	381	31.3	0.154	401	3713.629 (↓10.299%)	333 (↓12.598%)	28.7 (↓2.6)	0.149 (↓0.005)	184 (2.2×)
PCB12	1039.218	5	49.4	0.227	46	1040.660 (↑0.139%)	6 (↑20.000%)	49.2 (↓0.2)	0.226 (↓0.001)	9 (5.1×)
PCB13	1536.219	88	39.3	0.177	62	1553.605 (↑1.132%)	96 (↑9.091%)	36.8 (↓2.5)	0.167 (↓0.010)	33 (1.9×)
D1	956.218	49	28.2	0.144	146	967.265 (↑1.155%)	54 (↑10.204%)	26.5 (↓1.7)	0.139 (↓0.005)	20 (7.3×)
D2	1093.068	166	53.4	0.300	35	1064.590 (↓2.605%)	164 (↓1.205%)	52.4 (↓1.0)	0.294 (↓0.006)	11 (3.2×)
D3	3129.609	191	49.2	0.219	743	3143.539 (↑0.445%)	175 (↓8.377%)	46.5 (↓2.7)	0.209 (↓0.010)	148 (5.0×)
D4	7954.524	1278	51.1	0.287	3627	7760.650 (↓2.437%)	1347 (↑5.399%)	49.2 (↓1.9)	0.278 (↓0.009)	982 (3.7×)
Ave.	1996.911	206	44.8	0.224	370	1916.384 (↓2.791%)	199 (↓3.398%)	43.1 (↓1.7)	0.218 (↓0.006)	127 (2.9×)

E. Runtime Analysis

We profile our TRouter runtime on one of the small cases (PCB1) and one of the large cases (D4) as shown in Fig. 16. When the small design is routed, the runtime is dominated by the model loading as shown in Fig. 16(a). While the large design is routed, the runtime of the model loading can be totally ignored ($\leq$1%) as shown in Fig. 16(b). In addition, thanks to performing BP on GPU, it does not dominate the whole routing runtime.

We also run TRouter on CPU to test its runtime. In other words, our model is deployed on CPU to perform inference and BP in our TRouter. As shown in Fig. 17, compared with CPU+GPU, our TRouter on CPU brings 37.0× runtime speedup since the whole routing runtime is dominated by inference and BP.

F. Hyperparameter λ Analysis

The hyperparameter λ in (10) plays an important role in improving routing performances, including wirelength, via number, and thermal distribution. To test its impact on the performance, we set λ using different values. According to our experiences, all routing performances are not changed when $\lambda \leq 0.1$. There are design rule violations (crossing) or routing failures when $\lambda \geq 1000$. The reason behind the phenomenon is that a too small λ value cannot guide routing while a too large λ value will restrain path search. Thus, we set $\lambda = 0, 1, 10, 100$ to report routing performances in Fig. 18. In these cases, all nets are successfully routed and there are no design rule violations. The wirelength and via number are reported in the relative variation of PCBRouter [10] and thermal performances are reported in the absolute variation of PCBRouter [10].

As shown in Fig. 18(a) and (b), different λ values bring different wirelength and via number. Increasing λ brings the temperature reduction, as shown in Fig. 18(c) and (d). More interestingly, as shown in Fig. 18(a) and (b), increasing λ brings wirelength reduction and via number increase on average. A reason behind the phenomenon is that wirelength reduction and via number increase are potential methods to reduce the temperature. However, good thermal performance still relies on the routing pattern. Thus, this rule is not explicitly reflected in each case, but implicitly reflected in all cases on average.

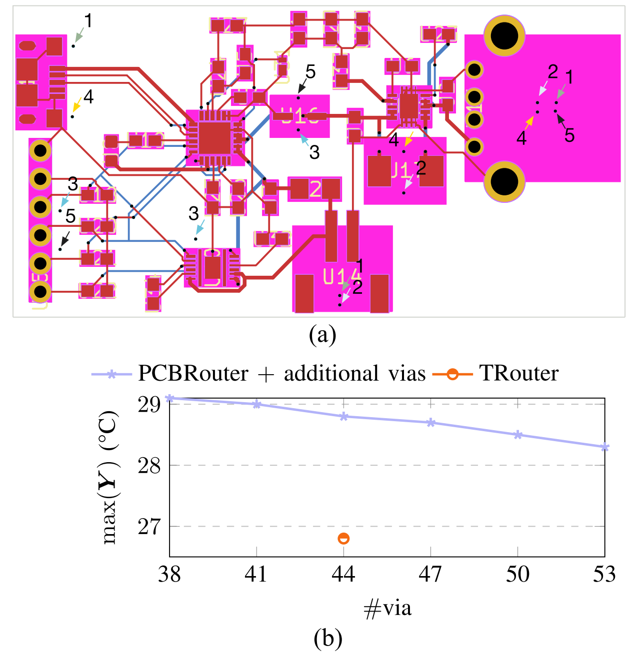

Fig. 19. (a) Layout by PCBRouter [10] with additional vias (the number denotes the punched via batch index). (b) Maximum temperature under different #via values.

G. Dummy via Insertion Versus Our TRouter

The temperature can be reduced by inserting additional dummy nonsignal vias. We compare PCBRouter [10] plus additional vias with our TRouter in PCB3 thermal performance, as shown in Fig. 19. Without any thermal-driven guide, we uniformly punch additional vias nearby components as shown in Fig. 19(a). In each step, we increase three vias as a batch to perform thermal simulation. The number denotes the punched via batch index. According to Fig. 19(b), the temperature reduces with increasing dummy via number. However, due to a lack of a thermal-driven guide, the temperature reduction has a low efficiency with respect to the increasing additional

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

via number. Our TRouter still achieves a lower temperature and fewer vias, compared with inserting 15 dummy vias. This also illustrates that good thermal performance relies on the routing pattern.

VII. Conclusion

In this article, we propose TRouter, a thermal-driven PCB routing framework via an ML model. The model is designed to capture the long-range contextual information from the PCB layout and predict thermal distribution. A gradient in each grid cell obtained from BP is integrated into a full-board routing algorithm to guide thermal-aware wire detour and via punching. To achieve a significant speedup, we construct a conflict graph according to whether overlapping among convex hulls of nets. A greedy-based method is adopted to remove nonroot nodes from all nodes. Then, a task graph is constructed to improve the parallelism. We conduct experiments on open-source benchmarks to illustrate our TRouter can achieve significant speedup and lower-temperature designs, compared with a state-of-the-art PCB routing algorithm. Moreover, the methodology of our TRouter can be easily extended to other performance- or reliability-driven routing frameworks.

References

[1] P. Singh and P. Viswanadham, *Failure Modes and Mechanisms in Electronic Packages*. New York, NY, USA: Springer, 1997.

[2] E. Bogatin, *Signal and Power Integrity–Simplified*. London, U.K.: Pearson Educ., 2010.

[3] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact thermal modeling for temperature-aware design," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2004, pp. 878–883.

[4] J. Cong and Y. Zhang, "Thermal-driven multilevel routing for 3-D ICs," in *Proc. IEEE/ACM Asia–South Pacific Design Autom. Conf. (ASPDAC)*, 2005, pp. 121–126.

[5] T. Yan and M. D. Wong, "Recent research development in PCB layout," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2010, pp. 398–403.

[6] T. Yan and M. D. F. Wong, "Correctly modeling the diagonal capacity in escape routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, pp. 285–293, Feb. 2012.

[7] M. M. Ozdal and M. D. F. Wong, "A length-matching routing algorithm for high-performance printed circuit boards," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2784–2794, Dec. 2006.

[8] T. Yan, P.-C. Wu, Q. Ma, and M. D. Wong, "On the escape routing of differential pairs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2010, pp. 614–620.

[9] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design considering signal skews," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 5, pp. 711–721, May 2010.

[10] T.-C. Lin, D. Merrill, Y.-Y. Wu, C. Holtz, and C.-K. Cheng, "A unified printed circuit board routing algorithm with complicated constraints and differential pairs," in *Proc. IEEE/ACM Asia–South Pacific Design Autom. Conf. (ASPDAC)*, 2021, pp. 170–175.

[11] W. Chu and W. Kao, "A three-dimensional transient electrothermal simulation system for IC's," in *Proc. IEEE Int. Workshop Thermal Investigat. ICs Syst.*, 1995, pp. 201–207.

[12] W. Huang, M. R. Stan, and K. Skadron, "Parameterized physical compact thermal modeling," *IEEE Trans. Compon. Packag. Technol.*, vol. 28, no. 4, pp. 615–622, Dec. 2005.

[13] J. Cong and Y. Zhang, "Thermal via planning for 3-D ICs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2005, pp. 745–752.

[14] T. Zhang, Y. Zhan, and S. S. Sapatnekar, "Temperature-aware routing in 3D ICs," in *Proc. EEE/ACM Asia–South Pacific Design Autom. Conf. (ASPDAC)*, 2006, p. 6.

[15] M. Pathak and S. K. Lim, "Thermal-aware Steiner routing for 3D stacked ICs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2007, pp. 205–211.

[16] M. Pathak and S. K. Lim, "Performance and thermal-aware Steiner routing for 3-D stacked ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 9, pp. 1373–1386, Sep. 2009.

[17] T. Chen, G. L. Zhang, B. Yu, B. Li, and U. Schlichtmann, "Machine learning in advanced IC design: A methodological survey," *IEEE Design Test*, vol. 40, no. 1, pp. 17–33, Feb. 2023.

[18] M. Rapp et al., "MLCAD: A survey of research in machine learning for CAD keynote paper," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 10, pp. 3162–3181, Oct. 2022.

[19] T. Chen, Q. Sun, and B. Yu, "Machine learning in nanometer AMS design-for-reliability," in *Proc. IEEE Int. Conf. ASIC (ASICON)*, 2021, pp. 1–4.

[20] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, "Deep H-GCN: Fast analog IC aging-induced degradation estimation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 7, pp. 1990–2003, Jul. 2022.

[21] W. Jin, L. Chen, S. Sadiqbatcha, S. Peng, and S. X.-D. Tan, "EMGraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 919–924.

[22] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, "Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks," in *Proc. IEEE/ACM Asia–South Pacific Design Autom. Conf. (ASPDAC)*, 2021, pp. 898–903.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[24] T. Chen et al., "An efficient sharing grouped convolution via Bayesian learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7367–7379, Dec. 2022.

[25] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proc. CM/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[26] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[27] H. Geng et al., "Hotspot detection via attention-based deep layout metric learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–8.

[28] B. Zhu et al., "Hotspot detection via multi-task learning and transformer encoder," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–8.

[29] H. Zhou, W. Jin, and S. X.-D. Tan, "GridNet: Fast data-driven EM-induced IR drop prediction and localized fixing for on-chip power grid networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–9.

[30] W. Jin, S. Sadiqbatcha, Z. Sun, H. Zhou, and S. X.-D. Tan, "EM-GAN: Data-driven fast stress analysis for multi-segment interconnects," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2020, pp. 296–303.

[31] Z. Zhou et al., "Congestion-aware global routing using deep convolutional generative adversarial networks," in *Proc. ACM/IEEE Workshop Mach. Learn. CAD (MLCAD)*, 2019, pp. 1–6.

[32] K. Zhu et al., "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2019, pp. 1–8.

[33] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. New York, NY, USA: Springer, 2012.

[34] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3431–3440.

[35] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[36] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNET: Criss-cross attention for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 603–612.

[37] "Pytorch." 2022. [Online]. Available: https://pytorch.org/

[38] C.-K. Cheng, C.-T. Ho, and C. Holtz, "Net separation-oriented printed circuit board placement via margin maximization," in *Proc. IEEE/ACM Asia–South Pacific Design Autom. Conf. (ASPDAC)*, 2022, pp. 288–293.

[39] S. Liu et al., "FastGR: Global routing on CPU-GPU with heterogeneous task graph scheduler," in *Proc. IEEE/ACM Design, Autom. Test Eurpoe (DATE)*, 2022, pp. 760–765.

[40] "Boost." 2022. [Online]. Available: https://www.boost.org/

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.

[41] T.-W. Huang, D.-L. Lin, C.-X. Lin, and Y. Lin, "Taskflow: A lightweight parallel and heterogeneous task graph computing system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1303–1320, Jun. 2022.

[42] "PCB benchmarks." 2021. [Online]. Available: https://github.com/aspdac-submission-pcb-layout/PCBBenchmarks

[43] "KiCad." 2023. [Online]. Available: https://www.kicad.org/

[44] "HyperLynx." 2022. [Online]. Available: https://eda.sw.siemens.com/en-US/pcb/hyperlynx/

[45] (Mentor Graph., Wilsonville, OR, USA). *HyperLynx Thermal User Manual*. (2008). [Online]. Available: http://www.edatop.com/down/faq/pads/pads-pcb-hyperlynx_thermal_user-1784.pdf

[46] (Synopsys, Mountain View, CA, USA). *PrimeTime User Guide*. (2015). [Online]. Available: http://www.synopsys.com/Tools/Implementation/SignOff/Documents/primetime_ds.pdf

[47] (Cadence, San Jose, CA, USA). *Tempus User Guide*. (2015). [Online]. Available: https://www.cadence.com/content/cadencewww/global/en US/home/tools/digital-design-and-signoff/silicon-signoff/tempustiming-signoff-solution.html

[48] "HyperLynx thermal system design Datasheet," Data Sheet, Mentor Graph., Wilsonville, OR, USA, 2007. [Online]. Available: https://www.cadware.cz/getFile/id:1946

[49] P. Lall, M. Pecht, and E. B. Hakim, *Influence of Temperature on Microelectronics and System Reliability: A Physics of Failure Approach*. Boca Raton, FL, USA: CRC Press, 1997.

[50] (EverySpec, Gibsonia, PA, USA). *Military Handbook: Reliability Prediction of Electronic Equipment*. (1991). [Online]. Available: http://everyspec.com/MIL-HDBK/MIL-HDBK-0200-0299/MIL-HDBK-217F_14591/

Tinghuan Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2021.

He is currently an Assistant Professor with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His research interests include machine learning for EDA and deep-learning accelerators.

Silu Xiong received the M.Eng. degree in computer science and technology from Sun Yat-sen University, Guangzhou, China, in 2017.

He is currently a Senior Algorithm Engineer with Huawei, Hangzhou, China. His research interests include automatic placement and routing algorithms in the PCB field.

Huan He received the Ph.D. degree in electronics engineering from Zhejiang University, Hangzhou, China, in 2014.

He then joined Huawei, Hangzhou, where he is currently serving as an Algorithm Expert. He has authored and coauthored more than ten patents and papers in parallel computation, data compression, image retrieval, and EDA, as well as developed the PCB tool.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received nine Best Paper Awards from DATE 2022, ICCAD 2021 & 2013, ASPDAC 2021 & 2012, ICTAI 2019, *Integration, the VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD contest awards. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of IEEE TCCPS Newsletter.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on September 21,2023 at 04:01:05 UTC from IEEE Xplore. Restrictions apply.