Simultaneous Reconnection Surgery Technique of Routing With Machine Learning-Based Acceleration

Peishan Tu, Chak-Wa Pui, and Evangeline F. Y. Young

***Abstract*—In global routing, both timing and routability are critical criteria to measure the performance of a design. However, these two objectives naturally conflict with each other during routing. In this paper, we propose reconnection approaches to fix timing. We first formulated a quadratic program (QP), which adjusts routing topologies of all the nets by only reconnecting critical sinks and takes congestion into consideration to tradeoff timing and routability objectives. A machine learning (ML)-based technique is applied to accelerate our algorithm, which offers a fast and effective way to solve the problem. By exploring more reconnection candidates, we then formulated a QP to reconnect any sink of a net and utilized a multilabel classifier to accelerate the process. The experimental results on ICCAD 2015 benchmarks show that our algorithms can achieve timing improvement with no significant degradation in routability and wirelength. With ML-based acceleration, our results can be obtained in almost negligible runtime.**

***Index Terms*—Global routing, physical design, timing optimization.**

I. Introduction

With the development of semiconductor industry, there is an increasing demand on producing high performance and power efficient chips [1]. A variety of technologies at different design levels, such as architectural, circuit, and material, have been come up to improve the performance of the design. Due to the delay of the transistors, the interconnect draws attentions over years [2]. For example, in the stage of circuit design, demand on improved routing strategies to meet timing requirements keeps increasing [3].

Since global routing plays an important role in both placement and routing phases, there are numerous previous works on global routing. NCTUgr [4] began with rectilinear minimum spanning tree (RMST) topologies and utilized the rectilinear Steiner minimum tree (RSMT) topologies to guide the following monotonic routing and negotiation-based rip-up and reroute. NTHURoute [5] also decomposed nets to two-pin nets based on routing tree topologies. Next, it utilized a rip-up and reroute approach, based on the congested region identification, to further improve routability. FastRoute [6] first built routing trees for all the nets and then adjusted the routing edges to reduce congestion. It then performed multisource multipin maze routing and three-bend routing with an adaptive cost function. MaizeRouter [7] initialized tree construction by FLUTE [8], which is an approach to build RSMT. It then shifted and retracted the edge to optimize congestion. Followed by layer assignment, the maze routing was performed. BoxRouter [9] decomposed nets based on the minimum Steiner tree construction and it proposed an integer linear programming (ILP) routing and an adaptive maze routing to iteratively reduce congestion. Overall, these global routers achieved good performance in terms of wirelength and routing congestion. However, timing is not considered in these routers.

Besides congestion driven routers mentioned above, some works optimized timing and congestion together. The work [10] proposed a global routing approach to incorporate both congestion and timing optimization. It assigned a routing tree topology to each net and adjusted *soft edge*, a nonhorizontal and nonvertical edge, to meet the timing constraint. It also slided Steiner points to achieve better timing. However, it took tens of seconds to handle at most 400 nets, which may not be fast enough for current designs. In addition, it targeted at small number of module nets instead of large number of standard cell nets. The unified timing and congestion optimizing (UTACO) [11] algorithm adopted a shadow price mechanism which considered timing and congestion as the sum of the price. It first built the minimum wire length Steiner tree and performed rip-up and reroute by optimizing the price of congestion and timing. However, it modeled the delay of each net individually which may neglect interaction among adjacent nets because actual gate delay is affected by values from upstream and downstream nets. The work [12] targeted at optimizing chemical–mechanical polishing (CMP) and timing in global routing besides congestion. It modeled timing by a guide of wire density. However, the gate model it considered is the lumped resistance model. The work [13] proposed a routing algorithm that considered timing optimization, buffer insertion, and power reduction. It first constructed minimum Steiner trees and additional detoured trees and buffered trees are then built to reduce congestion and timing. Next, it formulated an ILP to decrease power consumption. However, the buffer tree construction was time consuming and only wire delay was optimized without gate delay considered.

Manuscript received June 5, 2018; revised November 5, 2018 and February 16, 2019; accepted March 27, 2019. Date of publication April 23, 2019; date of current version May 22, 2020. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CUHK 14208914. The preliminary version has been presented at the 28th edition of the ACM Great Lakes Symposium on VLSI (GLSVLSI) in 2018. This paper was recommended by Associate Editors I. Bustany and C. Chu. *(Corresponding author: Peishan Tu.)*

The authors are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: tpeisharon@gmail.com).

Digital Object Identifier 10.1109/TCAD.2019.2912930

0278-0070 © 2019 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: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

In most global routers, a tree topology will be assigned to each net, better timing can be achieved by considering timing in construction of the routing tree topologies. Several algorithms [14], [15] are proposed to build timing-aware routing trees to achieve good performance on balancing net and gate delay. However, most of them adopt simple lumped resistance model as their gate delay model, which is inaccurate and inadequate for modern designs. Moreover, modern gate delay model requires that tree topologies should not be optimized individually. Hence, in timing aware global routers, methods are needed to capture delay more accurately and to consider the topologies of all the nets simultaneously.

As addressed in [16], machine learning (ML)-based approaches have been successfully applied in physical design to improve design performance and reduce design cost of new products in advanced nodes. In general, ML application in physical design aims at modeling new correlation mechanisms and making the design flow more predictable. However, the complicated design characteristics and design flow make it difficult for designers to achieve their goals. Designers take efforts to figure out useful information from the design, which requires experience and knowledge on physical design [17]. There are some works [18], [19] using ML techniques to improve their performance by extracting useful information from the circuit. They could quickly produce similar quality solution, which would otherwise take a long time to be generated. The advantage of ML is fast prediction even though the offline training process may take a long time. In this paper, the fast prediction of ML will be utilized to accelerate the traditional optimization process and useful information will be extracted by analyzing the design characteristics.

In this paper, we propose algorithms to adjust the routing topologies of all the nets to fit timing from a global perspective and consider routing congestion simultaneously. Our contributions are summarized as follows.

1) To optimize the tree topologies globally, a quadratic program (QP) is formulated to determine how to adjust the most critical sink connection to optimize timing and congestion.
2) We study various circuit properties and identified those that contribute to timing. Later, these features will be used to accelerate the QP-based tree surgery technique (QPTST) by an ML-based technique.
3) In order to obtain better solution quality, multiple sinks of each net are considered as our reconnection candidates. Compared with the most critical sink connection, it can achieve better performance on timing.
4) A multilabel classifier is proposed to accelerate our multiple sink QP solver in order to produce solution efficiently.
5) Experimental results show that we can improve timing of the design significantly with small increase in routing congestion.

The remaining of this paper is organized as follows. Section II introduces basic knowledge in timing analysis. Section III defines critical sink reconnection surgery problem (CSRSP) and multiple sink reconnection surgery problem (MSRSP). Section IV contains details of our algorithms on solving CSRSP, including QPTST and ML-based acceleration (MLA) technique. Section V explains how we build a QP solver to solve MSRSP and utilize a multilabel classifier to improve the optimization quality. Section VI shows the experimental results and we finally conclude this paper in Section VII.

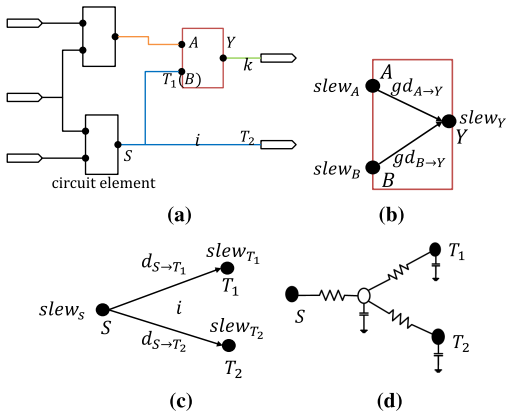

Fig. 1. Delay in the circuit. (a) Simple circuit. (b) Circuit element. (c) Interconnection. (d) RC model.

TABLE I
VARIABLE NOTATIONS IN SECTION II

$d_{A\to B}$	interconnect delay between node A and node B
$gd_{A\to B}$	gate delay from input A to output B
cap_i	total capacitance of net i (including sinks capacitance)
$slew_A$	slew of node A
C_k	lumped capacitance at node k

TABLE II
GATE DELAY LOOKUP TABLE

capacitance \ slew	x_1	x_2	x_3
y_1	z_{11}	z_{12}	z_{13}
y_2	z_{21}	z_{22}	z_{23}
y_3	z_{31}	z_{32}	z_{33}

II. PRELIMINARY

A simple circuit structure is shown in Fig. 1(a). It consists of circuit elements, IO pins, and interconnections. The circuit elements can be combinational logic elements or sequential elements. When signals travel from the primary inputs to the primary outputs of the circuit, the circuit elements and their interconnections will have delays which affect the performance of the circuit. In the following parts, some background is presented. The notations used are shown in Table I.

A. Circuit Element Delay

A circuit element shown in Fig. 1(b) is extracted from Fig. 1(a) which is marked red. It consists of input *A*, input *B*, and output *Y*, which is the source gate of net *k*. Based on a nonlinear gate delay model (NLDM), gate delay $gd_{A\to Y}$ is estimated based on a 2-D table shown in Table II with inputs slew slew_A and capacitance cap_k. Generally, given specific index values *x* and *y*, gate delay can be estimated by (1).

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

Assuming $x_1 < x < x_2$ and $y_1 < y < y_2$, solutions of the bilinear interpolation can be computed as in (1) and the coefficients can be obtained by (2) using z_{11}, z_{12}, z_{21}, and z_{22}. The details of the calculation are shown in [20]

$$L(x, y) = a_0 + a_1 \cdot x + a_2 \cdot y + a_3 \cdot x \cdot y \tag{1}$$

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} z_{11} \\ z_{12} \\ z_{21} \\ z_{22} \end{bmatrix}. \tag{2}$$

B. Interconnect Delay

For interconnect delay, an example is shown in Fig. 1(c). Net i consists of source S and sinks T_1 and T_2. Delay $d_{S\to T_1}$ can be obtained by an RC model as shown in Fig. 1(d). π model is used to represent a wire. Interconnect delay $d_{S\to T_1}$ is computed by

$$d_{S\to T_1} = \sum_{k\in N} R_{k\to T_1} \cdot C_k \tag{3}$$

where N is the set of sinks in net i, $R_{k\to T_1}$ is the total resistance of the common path between the path from S to k and the path from S to T_1, and C_k is the lumped capacitance at node k.

C. Slew Calculation

We make use of the slew calculation as proposed by Hu *et al.* [20]. As shown in Fig. 1, the slew is transited from S to T_1 and T_2. The output slew slew_{T_1} at T_1 is calculated by (4) and it shows that the output slew at a sink is related to the slew at the source slew_S and the impulse response on T_1 imp_{T_1}

$$\text{slew}_{T_1} \approx \sqrt{\text{slew}_S^2 + \text{imp}_{T_1}^2} \tag{4}$$

$$\text{imp}_{T_1} \approx \sqrt{2 \cdot \beta_{T_1} - d_{S\to T_1}^2} \tag{5}$$

$$\beta_{T_1} = \sum_{k\in N} R_{k\to T_1} \cdot C_k \cdot d_{S\to k}. \tag{6}$$

As shown in Fig. 1, slew propagated across a gate is calculated as follows: slew slew_Y is obtained by assuming the worst-slew propagation $\max\{\text{slew}'_A, \text{slew}'_B\}$, where slew'_A is obtained from the lookup table using slew_A and cap_i. It is similar to the calculation of the circuit element delay. slew'_B is calculated in the same way.

D. Timing Analysis

Generally, timing analysis is propagated from the primary input to the output to obtain the actual arrival time (aat) and from the output to the input to obtain the required arrival time (rat). We quantify the timing of a circuit at each node by the term slack which is computed by slack = rat − aat. Static timing analysis (STA) [20] is always performed to check the timing of the design.

E. Global Routing

In global routing, design is divided into an $N \times N$ routing grid. Each edge e of the routing grid has capacity $c(e)$, which is related to the number of available tracks of each edge. Demand $d(e)$ of an edge e is calculated by the number of interconnection that occupies the edge e. Overflow is induced by condition that $d(e) > c(e)$. Global routing aims at reducing overflow of the design.

In order to improve timing, our approach may reconnect some interconnections which may increase congestion. Hence, our optimization flow will consider the congestion issue as well.

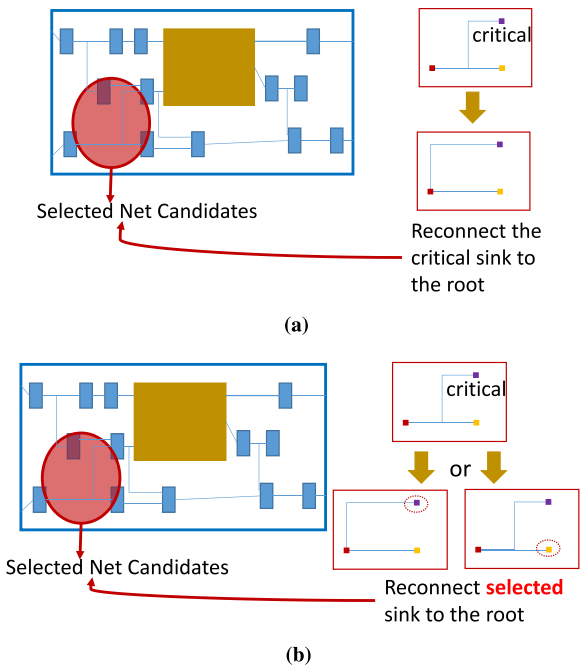

Fig. 2. Examples of problem formulation. (a) CSRSP. (b) MSRSP.

III. Problem Formulations

Given the placement of a design, nets are routed and timing information are obtained by STA. Our objective is to maximize the total negative slack (TNS) by adjusting the routing topologies. The technique is called tree surgery technique (TST). The proposed TST is to reconnect pins of nets to reduce TNS. The following two problems are formulated in order to achieve our goal by different TSTs.

A. Critical Sink Reconnection Surgery Problem

In CSRSP, TST tries to reconnect the critical sink shown in Fig. 2(a) to maximize TNS by reducing wire delay and gate delay on the critical path, which is formulated as

$$\begin{aligned} &\max \quad \text{TNS} \\ &\text{s.t.} \quad x_i \in \{0, 1\} \quad \forall i \in N_c \end{aligned} \tag{7}$$

where x_i denotes whether the most critical sink of each net $i \in N_c$ is reconnected and N_c is a set of net such that any net $i \in N_c$ with P_i sinks should satisfy the following constraints:

$$j_x^i = \underset{l\in P_i, \text{slack}_l<0}{\arg\min} \ \text{slack}_l \tag{8}$$

$$\text{parent}\left[j_x^i\right] \neq s_i \tag{9}$$

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

where j_x^i is the sink in net i whose slack is the most negative and the parent of j_x^i is not the source s_i. If the slack of every sink in net i is positive, the routing topology of net i will remain unchanged. Equation (9) requires that the sink j_x^i with the worst negative slack (WNS) is not connected to the root. It may then be possible to connect it to the root to improve the timing.

The explicit formulation of our objective is explained in Section IV.

B. Multiple Sinks Reconnection Surgery Problem

In MSRSP, TST tends to reconnect one pin among all sinks in each critical net so that TNS can be maximized shown in Fig. 2(b). Its formulation is shown as

$$\begin{aligned} \max \quad & \text{TNS} \\ \text{s.t.} \quad & \sum_{k=0}^{n_i} x_{ik} \le 1 \quad \forall i \in N_c \\ & x_{ik} \in \{0,1\} \quad \forall k \in n_i, i \in N_c \end{aligned} \tag{10}$$

where N_c is a set of nets as defined in Section III-A, n_i denotes number of sinks of net i, and x_{ik} denotes sink k of net i is reconnected or not.

The explicit formulation of our objective is explained in Section V.

IV. Tree Surgery Technique on CSRSP

TST is an approach to modify the tree structure, such as reconnection. TST is first formulated as a QP to maximize the TNS while congestion is also considered in the formulation. We then extract circuit properties which can influence timing and an MLA method is further proposed to speed up the QPTST. The notations of variables in this section are listed in Table III.

A. QP-Based TST

1) Timing Optimization: In order to achieve timing closure, STA is used to detect the timing problem of a design. It measures slack ($\text{slack}_{\text{po}} = \text{rat}_{\text{po}} - at_{\text{po}}$) at each timing end point po $\in$ PO, where PO is a set of primary outputs and register data ports. In STA, timing failure can be detected if the slack of a timing end point is negative ($\text{slack}_{\text{po}} < 0$). In order to reduce timing failure, our objective is to maximize TNS at critical timing endpoints, which is formulated in (11). PO_n denotes the set of timing end points with negative slack

$$\max \sum_{\text{po} \in \text{PO}_n} \text{rat}_{\text{po}} - \text{aat}_{\text{po}}. \tag{11}$$

The negative slacks of PO_n is mainly related to the nets with negative slack sinks, which are called critical nets N_c. Hence, in this paper, we will improve the slack of critical nets instead of directly optimizing the slack on the primary outputs, which is shown in

$$\max \sum_{i \in N_c} \sum_{l \in P_i^c} \text{rat}_l - \text{aat}_l \tag{12}$$

where P_i^c is the set of critical sinks of net i. Since simultaneously optimizing all the critical sinks of a net is hard to achieve and may cause congestion, we further simplify the problem such that only the most critical sink of each critical net will be considered and formulate it as

$$\max \sum_{i \in N_c} \text{rat}_{j_x^i} - \text{aat}_{j_x^i}. \tag{13}$$

Since the slack value on one critical timing path is the same, for each net, optimizing the slack of the source is equivalent to optimize the slack of the most critical sink. Hence, (13) can be transformed into the following equation:

$$\begin{aligned} \max \quad & \sum_{i \in N_c} \text{rat}_{s_i} - \text{aat}_{s_i} \\ & = \max \sum_{i \in N_c} \text{rat}_{j_x^i} - d_{s_i \to j_x^i} - \text{aat}_l - gd_{l \to s_i} \end{aligned} \tag{14}$$

where $d_{s_i \to j_x^i}$ is the net delay from s_i to j_x^i, $gd_{l \to} s_i$ is the gate delay and the input pin l of the gate containing node s_i determines the actual arrival time of source s_i. By assuming $\text{rat}_{j_x^i}$ and aat_l are constant, we can further simplify the problem as

$$\begin{aligned} \min \quad & \sum_{i \in N_c} d_{s_i \to j_x^i} + gd_{l \to s_i} \\ & = \min \sum_{i \in N_c} d_{s_i \to j_x^i} + L\big(\text{cap}_i, \text{slew}_l\big) \end{aligned} \tag{15}$$

where gate delay can be represented as $L(\text{cap}_i, \text{slew}_l)$.

In this paper, we minimize delay $d_{s_i \to j_x^i}$ in (15) by reconnecting the critical sink j_x^i directly to its source s_i. However, reconnecting all the nets N_c will increase total capacitance cap_i of each net i due to longer wirelength, which will increase the gate delay $L(\text{cap}_i, \text{slew}_l)$. In order to maximize the delay reduction $d_{s_i \to j_x^i} + L(\text{cap}_i, \text{slew}_l)$, the set of net that will be reconnected is found by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \Big(\beta \cdot \Delta L_i + \Delta d_{s_i \to j_x^i} \cdot x_i\Big) \\ \text{s.t.} \quad & x_i = \{0,1\} \quad \forall i \in N_c \end{aligned} \tag{16}$$

where x_i is a binary variable indicating whether net i is reconstructed. β is a user-defined parameter. $\Delta d_{s_i \to j_x^i}$ is the difference of interconnect delay on the path from the critical sink j_x^i to its source s_i before and after reconnecting it to the root, which is computed by

$$\Delta d_{s_i \to j_x^i} = d^o_{s_i \to j_x^i} - d_{s_i \to j_x^i}. \tag{17}$$

ΔL_i implies how much gate delay at node s_i can be reduced, which is computed by

$$\begin{aligned} \Delta L_i &= L\big(\text{cap}_i^o, \text{slew}_l^o\big) - L\big(\text{cap}_i, \text{slew}_l\big) \\ &= a_1 \cdot \big(\text{cap}_i^o - \text{cap}_i\big) + a_2 \cdot \big(\text{slew}_l^o - \text{slew}_l\big) \\ &\quad + a_3 \cdot \big(\text{cap}_i^o \cdot \text{slew}_l^o - \text{cap}_i \cdot \text{slew}_l\big) \end{aligned} \tag{18}$$

where $L(\text{cap}_i^o, \text{slew}_l^o)$ and $L(\text{cap}_i, \text{slew}_l)$ are the gate delay before and after reconnection, respectively. The value of ΔL_i is determined by reconnection of net i and net j, where net j influences the input slew at node l. It is easy to see that ΔL_i can be rewritten in the form of summation of terms with x_i

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

TABLE III
VARIABLE NOTATIONS IN SECTION IV-A

s_i	the source of net i
j_x^i	the sink of net i whose slack is negative and the worst
l	the input pin of the gate of s_i which affects actual arrival time of s_i
j	the net of input pin l
$L(cap_i, slew_l)$	gate delay $gd_{l\to S}$ from input l to source node S of net i
$d^o_{s_i\to j_x^i}$	delay from s_i to j_x^i before reconnection on net i
$d_{s_i\to j_x^i}$	delay from s_i to j_x^i after reconnection on net i
$\Delta d_{s_i\to j_x^i}$	delay difference before and after reconnection on net i, $d^o_{s_i\to j_x^i} - d_{s_i\to j_x^i}$
cap_i^o	lumped capacitance of node i before deciding whether performing reconnection on net i
cap_i	lumped capacitance of node i after deciding whether performing reconnection on net i, $cap_i^o - \Delta cap_i x_i$
Δcap_i	capacitance changed when reconnection is performed on net i
$slew_l^o$	slew of pin l before deciding whether performing reconnection on net j
$slew_l$	slew of pin l after deciding whether performing reconnection on net j, $slew_l^o - \Delta slew_l x_j$
$\Delta slew_l$	slew changed when reconnection is performed on net j
ΔL_i	gate delay difference of source s in net i considering reconnection of net i and net j

and x_j as in (19), where a_0, a_1, a_2, and a_3 can be obtained as shown in Section II-A

$$\begin{aligned}\Delta L_i = &\left(a_1 + a_3 \cdot \text{slew}_l^o\right) \cdot \Delta\text{cap}_i \cdot x_i \\ &+ \left(a_2 + a_3 \cdot \text{cap}_i^o\right) \cdot \Delta\text{slew}_l \cdot x_j \\ &- a_3 \cdot \Delta\text{cap}_i \cdot \Delta\text{slew}_l \cdot x_i \cdot x_j. \end{aligned} \tag{19}$$

With (16) and (19), we can formulate a QP to determine which net to be reconnected such that the TNSs is optimized.

2) Congestion Optimization: When we improve the timing by reconnecting sinks to their sources, routing congestion may be increased. Hence, routability needs to be considered when we optimize timing by reconnection. The general idea is to avoid reconnecting the critical sink which may go through congested routing regions.

More specifically, a penalty factor of each critical sink is obtained and such penalty will be added to the objective function in order to consider routability. The penalty factor of each critical sink is calculated from the overflow values of its source, which can be obtained after global routing. To honor our original tree topologies, both Steiner points and pins of each tree are treated as pins in the global router. How we calculate the overflow penalty po_i of critical sink i is illustrated by an example given in Fig. 3, where a, b, and c are the possible locations of the critical sink and s is the source. For each critical sink i, po_i can be obtained from the overflow values of its source, which are o_{e_u}, o_{e_r}, o_{e_d}, and o_{e_l} in this example. There are two kinds of situations as follows.

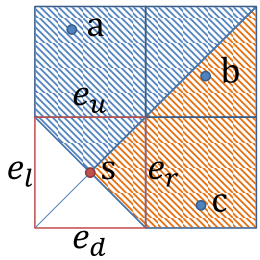

Fig. 3. Example of how to calculate potential routing overflow.

1) The routing grid of the critical sink is either horizontal or vertical to the one of its source, such as critical sinks a, c.
2) Otherwise, such as critical sinks b.

For the first situation, po_i is equal to the overflow of the edge cut through by the connection between the sink and source. For the other situation, po_i is equal to the maximum overflow of the edges cut through by the bounding box of the sink and source. In Fig. 3, the overflow penalties of a, b, and c are o_{e_u}, $\max(o_{e_u}, o_{e_r})$, and o_{e_r}, respectively.

By adding overflow penalty po_i into the objective function as shown in (20), we can optimize timing and congestion simultaneously

$$\begin{aligned}\max\ &\sum_{i=1}^{n}\left(\beta \cdot \Delta L_i + \left(\Delta d_{s\to j_x^i} + \alpha \cdot \text{po}_i\right) \cdot x_i\right) \\ \text{s.t.}\ &x_i = \{0, 1\} \quad \forall i \in N_c. \end{aligned} \tag{20}$$

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
DELAY AND SLEW RELATED FEATURES

DiffTopSlew (*DiffTopDelay*)	The difference of sink j_x^i slew (delay) and the largest slew (delay) of net i except slew (delay) of sink j_x^i.
IsWorstSlew (*IsWorstDelay*)	Whether slew (delay) of sink j_x^i is worst in net i.
WorstSlew (*WorstDelay*)	Value of the worst slew (delay) of net i.
TargetSinkSlew (*TargetSinkDelay*)	The slew (delay) of sink j_x^i.
TargetDeltaSlew (*TargetDeltaDelay*)	The difference slew (delay) of sink j_x^i before and after connecting to root.
CommonPathDelay	The delay accumulated on branches which connects s to j_x^i path.

TABLE V
DISTANCE AND LENGTH RELATED FEATURES

SourceSinkPathDist	The path length from source to the sink j_x^i on the routing tree.
SourceSinkDist	The Manhattan distance between the position of source s and sink j.

TABLE VI
PHYSICS RELATED FEATURES

TotalNetCap	The total capacitance of net i.
TotalNetRes	The total net resistance.
a_0,a_1,a_2 and a_3	The coefficients of lookup table function of source s.

B. Machine Learning-Based Acceleration

In this section, we first study how the circuit properties will affect the reconnection decisions. For example, the critical sink may have a large detour to the source in the original tree topology and the slew of the critical sink will be improved a lot after reconnection. We will select some of these properties as features and use a classification approach to speed up QPTST. In the following parts, we assume each net i has n sinks and a source s_i and sink j_x^i is the most critical one.

The circuit properties we study can be categorized into three types: 1) slew and delay related features as shown in Table IV; 2) distance and length related features as shown in Table V; and 3) physics related features as shown in Table VI.

Since the ranges of the values of the extracted features $\boldsymbol{f}$ vary a lot, (21) is used for normalization

$$\boldsymbol{f}' = \frac{\boldsymbol{f} - \min(\boldsymbol{f})}{\max(\boldsymbol{f}) - \min(\boldsymbol{f})} \tag{21}$$

where $\max(\boldsymbol{f})$ and $\min(\boldsymbol{f})$ are obtained in the training set.

Large number of features may cause inefficiency and overfitting. Hence, we need to reduce the number of features and features are selected according to their importance. The importances of features are obtained by an ML model. As shown in Fig. 4, features are ranked according to their importance produced by random forest (RF) [21].

After feature selection and preprocessing, we formulate our MLA problem as a classification problem. To be specific, a classifier is applied to each critical net to decide whether it should be reconnected and the results of QPTST are used as the golden results. We use the RF as our classification model and the top 15 important circuit properties shown in Fig. 4 are selected as final features. If congestion is also considered, factor po_i of each net i is added to our features during classification.

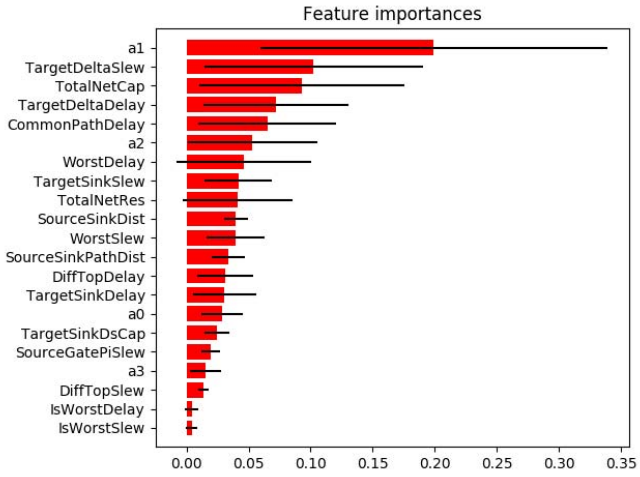

Fig. 4. Feature importances.

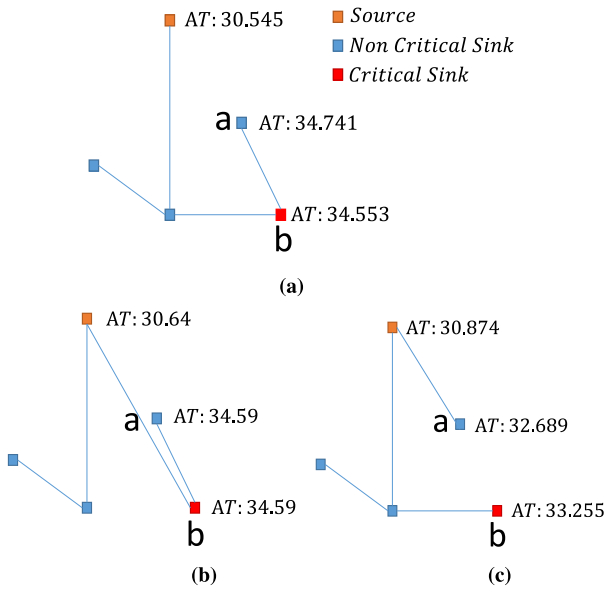

Fig. 5. Example of reconnection. (a) Net. Reconnection on the (b) critical sink and (c) noncritical sink.

V. MULTIPLE SINKS OPTIMIZATION ON MSRSP

In Section IV, only the most critical sink in each net is considered to be reconnected. However, only critical sink reconnection may restrict the improvement of solution quality.

As shown in Fig. 5, a routed net extracted from a design with arrival time information is shown in Fig. 5(a). AT denotes arrival time. Pin b is a critical sink and net delay on the path from source to critical sink b is $34.553 - 30.545 = 4.008$. Our target is to improve its timing. If only the critical sink is considered to be reconnected, arrival time could be improved as shown in Fig. 5(b). Due to wirelength increased by reconnection, total capacitance of that net is also increased. Therefore, gate delay at source is increased and the arrival time at source is also increased. Net delay on the path from source to critical sink is reduced to 3.95 by removing subtree of the path. Overall, arrival time of critical sink is increased and such reconnection may not be adopted. However, shown in

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

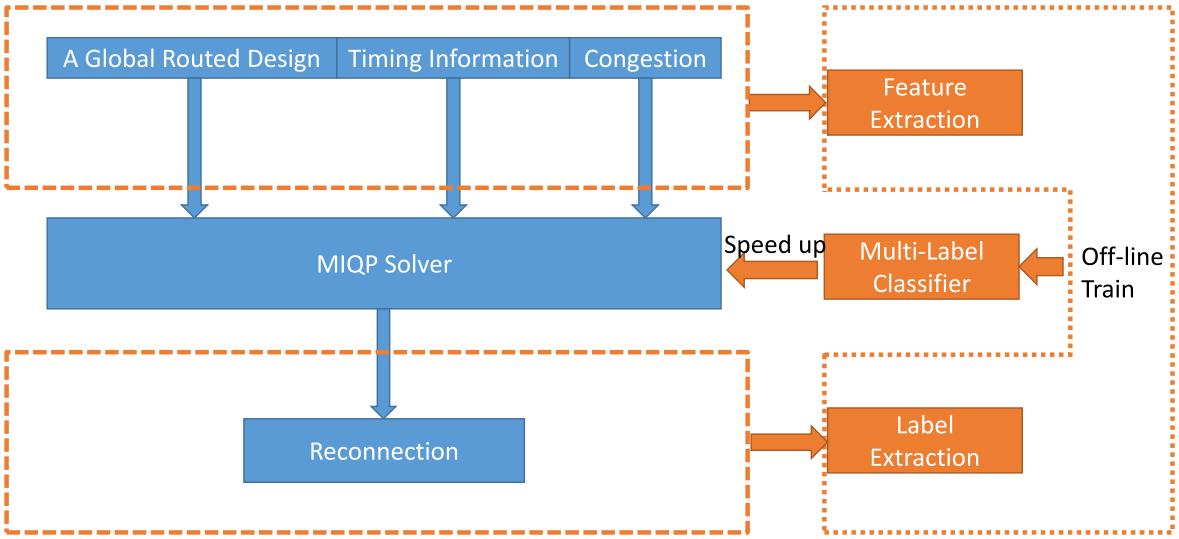

Fig. 6. Flow of Section V.

Fig. 5(c), noncritical sink is reconnected. With 0.329 gate delay increased, net delay of the path from source to critical sink b is only 2.381. The timing of critical sink b is improved while even timing of noncritical sink a is also improved. Observed from Fig. 5, more possibilities considered during reconnection optimization may lead to better performance.

In order to obtain better timing and keep routing topology less change, a more flexible optimization is to take all pins of a net as reconnected candidates and select at most one pin of each net to be reconnected, as formulated in Section III-B.

The MSRSP problem is quite different from the other. Approaches in Section IV cannot be applied directly to solve the problem. The proposed algorithm in Section IV can be extended to solve the MSRSP problem, as shown in Fig. 6. A QP is formulated to solve the problem described in Section III-B. Then, a multilabel classification-based algorithm is proposed to accelerate the QP optimization approach.

A. Multiple Sinks Optimization QP

According to the deduction in Section IV-A, $\max \sum_{i\in N_c} \Delta d_{s_i\to j_x^i} + \Delta L(\text{cap}_i, \text{slew}_l)$ formulated in (15) is our objective and the purpose is to maximize the delay reduction on the critical paths before and after reconnection in the design. In Section IV-A, net delay reduction $\Delta d_{s_i\to j_x^i}$ in (15) is minimized by reconnecting the critical sink j_x^i directly to its source s_i. When more sinks are considered to be reconnected, delay reduction $\Delta d_{s_i\to j_x^i}$ and gate delay reduction ΔL will be expressed by considering these sinks reconnection. The explicit formulation is shown in the following equation:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\left(\beta\cdot\Delta L_i + \sum_{k=0}^{n_i}\Delta d^k_{s_i\to j_x^i}x_{ik}\right) \\ \text{s.t.} \quad & \sum_{k=0}^{n_i} x_{ik} \le 1 \quad \forall i \in N_c \\ & x_{ik} = \{0,1\} \quad \forall k \in n_i,\ \forall i \in N_c. \end{aligned} \tag{22}$$

Assume there are n_i variables for each net i, representing n_i sinks of net i. Variable $x_{ik}(\forall k \in n_i)$ is introduced to represent whether pin k in net i is reconnected or not. Constraint $\sum_{k=0}^{n_i} x_{ik} \le 1$ denotes at most one sink in each net could be reconnected. In objective function, $\Delta d^k_{s_i\to j_x^i}$ denotes the delay change in the path from sink j_x^i to source s when sink k of net i is reconnected.

Accordingly, $\Delta\text{cap}_i = \Delta\text{cap}_{ik}x_{ik}$ represents capacitance variation of net i.

Derived from Δcap_i and (19), ΔL_i can be written as

$$\begin{aligned} \Delta L_i = {} & \left(a_1 + a_3\text{slew}_i^o\right)\sum_{k=0}^{n_i}\Delta\text{cap}_{ik}\cdot x_{ik} \\ & + \left(a_2 + a_3\text{cap}_i^o\right)\Delta\text{slew}_l^p\cdot x_p \\ & - a_3\sum_{k_1=0}^{n_i}\Delta\text{cap}_i^{k_1}x_{ik_1}\Delta\text{slew}_l^p\cdot x_p \end{aligned} \tag{23}$$

where x_p denotes whether the critical sink of net j is reconnected or not. Compared with formulation in Section IV, $N = \sum_{i=0}^{|N_c|} n_i$ variables are used in optimization instead of $|N_c|$ and the matrix is not that sparse in our formulation. The solving process could take longer time.

Congestion optimization formulated in (24) is similar to that in Section IV-A2. According to different sink of net i, overflow penalty factor po_i^k can be extracted based on overflow of routing grid edge, which reconnecting sink k to source may go across. Then penalty factor po_i^k of each variable x_{ik} is added in the objective function accordingly

$$\max \sum_{i=1}^{n}\left(\beta\cdot\Delta L_i + \Delta d^k_{s_i\to j_x^i}\cdot x_{ik}\right) + \sum_{k=0}^{n_i}\alpha\cdot\text{po}_i^k\cdot x_{ik}. \tag{24}$$

B. Multilabel Classifier

In this section, how ML-based approach can be applied to speed up the multiple sink optimization QP process is explained. The binary classification task in Section IV-B is

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

Fig. 7. Composition of data.

TABLE VII
SINGLE PIN FEATURES

PinDeltaCap	The difference in capacitance of sink j_x^i before and after pin k in net i is connected to root.
PinDeltaDelay	The difference in delay of sink k before and after pin k in net i is connected to root.
PinDeltaNetCap	The difference in net capacitance of sink k before and after pin k in net i is connected to root.
PinDeltaSlew	The difference in slew of sink k before and after pin k in net i is connected to root.

to figure out whether the net will be reconnected to its critical sink. Hence, the feature and label data of that problem is extracted based on net information. Different from binary classification task of Section IV-B, our task of problem Section III-B is to obtain the result of whether the pin of each net is reconnected or not. The feature and label extraction should be based on each single pin. Meanwhile, the net information should also be collected to keep the data integrity. The composition of data is shown in Fig. 7. Applying similar process as described in Section IV-B, feature information will be extracted from the design before an MIQP solver is invoked and the MIQP results will be assigned as the labels to the data. The whole flow is drawn in orange in Fig. 6.

1) Data Construction: Due to difference of problems in Sections III-A and III-B, features of Section III-B should not only reflect net information but also identify pin property of reconnection. Therefore, besides the features proposed in Section IV-B (net-based features in Fig. 7), pin property related features (pin-based features in Fig. 7), which are shown in Table VII, are also extracted from the design. When sink *k* of net *i* is reconnected, the change in delay, capacitance, and slew of the most critical sink of net *i* could be calculated. Net capacitance change also varies when different sink *k* is reconnected. In another word, our pin-based data consists of two parts: 1) net features proposed in Section IV-B and 2) pin features described in Table VII. Pin-based data of the same net will have the same net features but various pin features.

If our multiple sinks optimization QP are transferred to binary classification problem directly, net reconnection information will be lost. To extract label of our optimization, one way is to model it as multiclass classification. The different net and pin reconnection information can be combined to four classes. The other way to calibrate data is to encode it with two binary labels in order to distinguish its class more accurately. In this paper, the second way is adopted and two binary labels are net and pin label. Net label represents whether the net of pin *i* is reconnected or not. Pin label represents whether the pin *i* is reconnected or not. Net labels of sinks in the same net are same while their pin labels may vary. Net label together with pin label could guide the model to predict a more accurate result. In summary, pin-based sample owns two labels: 1) net and 2) pin reconnected label.

2) Optimization Procedure: Data preprocessing described in Section IV-B is performed. The feature importance figure

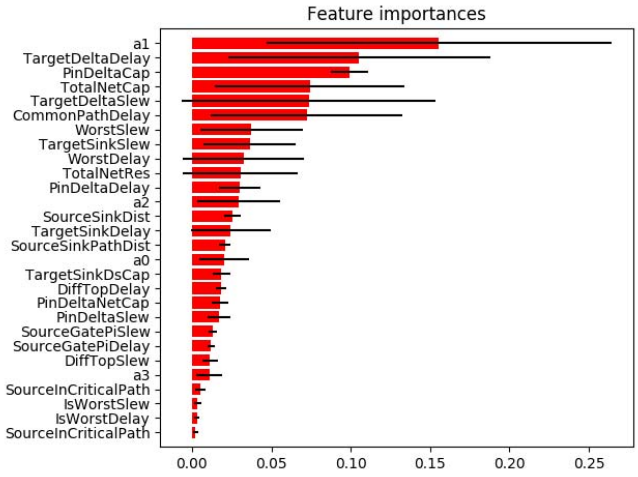

Fig. 8. Feature importances.

TABLE VIII
ICCAD 2015 BENCHMARKS INFORMATION

Designs	#nodes	#nets	clock periods (ns)
superblue1	1209716	1215710	9
superblue3	1213253	1224979	10
superblue4	795645	802513	6
superblue5	1086888	1100825	9
superblue7	1931639	1933945	5.5
superblue10	1876103	1898119	10
superblue16	981559	999902	5.5
superblue18	768068	771542	7

of our problem is shown in Fig. 8. We also only selected top 15 features and used RF as our multilabel classifier.

VI. EXPERIMENTAL RESULTS

In the experiments, the benchmarks of the contest in ICCAD 2015 [22] are used and these benchmarks provide timing information. Details of the benchmark are shown in Table VIII. OpenTimer [23] is used for STA. This paper is implemented in C++ and tested on a 2.1-GHz Intel Linux machine with a 64-GB memory. IBM ILOG CPLEX V12.7.0 [24] is used to solve the QP.

A. Experiments on CSRSP Problem

1) QPTST Results:

a) Timing results: The results of QPTST is shown in Table IX. Evaluation is performed by OpenTimer [23] and our results provide interconnection information to it. FLUTE baseline shows timing results when all the nets are constructed by FLUTE, which does not optimize timing. Direct connection is the experiment that the most critical sink of every net is reconnected to its source. It can shorten path length of all the nets but will increase wirelength a lot. QPTST is our result and congestion-aware QPTST is the algorithm described in Section IV-A2. β is set to 2500. QPTST takes 27.6 s while congestion-aware QPTST takes 30.71 s on average. *r_wns* and

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

TABLE IX
EXPERIMENTAL RESULTS OF TST

Benchmarks	FLUTE Baseline**						Direct Connection*			QPTST					Congestion Aware QPTST				
	WNS	r_wns	TNS	r_tns	stWL	r_stwl	r_wns	r_tns	r_stwl	r_wns	r_tns	r_stwl	r_d	CPU(s)	r_wns	r_tns	r_stwl	r_d	CPU(s)
superblue1	-0.50	1.00	-0.46	1.00	0.96	1.00	-0.26%	3.20%	-19.76%	1.76%	7.92%	-0.38%	16.18%	15.60	1.78%	4.81%	-0.37%	16.19%	25.05
superblue3	-1.01	1.00	-1.50	1.00	1.14	1.00	4.82%	5.79%	-18.87%	5.61%	7.16%	-0.11%	15.78%	6.12	5.37%	6.76%	-0.09%	15.80%	15.10
superblue4	-0.62	1.00	-3.47	1.00	0.71	1.00	0.90%	10.81%	-18.51%	1.60%	15.33%	-1.79%	14.10%	48.95	0.47%	15.19%	-1.58%	14.29%	57.36
superblue5	-2.57	1.00	-6.95	1.00	1.08	1.00	0.07%	1.42%	-17.24%	0.32%	4.17%	-0.62%	14.18%	11.93	0.12%	2.29%	-0.27%	14.48%	22.02
superblue7	-1.51	1.00	-1.84	1.00	1.40	1.00	0.00%	3.54%	-23.56%	0.00%	6.46%	-0.13%	18.96%	44.13	0.00%	3.21%	-0.08%	19.00%	20.91
superblue10	-1.65	1.00	-33.10	1.00	2.05	1.00	0.47%	2.47%	-13.92%	0.92%	3.88%	-0.79%	11.52%	69.73	0.00%	2.11%	-0.48%	11.80%	77.03
superblue16	-0.46	1.00	-0.76	1.00	0.93	1.00	3.58%	25.18%	-14.74%	3.94%	31.58%	-0.38%	12.52%	6.42	3.94%	29.57%	-0.36%	12.54%	15.21
superblue18	-0.46	1.00	-1.03	1.00	0.58	1.00	-0.75%	2.10%	-23.30%	2.27%	4.45%	-0.18%	18.75%	17.93	2.27%	4.45%	-0.18%	18.75%	13.01
Average	-1.10	1.00	-6.14	1.00	1.11	1.00	1.10%	6.81%	-18.74%	2.05%	10.12%	-0.55%	15.25%	27.60	1.74%	8.55%	-0.43%	15.35%	30.71

*Direct Connection: directly connect the critical sinks to the source for all nets.
**WNS is in $10^4 ps$. TNS is in $10^6 ps$. stWL is in $10^8 um$.

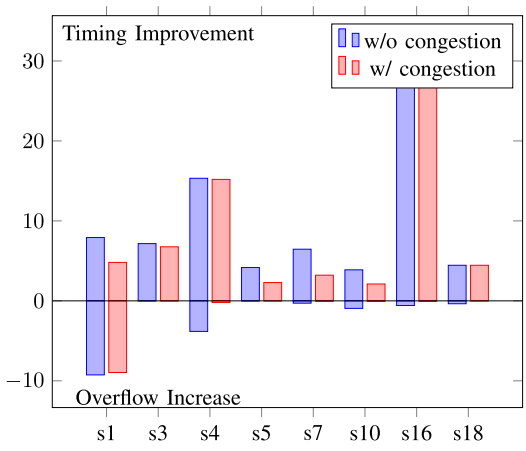

Fig. 9. Performance analysis on timing and routing congestion.

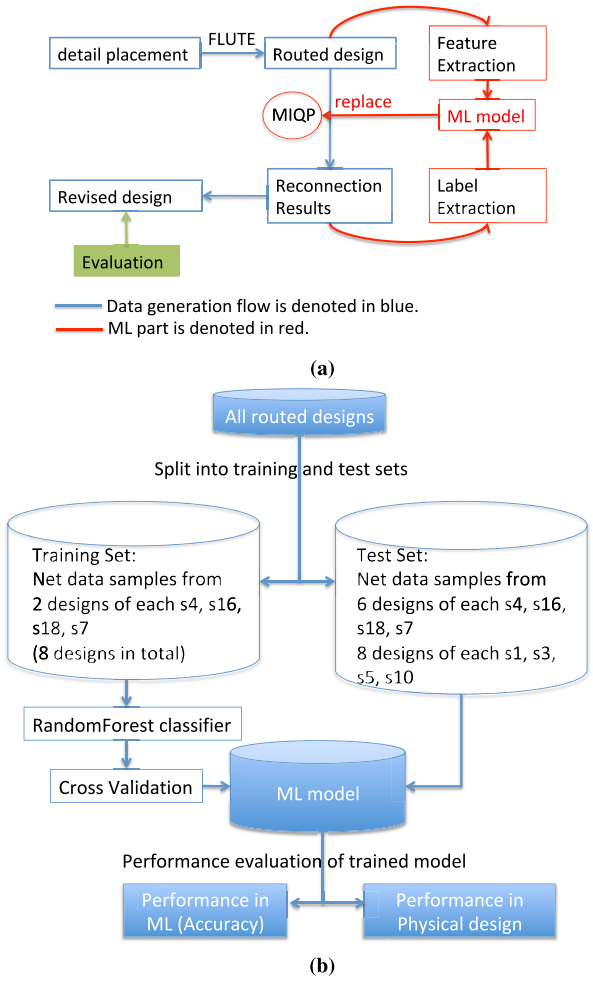

Fig. 10. MLA. (a) Data preparation and application flow. (b) ML flow.

r_tns denote WNS and TNS improvement over the FLUTE baseline. *r_stwl* and *r_d* denote tree length improvement over the FLUTE baseline and direct connection. Direct connection improves timing by 1.10% on WNS and 6.81% on TNS. However, it increases 18.74% wirelength as expected. QPTST obtains 2.05% and 10.12% improvement on WNS and TNS. The wirelength is only worse by −0.55%. If we also consider congestion in the objective function, timing is not as good as QPTST but can still get 1.74% and 8.55% improvement on WNS and TNS. The wirelength is improved compared with QPTST. All the results of our algorithms achieve better wirelength and timing compared with the FLUTE baseline and direct connection.

Besides using FLUTE as baseline, we also performed QPTST experiments on timing driven routing tree constructed by the PD method mentioned in [14]. Shown in Table X, our algorithm achieves 1.7% and 7.11% improvement on WNS and TNS with loss of 0.39% wirelength. It shows that our algorithm is efficient on both nontiming aware trees and timing driven routing.

b) Congestion results: Nets are decomposed to two-pin nets by FLUTE first and NCTUgr [25] is performed to measure congestion. As shown in Table XI, *r* denotes the improvement compared with FLUTE-based net decomposition. The overflow of QPTST and congestion aware QPTST increases 1.91% and 1.15%, respectively. Wirelength increases 1.29% and 0.79%, respectively. In congestion aware TST, α is set to 1000. Our algorithm QPTST can improve timing around 10% but congestion is increased by 1.91%. The outlier is due to rough congestion model. With parameter tuning, a better overflow result can be got but timing is not improved a lot. Hence, we keep results with this parameter to show better timing improvement.

c) Analysis: Fig. 9 shows the analysis of performance of our algorithms on timing and congestion. The part above 0 of the chart is timing improvement and the other part is plotted as increase of overflow. It is obvious that our algorithms can achieve significantly timing improvement with small increase in congestion.

2) Machine Learning-Based Acceleration Results: The data preparation and application flow shown in Fig. 10(a) is described as follows: given the detail placement, we routed the

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

TABLE X
COMPARISONS BETWEEN PD-BASED TREE CONSTRUCTION AND QPTST

Benchmarks	PD Baseline						Direct Connection			QPTST				
	WNS	r_wns	TNS	r_tns	stWL	r_stwl	r_wns	r_tns	r_stwl	r_wns	r_tns	r_stwl	r_d	CPU(s)
superblue1	-0.50	1.00	-0.47	1.00	1.01	1.00	-0.86%	1.53%	-14.78%	0.90%	5.46%	-0.24%	12.67%	14.14
superblue3	-1.01	1.00	-1.51	1.00	1.22	1.00	2.02%	2.90%	-14.80%	2.71%	4.15%	-0.07%	12.84%	5.91
superblue4	-0.63	1.00	-3.54	1.00	0.75	1.00	-0.63%	9.66%	-14.50%	-0.02%	13.20%	-1.29%	11.53%	39.80
superblue5	-2.57	1.00	-6.95	1.00	1.11	1.00	-0.18%	0.12%	-13.43%	0.10%	2.91%	-0.45%	11.44%	12.08
superblue7	-1.52	1.00	-1.81	1.00	1.51	1.00	0.00%	1.06%	-18.69%	0.56%	5.24%	-0.09%	15.67%	11.46
superblue10	-1.66	1.00	-33.14	1.00	2.12	1.00	4.65%	2.78%	-10.97%	5.12%	4.12%	-0.60%	9.34%	57.87
superblue16	-0.46	1.00	-0.69	1.00	0.96	1.00	2.41%	11.21%	-12.12%	3.10%	19.44%	-0.28%	10.56%	6.01
superblue18	-0.45	1.00	-1.04	1.00	0.63	1.00	-0.74%	-0.29%	-18.12%	1.10%	2.41%	-0.14%	15.23%	4.43
Average	-1.10	1.00	-6.14	1.00	1.16	1.00	0.83%	3.62%	-14.68%	1.70%	7.11%	-0.39%	12.41%	18.96

TABLE XI
COMPARISONS BEFORE AND AFTER CONSIDERING CONGESTION IN QPTST

	FLUTE Based Net Decomposition				QPTST				Congestion Aware QPTST			
Benchmarks	WL	r	Overflow	r	WL	r	Overflow	r	WL	r	Overflow	r
superblue1	0.78	1.00	3297.61	1.00	0.82	-3.16%	3603.07	-9.26%	0.82	-2.46%	3593.22	-8.96%
superblue3	1.07	1.00	2538.63	1.00	1.07	-0.83%	2538.63	0.00%	1.07	-0.83%	2538.63	0.00%
superblue4	0.77	1.00	749.92	1.00	0.78	-2.24%	778.55	-3.82%	0.77	-0.18%	751.39	-0.20%
superblue5	0.73	1.00	3060.91	1.00	0.73	-1.72%	3060.80	0.00%	0.73	-1.72%	3060.80	0.00%
superblue7	.37	1.00	3972.55	1.00	1.38	-0.93%	3984.21	-0.29%	1.37	-0.81%	3972.87	-0.01%
superblue10	1.71	1.00	7696.32	1.00	1.72	-1.69%	7768.53	-0.94%	1.71	-1.03%	7697.78	-0.02%
superblue16	0.98	1.00	684.93	1.00	0.98	-0.15%	688.89	-0.58%	0.98	0.17%	685.04	-0.02%
superblue18	.63	1.00	74.81	1.00	0.63	0.37%	75.08	-0.36%	0.63	0.55%	74.81	0.00%
Average	1.01	1.00	2759.46	1.00	1.01	-1.29%	2812.22	-1.91%	1.01	-0.79%	2796.82	-1.15%

TABLE XII
EXPERIMENTAL RESULTS OF MLA

Benchmarks	ML Accuracy			ML Over Base			ML Over QP		
	Accuracy	CPU(s)	QP-CPU(s)	r_wns	r_tns	r_wl	r_wns	r_tns	r_wl
superblue1	82.74%	2.98	14.87	1.71%	6.10%	-0.38%	0.42%	1.28%	-0.06%
superblue3	82.38%	1.13	6.16	3.67%	5.32%	-0.10%	0.37%	1.26%	-0.05%
superblue4	99.14%	6.39	49.13	1.55%	13.87%	-1.78%	-0.02%	0.11%	0.00%
superblue5	83.19%	2.97	11.87	0.25%	3.40%	-0.57%	0.05%	0.06%	-0.01%
superblue7	94.91%	1.59	45.63	0.00%	5.49%	-0.13%	0.00%	0.41%	0.00%
superblue10	87.99%	7.73	61.68	0.73%	3.71%	-0.76%	0.07%	0.55%	-0.08%
superblue16	95.53%	1.54	6.31	5.24%	29.54%	-0.32%	0.10%	0.14%	0.00%
superblue18	97.13%	1.21	18.65	0.09%	3.89%	-0.18%	-0.01%	0.01%	0.00%

design using different routing tree algorithms, such as FLUTE. Features are extracted from the routed design. Each data sample is based on each net information. We then performed our MIQP solver to get the reconnection result of each net, which is the label information of the data sample. Applying reconnection, we evaluated the design.

The training process is shown in Fig. 10(b). We originally obtained detail placements from [22] as our dataset. Eight benchmarks listed in Table VIII are utilized in the experiment. Each benchmark has eight detail placement results, which are two detail placement results given by the contest organizers and six timing driven detail placement results with long and short constraints generated by top three contestants in [22]. There are $\sum_{b=0}^{B}\sum_{p=0}^{P}$ #net$_{bp}$ data samples in total, where B is number of benchmarks and P is number of placement of each benchmark. #net$_{bp}$ is number of multiple pin nets with negative slacks of benchmark b on placement p. $B = 8$ and $P = 8$. As mentioned in Section IV-B, top 15 features are selected in our experiments. Training set consists of data originally from four placement results given by contest organizers (superblue4, superblue16, superblue18, and superblue7). Those data with

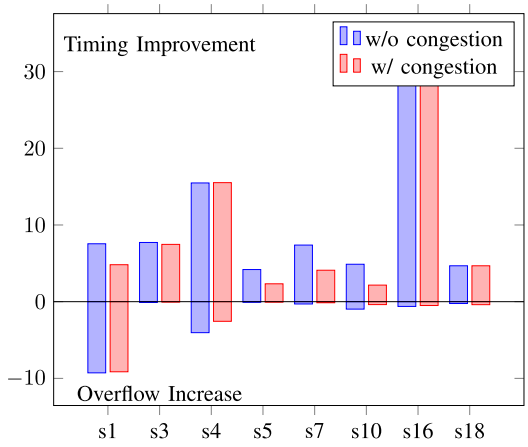

Fig. 11. Performance analysis on multiple sink optimization of timing and routing congestion.

scaled features are fed into cross-validation and our ML model is trained by RF method. The test data is from the rest of the dataset. We evaluated the performance of trained model by ML accuracy and timing improvement.

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

TABLE XIII
TIMING RESULTS FOR MULTIPLE SINKS CANDIDATES QP

Benchmarks	Baseline						Multiple Selection Optimization				Multiple Selection Congestion Optimization			
	WNS	r_wns	TNS	r_tns	stWL	r_stwl	r_wns	r_tns	r_stwl	CPU(s)	r_wns	r_tns	r_stwl	CPU(s)
superblue1	-0.50	1.00	-0.46	1.00	0.96	1.00	1.57%	7.55%	-0.38%	24.98	1.76%	4.82%	-0.32%	32.46
superblue3	-1.01	1.00	-1.50	1.00	1.14	1.00	5.98%	7.72%	-0.11%	16.48	5.83%	7.47%	-0.09%	25.32
superblue4	-0.62	1.00	-3.47	1.00	0.71	1.00	1.60%	15.48%	-1.85%	29.81	1.60%	15.52%	-1.75%	37.57
superblue5	-2.57	1.00	-6.95	1.00	1.08	1.00	0.34%	4.19%	-0.62%	16.29	0.14%	2.33%	-0.29%	25.98
superblue7	-1.51	1.00	-1.84	1.00	1.40	1.00	0.00%	7.38%	-0.14%	26.97	0.00%	4.10%	-0.09%	34.88
superblue10	-1.65	1.00	-33.10	1.00	2.05	1.00	4.45%	4.88%	-0.80%	48.54	0.05%	2.16%	-0.49%	56.86
superblue16	-0.46	1.00	-0.76	1.00	0.93	1.00	3.97%	33.30%	-0.40%	14.65	3.97%	29.81%	-0.38%	22.26
superblue18	-0.46	1.00	-1.03	1.00	0.58	1.00	3.29%	4.68%	-0.20%	11.66	3.29%	4.68%	-0.20%	19.42
Average	-1.10	1.00	-6.14	1.00	1.11	1.00	2.65%	10.65%	-0.56%	23.67	2.08%	8.86%	-0.45%	31.84

TABLE XIV
COMPARISONS BEFORE AND AFTER CONSIDERING CONGESTION IN MULTIPLE SELECTION OPTIMIZATION

	FLUTE Based Net Decomposition				Multiple Selection Optimization				Multiple Selection Congestion Optimization			
Benchmarks	WL	r	Overflow	r	WL	r	Overflow	r	WL	r	Overflow	r
superblue1	0.78	1.00	3297.61	1.00	0.82	-3.17%	3603.53	-9.28%	0.82	-3.01%	3598.93	-9.14%
superblue3	1.07	1.00	2538.63	1.00	1.07	-0.94%	2540.85	-0.09%	1.07	-0.93%	2539.45	-0.03%
superblue4	0.77	1.00	749.92	1.00	0.79	-2.41%	780.21	-4.04%	0.78	-2.05%	769.14	-2.56%
superblue5	0.73	1.00	3060.91	1.00	0.73	-1.75%	3062.74	-0.06%	0.73	-1.96%	3062.09	-0.04%
superblue7	1.37	1.00	3972.55	1.00	1.38	-0.95%	3984.37	-0.30%	1.37	-0.90%	3977.98	-0.14%
superblue10	1.71	1.00	7696.32	1.00	1.72	-1.69%	7770.84	-0.97%	1.72	-1.41%	7725.19	-0.38%
superblue16	0.98	1.00	684.93	1.00	0.98	-0.17%	689.17	-0.62%	0.98	-0.17%	688.36	-0.50%
superblue18	0.63	1.00	74.81	1.00	0.63	0.34%	74.98	-0.24%	0.63	0.36%	75.10	-0.39%
Average	1.01	1.00	2759.46	1.00	1.01	-1.34%	2813.34	-1.95%	1.01	-1.26%	2804.53	-1.65%

TABLE XV
EXPERIMENTAL RESULTS OF MULTILABEL CLASSIFIER

Benchmarks	ML Accuracy			ML Timing Over Base			ML Timing Over MQP		
	Accuracy	CPU(s)	MQP-CPU(s)	r_wns	r_tns	r_wl	r_wns	r_tns	r_wl
superblue1	86.74%	5.87	22.53	1.71%	6.68%	-0.36%	0.14%	0.28%	-0.01%
superblue3	85.81%	2.27	15.8	3.61%	5.18%	-0.09%	-0.10%	-0.26%	0.02%
superblue4	98.89%	13.08	25.81	1.55%	14.31%	-1.92%	0.24%	1.10%	-0.29%
superblue5	84.54%	17.56	15.90	0.20%	2.78%	-0.50%	-0.02%	-0.63%	0.12%
superblue7	96.41%	2.74	47.7	0.00%	6.38%	-0.13%	0.00%	0.23%	0.00%
superblue10	90.27%	15.19	51.38	0.64%	3.12%	-0.71%	-3.26%	-1.03%	-0.02%
superblue16	95.92%	3.36	13.33	5.57%	31.18%	-0.33%	0.06%	0.05%	0.00%
superblue18	97.20%	2.17	10.63	0.20%	4.16%	-0.19%	-0.08%	-0.04%	0.01%

The average evaluation of each benchmark is shown in Table XII. We can see from the table that runtime is reduced a lot. Moreover, we achieve a relative high accuracy classification rate. The final timing and wirelength improvement are also listed in Table XII. Classification over base shows timing results compared with baseline and classification over QP shows the comparison with the results of QPTST. The timing and wirelength quality are very compatible compared with our QPTST results. By such runtime reduction, we can use our ML classification approach to replace our MIQP solver to decide reconnection of each net.

B. Experiments on MSRSP Problem

1) Multiple Sinks Optimization QP Results:

a) Timing results: The results of MSRSP problem solved by Section IV is shown in Table XIII. The multiple selection optimization is the algorithm described in Section IV. The meaning of *WNS*, *r_wns*, *TNS*, $r_t ns$, *stWL*, and $r_s twl$ are adopted from Section VI-A1. Overall, our multiple selection optimization improved 10.65% TNS and 2.65% WNS on average. However, it takes 23.67 s. Compared with average improvement of reconnecting critical sinks, it improved the quality from 10.12% TNS and 2.05% WNS to 10.65% TNS and 2.65% WNS. With more sinks to be considered, the timing quality can be improved. Since more sinks are reconnected, wirelength increased 0.56% and it is 0.01% worsen than QPTST's.

The multiple selection congestion optimization is the approach that consider timing and congestion together by reconnecting multiple sinks. It improved 8.86% TNS and 2.08% WNS on average and it takes 31.84 s. The congestion aware optimization can improve WL but reduce the improvement on TNS and WNS.

b) Congestion results: The congestion analysis of multiple selection optimization and multiple selection congestion optimization are performed as same as described in Section VI-A1 congestion paragraph. As shown in Table XIV, the overflow of multiple selection optimization and multiple selection congestion optimization increased 1.95% and 1.65%, respectively.

c) Analysis: Fig. 11 shows the analysis of performance of multiple sink optimization on timing and congestion. The

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

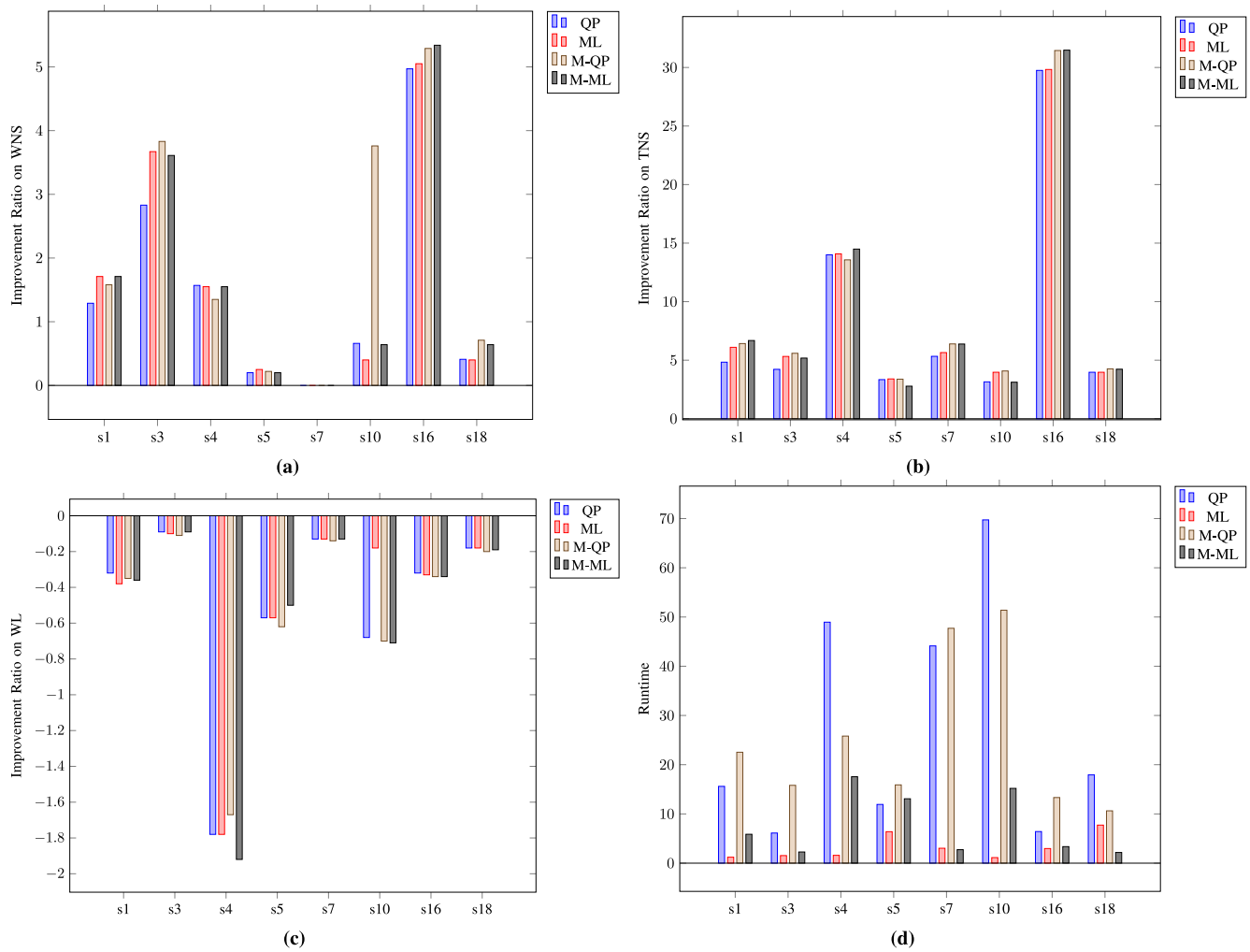

Fig. 12. Performance comparison on different methods. Improvement ratio on (a) WNS, (b) TNS, and (c) WL. (d) Runtime.

bar above 0 of the chart is timing improvement and the bar below 0 is plotted as increase of overflow. From the chart, we can conclude that overflow can be reduced a lot if the congestion is also considered.

2) Multilabel Classifier Results: The results of multilabel classifier is listed in Table XV. The training process is similar with the process described in Section VI-A2. The column ML Accuracy shows prediction accuracy and runtime comparison among benchmarks. The accuracy differs from benchmarks and the runtime is shorten compared with runtime of MQP. Our multilabel classifier takes more time compared with the previous binary classifier. The columns of ML over Base is the results compared with baseline. The columns of ML over MQP is the results compared with the multiple selection optimization in Table XIII. The quality is comparable with MQP's but runtime is much faster.

C. Comparisons Among Different Methods

We list four methods results: 1) QP (Section IV-A); 2) ML (Section IV-B); 3) M-QP (Section V-A); and 4) M-ML (Section V-B). Each design has eight different placement (only 7 for s7). The value of y-axis in each subfigure of Fig. 12 is the percentage ratio compared with baseline. The comparison is among WNS, TNS, WL, and runtime.

Fig. 12(a) is the WNS comparisons. M-QP performs better than QP on seven designs. ML approach has five designs no worse than QP and the rest two designs results are very similar. M-ML has five designs better than QP and four designs no worse than M-QP. On average, the improvement of WNS is 1.49%, 1.63%, 2.09%, and 1.71% of QP, ML, M-QP, and M-ML, respectively. M-QP shows the best performance based on WNS.

Fig. 12(b) is the TNS comparisons. The results of M-QP is better than QP's on seven designs and it is only worse 0.03% on design s4. ML outperformed QP among all designs. M-ML has similar results with M-QP excepts s5 and s10. On average, the improvement of TNS is 8.57%, 9.04%, 9.39%, and 9.29% of QP, ML, M-QP, and M-ML, respectively. M-QP obtains the best performance based on TNS.

Fig. 12(c) is the WL comparisons. On average, the improvement of WL is −0.51%, −0.45%, −0.52%, and −0.53% of QP, ML, M-QP, and M-ML, respectively.

Fig. 12(d) is the runtime comparisons. ML is better than QP and M-ML is better than M-QP among all designs. On

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.

average, the improvement of runtime is 27.6%, 3.2%, 25.39%, and 7.78% of QP, ML, M-QP, and M-ML, respectively.

Overall, M-QP has best timing quality but runtime is not that good. M-ML has a comparable results but much faster runtime.

VII. Conclusion

Timing is a critical issue for the design optimization and it is hard to improve timing without increasing routing congestion. In this paper, we formulate the tree surgery problem as a QP, which optimizes gate delay and net delay with adjacent nets considered. In order to enhance routability, congestion is also optimized in our algorithm. To speed up the process, an ML-based algorithm is proposed and features related to timing optimization are extracted from the design. By considering reconnecting more sinks, we formulate a QP to select reconnection among all sinks of each critical net and a multilabel classifier is proposed to accelerate optimization. In the experimental results, our algorithms can achieve high quality of timing improvement.

References

[1] J. Hu *et al.*, "Interconnect optimization considering multiple critical paths," in *Proc. ACM Int. Symp. Phys. Design*, 2018, pp. 132–138.

[2] V. F. Pavlidis, I. Savidis, and E. G. Friedman, *Three-Dimensional Integrated Circuit Design*. Cambridge, MA, USA: Morgan Kaufmann, 2017.

[3] A. B. Kahng, "New game, new goal posts: A recent history of timing closure," in *Proc. 52nd Annu. Design Autom. Conf.*, 2015, p. 4.

[4] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 709–722, May 2013.

[5] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-Route 2.0: A robust global router for modern designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1931–1944, Dec. 2010.

[6] M. Pan, Y. Xu, Y. Zhang, and C. Chu, "FastRoute: An efficient and high-quality global router," *VLSI Design*, vol. 2012, p. 14, Apr. 2012.

[7] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 2017–2026, Nov. 2008.

[8] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 70–83, Jan. 2008.

[9] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2130–2143, Dec. 2007.

[10] J. Hu and S. S. Sapatnekar, "A timing-constrained algorithm for simultaneous global routing of multiple nets," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2000, pp. 99–103.

[11] T. Jing *et al.*, "UTACO: A unified timing and congestion optimizing algorithm for standard cell global routing," in *Proc. ACM Asia South Pac. Design Autom. Conf.*, 2003, pp. 834–839.

[12] M. Cho, D. Z. Pan, H. Xiang, and R. Puri, "Wire density driven global routing for CMP variation and timing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2006, pp. 487–492.

[13] A. Youssef *et al.*, "A power-efficient multipin ILP-based routing technique," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 225–235, Jan. 2010.

[14] C. J. Alpert, A. B. Kahng, C. Sze, and Q. Wang, "Timing-driven Steiner trees are (practically) free," in *Proc. 43rd Annu. Design Autom. Conf.*, 2006, pp. 389–392.

[15] G. Chen, P. Tu, and E. F. Y. Young, "SALT: Provably good routing topology by a novel Steiner shallow-light tree algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2017, pp. 569–576.

[16] A. B. Kahng, "Machine learning applications in physical design: Recent results and directions," in *Proc. ACM Int. Symp. Phys. Design*, 2018, pp. 68–73.

[17] L.-C. Wang, "Machine learning for feature-based analytics," in *Proc. ACM Int. Symp. Phys. Design*, 2018, pp. 74–81.

[18] W.-C. Chang, I. H.-R. Jiang, Y.-T. Yu, and W.-F. Liu, "iClaire: A fast and general layout pattern classification algorithm," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[19] W.-T. J. Chan *et al.*, "Learning-based prediction of embedded memory timing failures during initial floorplan design," in *Proc. IEEE 21st Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2016, pp. 178–185.

[20] J. Hu, G. Schaeffer, and V. Garg, "TAU 2015 contest on incremental timing analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2015, pp. 882–889.

[21] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[22] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2015, pp. 921–926.

[23] T.-W. Huang and M. D. F. Wong, "OpenTimer: A high-performance timing analysis tool," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2015, pp. 895–902.

[24] IBM. (2017). *CPLEX*. [Online]. Available: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[25] W.-H. Liu, C.-K. Koh, and Y.-L. Li, "Optimization of placement solutions for routability," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2013, pp. 1–9.

Peishan Tu received the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2018.

Her current research interests include placement and routing in physical design.

Dr. Tu was a recipient of the Best Paper Award (twice) in the 2017 International Conference on Computer Aided Design 2017 and the 2017 ACM/IEEE International Workshop on System-Level Interconnect Prediction.

Chak-Wa Pui received the B.Sc. degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. E. F. Y. Young.

His current research interests include physical design for both ASICs and field-programmable gate arrays, Boolean matching in logic synthesis, and machine learning in physical design.

Mr. Pui was a recipient of the two best paper award nominations in DAC and ISPD and four ISPD/ICCAD contest awards.

Evangeline F. Y. Young received the B.Sc. and M.Phil. degrees in computer science from the Chinese University of Hong Kong (CUHK), Hong Kong, and the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 1999.

She is currently a Professor with the Department of Computer Science and Engineering, CUHK, researching on floorplanning, placement, routing, design for manufacturability, and algorithmic designs. Her current research interests include algorithms and very large-scale integration CAD.

Authorized licensed use limited to: SOUTH CHINA UNIVERSITY OF TECHNOLOGY. Downloaded on November 19,2024 at 06:26:46 UTC from IEEE Xplore. Restrictions apply.