Synergistic Topology Generation and Route Synthesis for On-Chip Performance-Critical Signal Groups

Derong Liu, Bei Yu, *Member, IEEE*, Vinicius Livramento, Salim Chowdhury, *Member, IEEE*, Duo Ding, Huy Vo, Akshay Sharma, and David Z. Pan, *Fellow, IEEE*

***Abstract*—As very large scale integration technology scales to deep submicron, design for interconnections becomes increasingly challenging. The traditional bus routing follows a sequential bit-by-bit order, and few works explicitly target interbit regularity for signal groups via multilayer topology selection. To overcome these limitations, we present Streak, an efficient framework that combines topology generation and wire synthesis with a global view of optimization and constrained metal layer track resource allocation. In the framework, an identification stage decomposes binding groups into a set of representative objects; with the generated backbones, equivalent topologies are accompanied by the bits in every object; then a formulation guides the routing considering wire congestion and design regularity. Furthermore, a bottom-up clustering methodology based on layer prediction targets to enhance the routability; a post-refinement stage is developed to match the source-to-sink distance deviation among bits in one group. Experimental results using industrial benchmarks demonstrate the effectiveness of the proposed technique.**

***Index Terms*—Primal-dual, regularity, routing, signal groups.**

I. Introduction and Related Work

As very large scale integration technology scales to deep submicron and beyond, design for on-chip interconnections becomes increasingly challenging. In current industrial designs, data and control signals loading messages from various sources can be bound as signal groups, as shown in Fig. 1. Observe that there are three signal groups marked with different colors. The signal bits in one group may have different numbers of pins, resulting in different routing styles. In this example, one style is signified with a pair of solid lines and a dashed line in the middle. The solid lines represent two signal bits on the border, as pointed in Fig. 1, while the dashed lines represent multiple bits inside. For signal bits with different pin locations in one group, they have to be routed in a regular manner. That is to say, common topologies are preferred to be shared among all the bits for design regularity, which is an extension of classic bus routing [1]–[3]. Meanwhile, with more metal layers integrated, it faces more challenges to control the routing congestion among multiple layers. For the performance-critical signal bits, the routability and wire-length should also be optimized to avoid functional inaccuracy. Therefore, an advanced synergistic router should be able to not only control routability and wire-length but also guide each bit routing intelligently for design regularity.

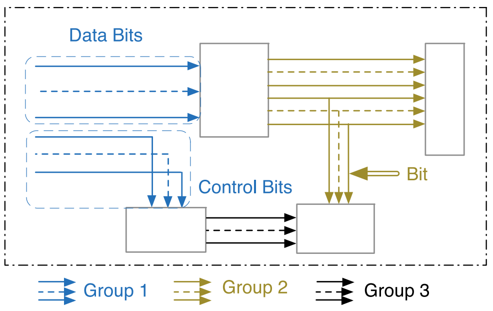

Fig. 1. Example of on-chip signal groups.

To realize these requirements, we propose an automatic topology generation and synthesis engine which is able to guide the routing of signal groups with a global view. Besides the improvement of routability and wire-lengths, we should also pay attention to the specific constraints brought by signal groups, where the bits in one group are encouraged to be routed in parallel tracks and share common topologies/layers for regularity. Meanwhile, instead of a bit-by-bit routing, signal bits can be clustered based on their possible route styles, as seen in Fig. 1, where two styles in *Group*1 are circled to be treated as an individual object. Then the problem size can be reduced by condensing several bits into an object, but with the resulting parallel routes, capacity constraints become more

Manuscript received September 23, 2017; revised February 5, 2018; accepted April 3, 2018. Date of publication May 8, 2018; date of current version May 20, 2019. This work was supported by the Research Grants Council of Hong Kong SAR under Project CUHK24209017. The preliminary version has been presented at the Design Automation Conference (DAC) in 2017. This paper was recommended by Associate Editor L. Behjat. *(Corresponding author: Derong Liu.)*

D. Liu and D. Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: derongliuliu@gmail.com).

B. Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

V. Livramento is with the Automation and Systems Engineering Department, Federal University of Santa Catarina, Florianópolis 88040-900, Brazil.

S. Chowdhury is an Independent Consultant in Austin, TX, USA.

D. Ding, H. Vo, and A. Sharma are with Microelectronics Group, Oracle Corpration, Austin, TX 78729 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2834424

stringent. During the whole procedure, all these constraints should be taken into accounts carefully.

There are a few previous works focusing on bus architecture synthesis for on-chip designs. Some bus-oriented work incorporates with floorplanning to satisfy the timing constraints [4], minimize total bus area [5], or improve dead space [6]. For timing analysis, an automated bus synthesis framework, FABSYN, incorporates the floorplanning with wire delay estimation engine to detect the potential violations [7]. Considering the impact of vias on lithography, a revisiting methodology is proposed to minimize the routing vias while controlling the loss of chip area and wire-length [8]. Furthermore, an OPC-friendly bus floorplanning algorithm allocates the bus positions with the consideration of the impact of off axis illumination on pitches [9]. Especially, multibend shapes are considered in [6] for providing more topology candidates through simulated annealing. Additionally, an effective algorithm minimizes the deviation for large-scale buses while improving the dead spaces and wire-length [10]. And a bus thermal analyzer models the potential hot spots on chips [11]. There are also some literature about escape routing on printed circuit board design: such as pin ordering and untangling [12], layer resource minimization [13], and an automatic planning flow in [14] including bus decomposition, escape routing, layer assignment, and global routing. Compared with the previous work, our synthesis tool provides a more extensive view to deal with bundled signal groups with more possibilities.

Very few of previous routing work targets at synergistic topology generation and routing synthesis of signal groups with multipin connections. For current industrial designs, regular topologies with parallel routes are highly preferred to reduce interbit variability spread on silicon. Therefore, an efficient topology generator should be able to facilitate signal bits directing to different cells with low twisting or distorted connections. Besides, compared to two-pin buses, signal groups contain the bits with varying numbers of pins according to their specified logic connections. This also increases the problem complexity by providing more routing possibilities and congestion challenges. Additionally, considering the requirement of source-to-sink distance control, appropriate twisting routes are required to complement the deviation among multiple bits in groups. Therefore, an intelligent framework is essential to guide the routing with the respect of regularity and wire-length efficiently.

In this paper, we propose an automatic topology generator and routing synthesis for on-chip signal groups. Our contributions are highlighted as follows.

1) An automatic framework directs topology and routing synthesis of bundled groups with multipin connections.
2) An identification stage partitions signal groups into a set of objects, where each bit has an equivalent topology.
3) A mathematical formulation improves routability and wire-length while handling the topology similarity.
4) A primal-dual flow benefits the runtime while keeping very comparable quality.
5) A bottom-up clustering strategy integrates with layer prediction to enhance the routability of signal groups.
6) A refinement stage allows appropriate twisting routes to reduce the source-to-sink distance deviation.

The remainder of this paper is organized as follows. Section II presents the overview of our framework and adopted models. Section III describes our synergistic topology generation procedure, presents a mathematical formulation to optimize wire-length and routability while controlling regularity, and a prime-dual flow benefits the runtime. Section IV provides a post optimization stage to further enhance the signal routability and match the source-to-sink distances among different bits. Section V reports the experimental results and followed by the conclusion in Section VI.

II. Preliminaries

In this section, we provide the overview of our proposed framework, and illustrate the adopted model and methodology, based on which a problem formulation is given.

A. Streak Flow

To provide an explicit view of Streak framework, the overall flow is illustrated in Fig. 2. Initially, the information of specified track allocation and pin locations from bits bundled in signal groups is provided. To make it explicit, the definition of signal groups is defined as follows, and signal groups are predefined and provided by users.

Definition 1 (Signal Group): The performance-critical bits whose pins are located in adjacent locations and required to share common topologies are defined as a signal group.

Considering that the bits in a group may require various routing types, as shown in Fig. 1, we identify the possible routing types of each bit based on its pin locations. Those bits are combined as one routing object and able to obtain equivalent topologies. Since our framework targets at multilayer structure, 3-D topology candidates are required for the objects on different layers. To achieve this, we construct a set of 2-D backbones for each object and derive equivalent topologies for each bit in an object. In our flow, a backbone contains a complete routing solution of all the bits in reference to this backbone. Then the acquired topology candidates are developed to different layers for a 2-D solution, all of which are considered as candidates for selection. After handling the equivalence of each object, we further quantify the dissimilarity among objects in a group through regularity ratio. Based on these operations, a primal-dual flow solves all the objects efficiently. During the primal-dual flow, we search for the most appropriate solution for each signal routing object in a progressive manner. The details of each step in the flow will be given in Section III. The following sections describe the routing model adopted through our framework to handle the topology generation and route synthesis for synergies.

B. Proposed Signal Model

Similar as global routing, signal route can also be modeled on a 3-D global grid model. In real industrial designs, 3-D routing is preferred to avoid the suboptimality of a post layer assignment step. Similarly, each layer is also divided into a set of rectangular routing cells in a 2-D manner, i.e., *G-Cell*,

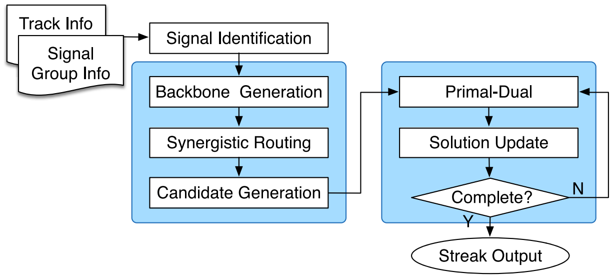

Fig. 2. Overall streak flow.

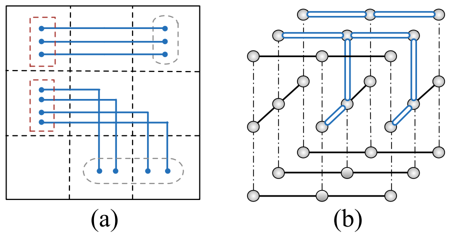

Fig. 3. Illustration of signal routing model. (a) Example of 2-D routing and (b) 3-D routing.

shown as a vertex in Fig. 3(b). Additionally, the edges connecting vertices in 2-D planes are for routing wires, whose capacity constraints have to be satisfied. This means that the number of passing bits cannot exceed the maximum capacity for each edge. Different from traditional routing, signal bits prefer to be routed in parallel tracks and share common topologies as much as possible for regularity. For a signal group, several bits may occupy the same edge simultaneously, which aggravates the routing congestion. Therefore, edge capacity constraint becomes more challenging through guiding the overall route of all signal bits.

Based on the 3-D grid model, efficient routes can be generated considering the specified requirements of signal groups. By modeling *Group*1 from Fig. 1 on a 2-D grid, as shown in Fig. 3(a), this group is to be divided into two routing styles based on their pins' locations as circled. Each style corresponds to an individual object consisting of several bits, and the topologies of these two objects are encouraged to be shared as much as possible. Therefore, it turns out that they are routed in horizontal tracks from the drivers, and their corresponding 3-D solutions are provided in Fig. 3(b), where the horizontal trunks are assigned on the same metal layer.

C. Proposed Bit Model

As shown in Fig. 1, one signal group contains a specified number of bits, which may have various numbers of pins located in different directions. For the bits belonging to one group, their routes should be coordinated and adhere to some constraints: topology variance should be well controlled by matching the connection with the mapped pins located at the similar direction in the bits; for a pair of mapped pins, their source-to-sink distances should be within certain bounds to reduce the deviations. As shown in Fig. 4(a), three bits are listed with the mapped pins as clustered together, where the

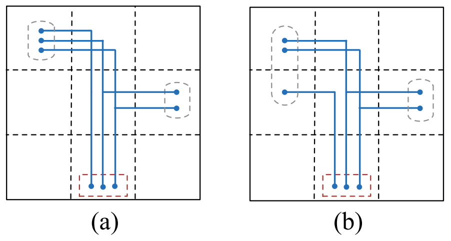

Fig. 4. Illustration of source-to-sink distance for signal bits. (a) Example of equivalent distance for all bits. (b) Example of interbit distance deviation.

red dashed squares signify the drivers of the bits. Observe that the distances between the driver and the mapped pin are the same in spite of existing different pins. Nevertheless, not all the bits in one group are able to achieve the equal distance ideally. An example is provided in Fig. 4(b), from which a much shorter distance exists for one sink in the leftmost bit compared to the other bits. This will induce the source-to-sink distance deviation for this pin, and further result in diverse arrival times for the modules. To avoid the possible malfunction, it is essential to control the distance deviation in an acceptable range. Therefore, a threshold is introduced so that the deviation should be under this constraint.

D. Proposed Similarity Vector Model

Based on the examples in Figs. 3 and 4, it is imperative to present a model which distinguishes the bits in a bundled group according to their different pin connections. That is to say, all the bits in a distinguished object are to acquire equivalent topologies. FLUTE [15] provides an elegant definition of equivalent topology through vertical sequences, where the same sequence is guaranteed to produce an equivalent topology. By extending this, we develop a similarity vector for each pin, SV(p_m), to capture its relative location in its bit. Furthermore, SV(p_m) is also utilized to find the corresponding pin in another bit from one signal group. Based on the corresponding pins from other bits, their routes can be coordinated in a synergistic manner through appropriate mapping and calibration.

Since the corresponding pins of different bits can be located in various *G-Cell*s, we prefer to use the relative direction rather than distance to describe each pin's location. As shown in Fig. 5(a), the SV for pin p_m in its net is decided through a quadrant-based model, which characterizes the connecting directions in comparison to p_m. It is seen that there are eight directions in total: each quadrant contributes a direction while both x- and y-axes contribute two directions. Then a similarity vector is presented as shown

$$\mathrm{SV}(p_m) = \{n_p(+x), n_p(I), n_p(+y), \ldots, n_p(IV)\} \qquad (1)$$

which records the number of other pins in this bit, i.e., n_p, from each direction by a counter-clockwise sequence. For the example shown in Fig. 5(a), assume that the driver is in the middle and each "X" represents a sink, then SV of this driver is {1, 1, 1, 1, 1, 1, 1, 1}. Taking the example in Fig. 3(a) as an instance, there are two routing styles which can be distinguished through the vector. For the top style, the SV of the driver is {1, 0, 0, 0, 0, 0, 0, 0}, while the sink has the SV as {0, 0, 0, 0, 1, 0, 0, 0}. Thus, in each routing style every pin has the same SV, and these pins belonging to various bits in a group can be mapped mutually. Based on the mapped pins, we are able to provide equivalent topologies for the bits in an object, while topologies among objects can also be coordinated to reduce the dissimilarities. Therefore, SV plays an important role in processing topology synergy of the bits in a group.

E. Problem Formulation

Based on the proposed flow and routing model discussed in the preceding section, we define the synergistic topology generation and route synthesis (Streak) problem as follows.

Problem 1 (Streak): Given signal bits in bundled groups and layer capacity information, Streak determines the routing topology and layer assignment for each signal bit so that the routability, wire-length, and topology regularity can be optimized while the edge capacity constraints are satisfied.

III. Algorithms

In this section, we present the technique details adopted through Streak flow. A preprocessing stage partitions each signal group into a set of routing objects; a set of backbone structures is constructed and equivalent topologies are developed; a mathematical formulation selects the appropriate topology and assigns penalties to control irregular topologies; and a primal-dual algorithm is presented finally for speed-up.

A. Identification of Signal Isomorphism

Besides covering general bus routing, our framework provides more feasibilities to handle groups of signal bits, which can be loaded from data or control information. A preprocessing stage provides a set of bits clustered in different groups, which can belong to multiple buses and prefer to share common topologies as much as possible. In comparison to general bus planning, these binding signal bits possess different numbers of pins which lead to a set of adjacent physical locations. Therefore, with the integration of signal groups, the algorithmic complexity increases with more possibilities.

To provide regular routes for bundled signals, we prefer to partition a provided signal group into a set of subgroups, and deal with each subgroup as an individual routing object. In each object, every bit is able to acquire an equivalent topology and all its pins have the same SVs as the pins in other bits. That is to say, each pin is able to find its corresponding reflection from any other bit in the same object. After the routing flow, each bit in an object obtains an equivalent topology while for different objects in a signal group, they are preferable to share common topologies as much as possible.

With this objective, the partition strategy is illustrated in Fig. 5(b). The input is one signal group represented by a white root node, and the output is a set of routing objects represented by the gray nodes, containing the bits with the same SVs for the pins. The right squares provide the pin distributions of one signal bit belonging to an object. When the pins have the same

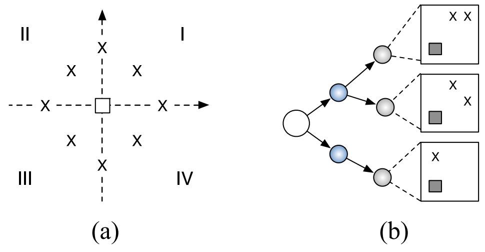

Fig. 5. Example of signal identification. (a) Quadrant-based similarity vector. (b) Hierarchical isomorphic identification.

SVs, their bits are able to reach one topology. The methodology is intuitive but naive by calculating the SV for each net pin, which would result in considerable calculation overhead. For those bits that are deemed to obtain different topologies based on their pin locations, we prefer to distinguish them as soon as possible. Therefore, we adopt a hierarchical strategy based on the premise that the driver pins of various bits in one group are mapped mutually. In this way, we calculate the driver's SV for each bit at first. Those bits with different values are separated as the middle blue nodes in Fig. 5(b). For the signal bits in the top blue node, their drivers have the same SV as {0, 2, 0, 0, 0, 0, 0, 0}, although they are not able to obtain one topology. It is easy to see that equivalent topologies are infeasible for the bits whose drivers have different vectors. With this stage, the complexity decreases without traversing all the pins due to the fact that the pins located in each direction to the driver is quite limited. Then, we only need to evaluate those pins located in one quadrant, as shown in Fig. 5(b). By taking the top blue node as an example, we just compare the SVs of the pins in the first quadrant to the driver. Finally, the bits with the same SV for all the pins are combined as an object, and common topologies are encouraged for the objects in one group. In industrial designs, the signal groups are user-defined by referencing the specifications from many aspects, such as signal shielding, cell connection, etc.

B. Topology Generation and Evaluation

Before solving the signal routing problem, an efficient topology generation procedure is essential to provide candidate solutions. In this section, we propose a synergistic topology generation strategy for multipin connections which require equivalent topologies in one object and sharing topologies among objects in one group. It consists of three steps: 1) backbone generation for each single object; 2) equivalent routes generated for the bits based on the backbone; and 3) regularity evaluation among backbone topologies of various objects.

1) Backbone Structure Construction: After the isomorphic identification, a set of routing objects can be acquired from a group, where all the bits can be routed with a topology, i.e., backbone structure. In essence, it is a topology prototype of all the bits in this object. The pins of one representative bit serve as the input information, and the output is a set of rectilinear connections (RCs) of the pins and bending points with the same *X*/*Y* coordinates. Its formal definition is given as follows. To select a bit in the object, we choose one bit in the center region of an object and take its pins as the input.

Since the identification stage distinguishes the bits sufficiently, a selected bit can be representative of all the bits in one object.

Definition 2 (Backbone Structure): With a representative bit's pin locations, backbone structure is defined as a routing topology which every bit in the same object is able to use.

For the topology generation, we extend the batched iterated 1-Steiner (BI1S) algorithm [16] based on an industrial flow. Since the topologies with many bends are not suitable for signal groups, the number of bending points is also an important index besides the wire-length. Considering that a backbone would affect all the bits in an object, it is essential to save wire-length while keeping as few bending points as possible. Therefore, a set of promising bending points should be selected for BI1S. It is known from Hanan grids that Steiner points should be located at the crossing points of input/output pins, which also conforms to bending points in our flow. Nevertheless, it is trivial to traverse all the internal edges connecting the pins and points, which may result in too many inferior candidates. Thus, we only extract the promising points and remove those resulting in long wire-lengths or complicated topologies. Then the selected points are saved into one queue with the priorities which indicate their potential wire-lengths and bending costs. To generate a set of topologies, we pick and insert the points from the queue to construct Steiner trees with the consideration of both wire-length and bending costs. Through the combination of pins and inserted points with a set of RCs, we are able to achieve a rectilinear Steiner tree. Then we select a noninserted point from the queue with the highest priority to construct another tree. For each tree, at least one different bending point is adopted for the topology candidate. After visiting all the promising points at least once, we obtain an appropriate set of backbones on 2-D plane for post-processing.

Since the objective of this procedure is to provide topology prototypes for the objects, here we do not consider the required demands of tracks and capacity constraints. For the demands through *G-Cell*s, the following equivalent topology generation phase will collect the topologies of the bits and calculate the total required tracks. With the exact topology of each bit, then the calculation will be accurate for reference. Besides, the possibly occuring conflicts from the objects will be taken into careful consideration in Section III-C. The reasons are as follows. First, the routing layer has not been decided in the current stage, so the conflicts cannot be obtained due to the various capacities in multilayer structure; second, as our optimal objective is to coordinate the routes from the objects while satisfying the capacity constraints, a comprehensive formulation with the global view of all these points is provided in (3) with more details.

2) Equivalent Topology Generation: Compared with classic escape routing, signal routing has more stringent constraints for the bits in a binding group: topology equivalence is required for those bits in an object; common topologies among objects should be shared as much as possible. Section III-A describes how to partition a signal group into a set of objects, and a set of backbones is constructed for each object in Section III-B1. This section focuses on equivalent topology generation for an object according to each backbone. To achieve this objective, we refer to the similarity vector presented in Section II-B. Through making sure the corresponding pin in backbone for each bit, we are able to generate a topology same as backbone. The equivalent topology generation details are given in Algorithm 1.

Algorithm 1 Equivalent Topology Generation

```
Require: Initial backbone $t_o$;
 1: Build LUT with $SV(p), p \in t_o$;
 2: Record $bt(t_o)$ with its connecting pins;
 3: for each bit $b \in$ do
 4:     Map $p \in b$ to $p \in t_o$;
 5:     while $\exists$ non-visited $bt(t_o)$ do
 6:         Select a non-visited $bt(t_o) \in t_o$;
 7:         Find $p_x(bt, t_o), p_y(bt, t_o)$;
 8:         Acquire $p_x(bt, b), p_y(bt, b)$ from map;
 9:         Determine $bt(b)$ based on $p_x(bt, b), p_y(bt, b)$;
10:         $bt(b)$ connect $p_x(bt, b)$, $bt(b)$ connect $p_y(bt, b)$;
11:     end while
12: end for
```

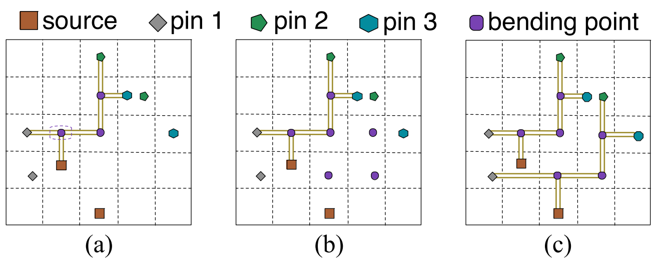

Fig. 6. Equivalent topology generation example: (a) Pin mapping through similarity vector. (b) Bending points aligning. (c) Topology generation by connecting mapping pins and points.

After the identification stage, all the bits in one object have the same SV for each pin. Thus, it is explicit to find the corresponding pin in backbone for each bit, and build a map to show this relationship. Based on a set of backbones generated beforehand, each equivalent topology is accompanied for each bit with the same connection of corresponding pins. To achieve this objective, we first construct a look-up table (LUT) which captures the relative location of each pin in the corresponding bit. Take the example in Fig. 6(a), pin_1 in the backbone will be mapped to the SV as {0, 2, 0, 0, 0, 0, 0, 1}. During LUT construction, all the pins are traversed to record their SVs for further matching (line 1). Meanwhile, based on Hanan grids, bending points in the backbone are located with the same X/Y coordinates as its pins. Hence, each bending point can be recognized easily through its neighboring connected pins (line 2). For instance, the circled bending point in Fig. 6(a) can be located by pin_1 and the source pin. Then we traverse each bit for topology generation in reference to each given backbone.

For each bit, through the LUT, each pin can be mapped to its reflection in the backbone according to its SV (line 4). For pin_1 of the nonrouted bit in Fig. 6(a), due to the equivalence of its SV to pin_1 in the backbone, these two pins are mapped to each other. In the shown example, each pair of mapped pins are identified with the same shape in one color. After setting this matching relationship of pins, we start to build the topology by calibrating the bending points in each signal bit. During each iteration, a nonvisited bending point is selected arbitrarily from the backbone, which has both horizontal and vertical connections to the pins (line 6). These connected pins, $p_x(bt, t_o), p_y(bt, t_o)$, are taken and utilized as reference to align this bending point $bt(t_o)$ (line 7). Based on these pins, we obtain the corresponding matched pins in this signal bit, $p_x(bt, b), p_y(bt, b)$, with the assistance of the constructed map (line 8). Then, the bending point in the bit can be located with the same X coordinate as the vertical pin $p_x(bt, b)$, and Y coordinate as the horizontal pin $p_y(bt, b)$ (line 9). For the circled bending point in Fig. 6(a), the corresponding point will be aligned based on the X coordinate of the source pin, and Y coordinate of pin_1 in Fig. 6(b), and so on for the other bending points. Through connecting the bending points with these neighboring pins with the same X/Y coordinates, an equivalent topology is able to be obtained (line 10). It is seen that with the LUT, the runtime of this algorithm is within $\mathcal{O}(|P_b||N_b| \log |P_b|)$, where $|N_b|$ represents the number of bits in an object, and $|P_b|$ represents the number of pins in a bit.

An explicit example is illustrated with three phases in Fig. 6. Fig. 6(a) provides the backbone and the mapped pins through LUT, while each corresponding pin is identified with the same shape in one color. With these mapped pins, the internal bending points are determined and aligned in Fig. 6(b). Finally, an equivalent topology is given through connecting the pins and points for the specified bit in Fig. 6(c).

Additionally, considering the existing multilayer structure for current industrial designs, we develop a series of topologies with different layers based on each 2-D routing tree. For regularity, the horizontal and vertical trunks should be assigned on the same uni-directional layer; in the meantime, these trunks are preferred to be assigned on the neighboring layers in order to save the unnecessary via overheads.

3) Regularity Evaluation: Through the previous stages, equivalent topologies are guaranteed in each routing object. Nevertheless, since the signal groups are user-defined with pin locations in different directions, it is infeasible to enforce topology equivalence for all the objects in a given group. Therefore, we prefer to use a novel metric to quantify their topology differences. Considering that a backbone is able to represent the key structure for each object, it is explicit to take backbones into accounts for irregularity evaluation.

As described in Section III-A, for two bits with the same number of pins, these pins can be mapped reciprocally according to their SVs. However, for those with different numbers of pins, SV is able to target the most probable pin of another bit. To reach this objective, we adjust SV by incrementing the weight of driver pin which should be mapped to the drivers of other bits as expected. The weight is set to a value higher than the overall number of pins. Through this adjustment, the relative position of each pin to its driver is emphasized. Also, we calculate the SV for each bending point so that they can also be mapped to the pins or bending points of other topologies. By matching the pins/points with the closest SV in two topologies, t_1 and t_2, the regularity rate is computed as in (2). It is equal to the number of mapped RCs formed by two mapped pins/points, NM_{RC}, divided by the minimum number of RCs in t_1 and t_2. As shown in Fig. 3(a), although the bottom object has one more bending point than the other, the topologies of these two objects are still regarded as similar topologies since this point can be mapped to the sink of the other object. Therefore, for this example, the ratio is set to 100% because both the number of mapped RCs and minimum number of RCs are equal to 1. In our algorithm flow, it is preferable to keep this ratio as high as possible to eliminate the dissimilar topologies. Since the denominator is more than or equal to the numerator in (2), the highest value of the ratio is 1, which indicates that topologies, t_1, t_2, share one topology

$$\text{Ratio}(t_1, t_2) = \frac{\text{NM}_{\text{RC}}(t_1, t_2)}{\min\{N_{\text{RC}}(t_1), N_{\text{RC}}(t_2)\}}. \tag{2}$$

C. Mathematical Formulation

The mathematical formulation of Streak is provided in (3). In the objective function, the first term is to calculate the total costs of all the objects, where $c(i,j)$ gives the cost of candidate x_{ij} of object i based on its wire-length and assigned layers. Since layering is taken into accounts, a post layer assignment stage can be saved to avoid potential suboptimality. The second item is to enforce the routing of objects and M is a large penalty for those nonrouted objects, whose s_i will be set to 1. Here, S_c refers to the set of solution candidates, while S_o refers to the set of routing objects. To minimize the topology variance, we add the third item in (3a). It helps to quantify the topology irregularity of any two objects in group g, which is the reciprocal of the regularity ratio. For two topologies without sharing any RCs, a large penalty will be set but which should be smaller than M to ensure the first priority of routability. Meanwhile, for x_{ij} and x_{pq}, if the RCs are shared but the routed layers are not adjacent, a penalty proportional to the layer difference will also be assigned

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in S_c} c(i,j)\cdot x_{ij} + \sum_{i\in S_o} M\cdot s_i \\
& + \sum_{(i,p)\in g}\sum_{(i,j)\in S_c}\sum_{(p,q)\in S_c} c(i,j,p,q)\cdot x_{ij}\cdot x_{pq} && \text{(3a)}\\
\text{s.t.} \quad & \sum_{(i,j)\in S_c} x_{ij} + s_i = 1, \quad \forall i\in S_o && \text{(3b)}\\
& \sum_{(i,j)\in e_l} u_{e_l}(i,j)\cdot x_{ij} \le \text{cap}_{e_l}, \quad \forall e\in E, \forall l\in L && \text{(3c)}\\
& s_i \ge 0, x_{ij} \text{ is binary}, \quad \forall i\in S_o, \forall j. && \text{(3d)}
\end{aligned}$$

Meanwhile, constraint (3b) is to ensure that at most one topology is selected for each routing object; while constraint (3c) places the capacity limitation of each edge on different layers, i.e., cap_{e_l}. Due to the sharing topologies, we deal with a stringent edge capacity constraint for one edge can be utilized multiple times by several bits in an object concurrently, as shown in Fig. 3(a). Thus, we prefer to add one constant to provide the edge usage by the current topology, i.e., $u_{e_l}(i,j)$. Finally, with the constraints of both

x_{ij} and s_i as binary variables, it is seen that this quadratic programming problem can be solved through integer linear programming (ILP).

D. Primal-Dual Algorithm

Although an ILP solver can be utilized to solve (3), in real design it is not preferable due to its prohibitive runtime when lots of variables exist. We thus design a primal-dual algorithm to provide an efficient solution. With the generated topologies, a fast flow is essential to make a sensible selection of candidates while satisfying the given requirements. A primal-dual algorithm is generally utilized for vertex covering problem, such as layer decomposition work in [17], which can also be applied in routing flow by incrementing the dual variables accordingly. This section provides the details to solve the routing flow through a primal-dual algorithm.

At first, we prefer to linearize the quadratic terms in (3) for a primal formulation. Some previous works predefine one of these two variables as a known value through an iteration-based framework [18], [19], while [20] takes the quadratic terms through extensive semidefinite programming for more accuracy. Considering the properties of primal-dual, we search for the allowable minimum value of each term based on the states of x_{ij} and x_{pq}. Therefore, (4) provides a relatively accurate approximation

$$\sum_{(i,p)\in g}\sum_{(i,j)\in S_c}\sum_{(p,q)\in S_c} c(i,j,p,q)\cdot x_{ij}\cdot x_{pq} \approx c'(i,j)\cdot x_{ij} \tag{4}$$

where

$$c'(i,j) = \begin{cases} c(i,j,p,q), & \exists x_{pq}=1 \\ \min\{c(i,j,p,q)\}, & \forall x_{ij}\cdot x_{pq}\neq 0. \end{cases} \tag{5}$$

Since the primal-dual algorithm is a progressive flow through which x_{ij}s increase in a step-by-step manner, for a determined solution x_{pq} as 1, its combining cost with x_{ij} will be integrated with $c(i,j)$ as an additional cost. Nevertheless, if no solution has been decided for p, the minimum combining cost with any feasible x_{pq} will be considered as the cost. Here, the feasibility refers to whether the combination of x_{ij} and x_{pq} can satisfy the current edge capacities. If not, this combination will be removed from the solution set. With this linear approximation, a dual problem ($\mathcal{DP}$) can be acquired as in (6)

$$\mathcal{DP}: \quad \max \sum_{(i,j)\in S_c} \alpha_{ij} + \sum_{e_l\in E,L} \mathrm{cap}_{e_l}\cdot \beta_{e_l} \tag{6a}$$

$$\text{s.t.} \quad \alpha_{ij} + \sum_{i,j:e_l\in x_{ij}} u_{e_l}(i,j)\cdot\beta_{e_l} \le c(i,j)+c'(i,j), \forall i,j \tag{6b}$$

$$\alpha_{ij}\le M, \quad \forall i\in S_o, \forall j \tag{6c}$$

$$\beta_{e_l}\le 0, \quad \forall e\in E, \forall l\in L. \tag{6d}$$

Formula (6) provides the dual form of (3) with the linearizing item, which incorporates two types of dual variables: α_{ij} for constraint (3b) and β_{e_l} for constraint (3c). Based on the strong duality, the optimal solution for (3) can be determined by satisfying the constraints in (6). Therefore, we prefer to start with a primal infeasible but dual feasible solution set, and increment the primal solutions accordingly until a feasible solution is obtained.

Algorithm 2 Primal-Dual Algorithm

```
Require: A set of routing objects with its candidate set.
1:  Initiate primal solutions $x_{ij}, s_i$ to 0;
2:  Initiate dual solutions $\alpha_{ij}, \beta_{e_l}$ to 0;
3:  Calculate $c(i,j), c'(i,j)$ for each $x_{ij}$;
4:  while $\exists \sum x_{ij} + s_i = 0$ do
5:      Search for a set of infeasible objects $i$;
6:      Select $x_{ij}$ with the minimum $c'(i,j)+c(i,j)$;
7:      $x_{ij} \leftarrow 1$, $s_i \leftarrow 0$;
8:      Update $cap_{e_l}$ where $e_l \in x_{ij}$;
9:      Remove infeasible primal solutions;
10:     if no feasible $x_{pq}$ for $p$ then
11:         $s_p \leftarrow 1$;
12:     end if
13:     Update $c'(p,q)$ for residual feasible solutions;
14: end while
```

The outline of primal-dual algorithm is described in Algorithm 2, where the input is a set of routing objects with their candidates. The initial primal solutions are set to 0 while keeping the dual variables also to 0 for their feasibilities (lines 1 and 2). Then the minimum required cost is calculated for each candidate to reach the upper bound of constraint (6b) (line 3). For each iteration, we check whether there still exist infeasible x_{ij}s and s_i, and the infeasible one with the minimum cost will be selected to increase its primal solution value (lines 5 and 6). Notably, here x_{ij} should be able to satisfy the current capacity constraints without any violations. Then x_{ij} increases to 1 while s_i is kept to 0 due to the primal constraint. With the update of x_{ij}, we recalculate the available routing tracks of each edge passed by x_{ij} (line 8). Meanwhile, due to the decreasing usable tracks, some x_{pq}s become infeasible. Thus, they can be removed securely without affecting the solution quality. For a specified object p, if all its x_{pq}s have been abandoned, s_p can be set to 1 (lines 10–12). Considering the existence of $x_{ij}\cdot x_{pq}$ in (3), $c'(p,q)$ should be updated if it relates with x_{ij} (line 13). Since the physical characteristic of this quadratic term is the combination of x_{ij} and x_{pq}, $c'(p,q)$ should be recalculated when the combination is not available due to the reduced capacities. Through the search procedure, the sum of these dual variables keeps enhancing until an upper bound is reached by finishing all the solutions. During the whole process, edge capacity constraints are always held for infeasible solutions are already bounded beforehand.

IV. Post Optimization

Section III provides a complete flow to coordinate topology and layering assignment for signal groups appropriately. Through the proposed similarity vector model in Section II-D, the bits in a single object can be determined to reach an equivalent topology. By generating a set of candidates, we obtain the topology and layer assigning result for each object. Nevertheless, after the primal-dual flow, some objects may not be routed due to the high number of bits in an object. That is to say, even for an object, where its bits can reach one topology, we may not be able to provide adequate routing resources

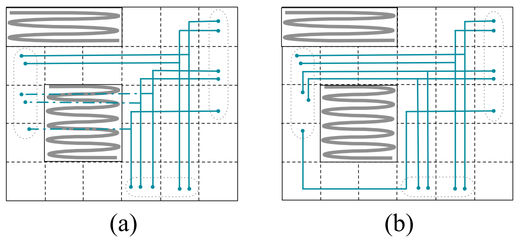

Fig. 7. Example of blocked routing instance. (a) Routing of some bits blocked by obstacles. (b) Multiple topology selection for each cluster.

because of its required high widths. Therefore, it is imperative to provide further division for the nonrouted objects so that more flexibilities are allowed, and the topology variance should also be controlled among the bits for regularity.

To make it explicit, a blocked instance is given in Fig. 7(a), where the dashed circles signify the mapped pins for each bit. It is seen that all the bits can reach the same topology if there is no obstacle. To deal with this issue, we allow further division for those bits so that we acquire more opportunities to enhance the final routability. Fig. 7(b) provides a possible solution, where three routing patterns are shown for all the bits. In this way, the blockage is bypassed without paying a high penalty of wire-length and topology variance. Generally, with a slight degradation of regularity, multiple clusters can benefit the routability of those blocked objects. To ensure the effectiveness, it is essential to balance the tradeoff between routability and design regularity.

Fig. 8 lists the outline of our post-optimization, which targets at improving the routability and refining topologies of signal groups. Instead of using common rip-up and reroute technique, during signal routing we prefer to maintain the current topology and layer assignment result. The reasons are twofold: First, since one signal group contains the bits with regular routes and concurrent bending points, this increases the complexity of splitting those bits simultaneously. It is also hard to find another feasible routing space for rerouting the bits based on the limited track resources, and a domino effect can be caused by ripping up others continuously, resulting in unexpected distortion. Second, the proposed primal-dual flow considers the optimization of wire-length and topology regularity concurrently. Based on its closure to a global optimal result, it is intuitive to provide an incremental approach to take advantage of the residual resources without causing further disturbance. Therefore, our post optimization flow is provided in Fig. 8. For the signal groups to be routed, the preferable layers are predicted based on the congestion; and a bottom-up clustering methodology combines the bits while keeping legality for capacity constraints. This procedure continues until all these groups have been traversed. After this stage, we check if there exist source-to-sink distance deviation violations. If so, we will introduce appropriate twisting detours to refine the topologies. The details of each step are given in the following sections.

A. Possible Layer Prediction

Different from traditional layer assignment which behaves after 2-D routing, we take layering into consideration before

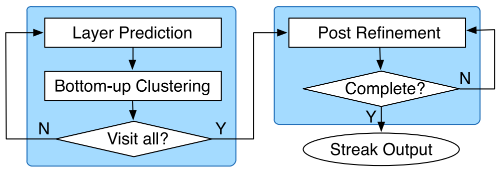

Fig. 8. Post-optimization flow.

exact routing. Based on the limited resources, we can narrow down the solution space by predicting the layers efficiently. Due to unidirectional routing, it is required to select two layers favoring horizontal and vertical directions. Since the eventual routes have not been decided, a predictive methodology is utilized to estimate track usages on each layer. With this approximation, the appropriate layers are acquired with the least conflict values regarding the already routed bits.

To provide an estimation of track utilization, we take all the available topologies of the bits into accounts. For a 2-D edge e, its possible usage by a group g is calculated as follows:

$$u(e, g) = \sum_{b \in g} \sum_{t_j \in S_c(b)} \frac{1}{|S_c(b)|} \cdot u(e, t_j) \tag{7}$$

where $S_c(b)$ denotes the set of solution candidates for bit b, and $u(e, t_j)$ denotes if this edge is used by the jth topology candidate of bit b. Different from before, here we handle a nonrouted bit as an individual object. Thus, two bits even in an object can have different routing styles in order to reach a higher routability. Since backbone generation provides a series of topologies for objects, every bit owns the same set of topologies according to that of backbones, i.e., $S_c(b)$. Assume that each candidate has an equal probability to be routed for bit b. We divide the sum of track usages from all the candidates by the set size. This calculation offers a close approximation of resource utilization by accumulating the bits in group g. Through considering all the bits' candidates, we obtain an estimated usage map of each concerned 2-D edge. Based on the map, we calculate the possible routing conflicts for each layer, as shown

$$cf(l, g) = \sum_{e_l \in e} \max(u(e, g) - \text{cap}_{e_l}, 0) \tag{8}$$

where e_l is the corresponding 3-D edge on layer l for 2-D edge e in (7), cap_{e_l} provides the available tracks for e_l, and $cf(l, g)$ is the estimated conflict value of routing g on layer l. In each routing direction, the layer with the minimum conflict has the highest probability to assign. As this conflict is based on an approximated congestion map, a legal solution can still be achieved even for a positive value. Therefore, an efficient clustering and routing scheme plays an important role to avoid the conflicts while keeping regularity.

B. Bottom-Up Clustering and Routing

In order to enhance the routability, it is feasible to allow different topologies for the bits in one object. In this way, each bit can be handled as an individual for routing so that a higher routability can be achieved. Based on the obtained layers, we are able to search for the solutions while encouraging the routability and topology sharing among all the bits. Taking the example in Fig. 7(b), there are three routing styles, instead of one in Section III-A. Here, one style corresponds to one cluster, where the bits share a common topology for regularity. To achieve this, we propose a bottom-up clustering strategy to handle multibit routing intelligently.

The whole procedure is listed in Algorithm 3, where a set of nonrouted groups serves as inputs. Initially, we produce the same set of topologies for the bits based on the backbones (line 1), and predict the layers with the highest probability for every group in both horizontal and vertical dimension (line 2). In each group, we construct a cluster for each bit so that they can be combined with others later (line 4). As a bottom-up clustering method, during each iteration, we ensure if there is a nonvisited pair of clusters (line 5). If so, we will select a pair with the minimum achievable cost (line 6). To obtain this cost, we employ a similar way of cost calculation in Algorithm 2. For two clusters, if neither of them is routed, all the candidates will be traversed and the feasible solutions will be recorded with the corresponding cost; if one of them has been routed successfully, we will only take the nonrouted cluster into accounts during the calculation. However, if no legal solution has been found, then a large penalty value will be counted. By considering all the available routes in two clusters, we acquire the minimum cost, which is set to the weighted sum of wire-length and regularity ratio. For the nonrouted cluster, the candidate route with the best cost will be adopted (lines 7–9), and this pair of clusters will be marked as a visited one (line 10). Also, based on the routing styles, we check the regularity ratio to see if they share the equivalent topology (line 11). In this case, these two clusters should be combined further and the second one will be removed (lines 12 and 13). Through traversing and combining the cluster pairs appropriately, the bottom-up scheme explores the solution space efficiently with adequate options for the signal bits.

Algorithm 3 Bottom-Up Clustering Algorithm

```
Require: A set of non-routed signal groups.
 1: Topology candidate generation for bits in groups;
 2: Possible layer prediction of signal groups;
 3: for each group do
 4:     Build one cluster $clus$ for each bit;
 5:     while $\exists$ non-visited pair of clusters do
 6:         Find a pair $(clus_1, clus_2)$ with the minimum cost;
 7:         if $clus_1, clus_2$ not routed then
 8:             Route with the minimum cost route;
 9:         end if
10:         Mark this pair as visited;
11:         if Ratio$(clus_1, clus_2) = 1$ then        ▷ Equation (2)
12:             Merge two clusters $clus_1$ & $clus_2$ into $clus_1$;
13:             Remove $clus_2$;
14:         end if
15:     end while
16: end for
```

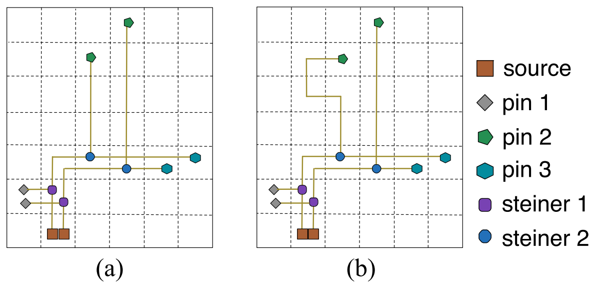

Fig. 9. Example of bit-based source-to-sink distance adjustment. (a) Pin 2 violates the distance constraint. (b) Violation is fixed by introducing detour for pin 2.

C. Post-Routing Refinement

The techniques above provide efficient routing control of signal bits through both top-down and bottom-up methodologies. Nevertheless, non-negligible source-to-sink distance variations result in possible signal malfunction. Different from classic bus routing, our flow deals with signal groups in which bits may have a different number of pins in various locations. Considering that the movement of one pin may disturb the other pins in a bit, the problem becomes more complicated. Meanwhile, topology regularity should also be taken into accounts. Therefore, we present the following routing refinement methodology which shrinks the distance difference with the consideration of regularity simultaneously.

As stated, it is likely that only partial pins in a bit violate the source-to-sink distance constraint. Thus, our target is to adjust the distances of the violating pins while maintaining the other pins' connections. An example is illustrated in Fig. 9, where two bits possess the same number of pins and each pair of mapped pins is signified with the same shape in one color. Observe that for these two bits the regularity ratio is equal to 1. However, in Fig. 9(a), large distance difference exists for pin_2 but pin_1 and pin_3 share similar values without exceeding the threshold. To handle this, we split the topology of each bit into a set of RCs and only consider those connecting to pin_2. This is to say, only the connection from $Steiner_2$ to pin_2 should be reconstructed. In this manner, not only does the problem size reduce, but the topology regularity is also under control by keeping the major topology. The resulting topology is shown in Fig. 9(b), where a twisting route is added for pin_2 to alleviate the distance violation.

The refinement details are provided in Algorithm 4, where a set of violating groups is taken as the input. First, we locate those bits which exceed the distance threshold. As the wire-length has been considered during primal-dual flow, there is little space to reduce the maximum distance for its close to optimality. Thus, we signify the bits whose pins show much shorter distances compared to the other mapped pins (line 1). The distances of the pins, i.e., dst_{p_v}, are recorded while the target distances, i.e., dst'_{p_v}, are also calculated (lines 2 and 3). Since there may be a few violating pins for a multipin bit, we traverse the original topology and locate the connections to be adjusted (line 4). Then we come to the details about allowing appropriate detours.

Algorithm 4 Post Routing Refinement

```
Require: Set of violating signal groups $g_v$s;
 1: Find violating bits $b_v$s and pins $p_v$s in $g_v$s;
 2: Calculate current distance $dst_{p_v}$ for $p_v$, $p_v \in b_v$;
 3: Calculate target distance $dst'_{p_v}$ for $p_v, p_v \in b_v$;
 4: Acquire connection $conn(p_v)$ for $p_v, p_v \in b_v$;
 5: for each group $g_v$ do
 6:     for each bit $b_v$ do
 7:         for each pin $p_v$ do
 8:             Get starting point $sp_{p_v}$ of $conn(p_v)$;
 9:             Get ending point $ep_{p_v}$ of $conn(p_v)$;
10:             if $conn(p_v)$ is horizontal then
11:                 VerticalShift($conn(p_v), dst'_{p_v}$);
12:             else if $conn(p_v)$ is vertical then
13:                 HorizontalShift($conn(p_v), dst'_{p_v}$);
14:             else
15:                 for $x \leftarrow 0$ to $dst'_{p_v}$ do
16:                     $y \leftarrow dst'_{p_v} - x$;
17:                     VerticalShift($conn(p_v), x$);
18:                     HorizontalShift($conn(p_v), y$);
19:                     if $conn(p_v)$ is updated then
20:                         Break;
21:                     end if
22:                 end for
23:             end if
24:             if $conn(p_v)$ is updated then
25:                 Re-connect $sp_{p_v}$ and $ep_{p_v}$;
26:             end if
27:         end for
28:     end for
29: end for
```

During detour production, our flow takes multilayer capacity constraints into careful consideration to avoid further overflows. Thus, the expected twisting route should complement the distance difference, i.e., $dst'(p_v) - dst(p_v)$, without any capacity violations. To exploit the residual available tracks, we allow the twisting route in four directions, i.e., left, right, lower, and upper directions. As shown in Fig. 10, three possible types of horizontal shifting (left and right) can make up for the distance deviation, where the red (blue) points refer to the starting (ending) points of this connection. Each type has an equal probability to be adopted as long as capacity constraints can be satisfied. Similarly, vertical (lower and upper) shifting is performed when the starting and ending points have the same Y co-ordinate. In the shifting methodology, as we focus on modifying the connections to the violating pins, topology regularity can still be maintained as much as possible.

Taking advantage of this shifting method, we traverse every violating pin p_v in the bit. Based on the acquired connection, we ensure its starting point, sp_{p_v}, and ending point, ep_{p_v} (lines 8 and 9). Then we investigate whether this connection is an L-shape or a straight horizontal/vertical connection. For the horizontal connection, we perform vertical shifting in either upper or lower direction (lines 10 and 11). Similarly, horizontal shifting adjusts the vertical connection (lines 12 and 13).

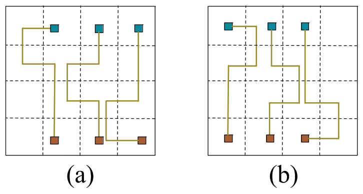

Fig. 10. Example of horizontal shifting for source-to-sink distance matching. (a) Left shifting. (b) Right shifting.

Nevertheless, for an L-shape connection, we search for twisting routes in both two directions and obtain more choices for tuning (lines 17 and 18). Due to the stringent capacity constraint, we traverse all the possible candidates for a legal solution. If it is found in both directions, then this searching procedure can be terminated to save the runtime overhead (lines 19–21). Based on the updated connection, the previous one is reconnected to form a new topology (lines 24–26). Since we build the new tree by traversing each violating pin, the final topology should be a connected tree structure without any loops. After visiting the violating pins in the certain bits and groups, the refinement stage returns the improved routes. With the slight degradation of wire-lengths, the source-to-sink distance deviation can be controlled efficiently.

V. Experimental Results

We implemented the proposed Streak framework in C++, and tested it on a Linux machine with eight 3.3 GHz CPUs. Meanwhile, we selected GUROBI [21] as our ILP solver. To evaluate its performance, we adopt seven industrial benchmarks with 10 nm technology node: `Industry1`–`Industry7`. Each benchmark provides a set of signal groups which require further identification and synergistic operations as individual objects. The details of each benchmark suite are listed in the left part of Table I. Here, column "#SG" provides the number of signal groups, and column "#Net" corresponds to the total number of nets. With the existing multipin benchmarks, the maximum pin number of all the nets is listed in column "$Np_{\max}$," and the maximum bit number in each benchmark is also listed in column "$W_{\max}$."

A. ILP + Primal-Dual Performance Comparison

Considering that few works handle signal routing of bundled bits with a varying number of pins in different directions, we obtain the manual designs by experienced designers from industry as shown in Table I. Column "Route" provides the routability of all the groups, and column "WL" provides the wire-length measured manually. Since Streak also targets at synergistic routing for bits bundled in groups, an evaluation metric, "Avg(Reg)," is listed to show the average routing regularity for all the routed groups so that the routing synergy can be reflected without relying on the routability. Equation (9) explains how to calculate Reg for each group

$$\mathrm{Reg} = \frac{2 \cdot \sum_{t_i, t_p \in g} \mathrm{Ratio}(t_i, t_p)}{N_o \cdot (N_o - 1)} \tag{9}$$

TABLE I
PERFORMANCE COMPARISONS ON 10 nm INDUSTRIAL BENCHMARKS

Bench	#SG	#Net	Np_{max}	W_{max}	Manual Design		ILP				Primal-Dual			
					Route	WL (10^5)	Route	WL (10^5)	Avg(Reg)	CPU (s)	Route	WL (10^5)	Avg(Reg)	CPU (s)
`Industry1`	230	3722	2	75	100%	7.01	99.13%	7.30	99.13%	5.7	99.13%	7.30	98.12%	0.8
`Industry2`	492	12239	2	136	100%	17.24	99.59%	17.93	98.75%	107.6	99.59%	17.93	98.14%	2.0
`Industry3`	234	4402	2	70	100%	7.41	98.72%	7.34	96.94%	> 3600	98.72%	7.34	96.94%	1.2
`Industry4`	146	3446	2	147	100%	7.82	100.00%	7.79	97.72%	4.6	100.00%	7.79	97.72%	0.6
`Industry5`	587	11185	14	77	100%	15.00	99.32%	17.12	90.27%	> 3600	98.64%	17.23	89.85%	149.5
`Industry6`	409	7278	9	256	100%	11.25	99.27%	11.40	91.84%	> 3600	99.27%	11.40	90.98%	143.1
`Industry7`	171	4087	7	147	100%	12.40	100.00%	12.47	95.82%	54.7	100.00%	12.47	95.02%	1.2
average	-	-	-	-	100%	11.16	99.43%	11.62	95.78%		99.34%	11.64	95.25%	
ratio	-	-	-	-	**1.00**	**1.00**	**0.9943**	**1.041**	–	–	**0.9934**	**1.043**	–	–

where t_i, t_p represent the solutions from any two objects i, p in group g, and N_o is the number of objects in this group which should be larger than 1. Explicitly, for two topologies with more mapped RCs, the ratio will be higher but still smaller than 100%. In real design, the majority of signal groups are routed for regularity and wire-length improvement. In this manner, the signal bits in one group are encouraged to share the parallel routes, as the example shown in Fig. 1, and the parallel connections are assigned on the same layer. For the residual signal groups with complicated routing styles, the commercial tool, ICC [22], is called to accomplish the whole design, so the regularity ratio may not be guaranteed with the integration of this commercial tool. Finally, column "CPU" provides the runtime in seconds.

From the experimental results, it is shown that compared to manual design, around 4% wire-length overheads exist in average for seven benchmarks from ILP, where primal-dual provides a slightly higher value. To make a fair comparison, we calculate the total wire-length including both the routed and nonrouted signal groups. For the nonrouted groups, we estimate the wire-length based on rectilinear Steiner minimum tree algorithm. Thus, the reported wire-length represents the routing condition of a whole design. And both the average routability for ILP and primal-dual are more than 99%. Meanwhile, for the regularity rate, ILP and primal-dual can reach over 95% for two-pin signal groups, and keep more than 88% for test cases with multipin signal bits. Considering that a bit may have sinks in different directions to the driver, the regularity rate has already been constrained and this value is reasonable. Due to the capacity constraint in our flow, there is no capacity violation for all the benchmarks.

Additionally, the problem becomes complicated with both congestions and multipin connections. For a multipin design with low congestion, e.g., `Industry7`, ILP provides a good performance in short runtime. Nevertheless, for those with serious congestions, the ILP runtime is prohibitively long, so we terminate the flow by setting a timing limit to 3600 s. Comparatively, primal-dual is able to achieve comparable wire-length, routability, and regularity rate much faster.

To provide a detailed comparison, we show the congestion densities for `Industry7` in Fig. 11 and `Industry6` in Fig. 12. Fig. 11(a) gives the congestion map from manual

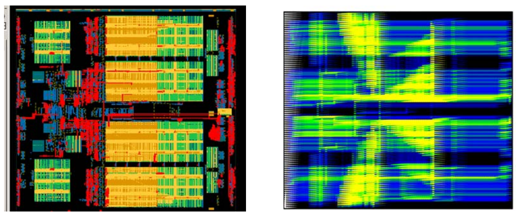
(a) (b)

Fig. 11. Routing congestion map for `Industry7`. (a) Manual design result. (b) Streak result.

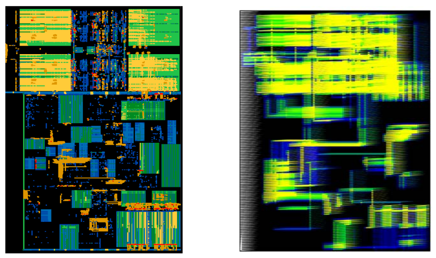
(a) (b)

Fig. 12. Routing congestion map for `Industry6`. (a) Manual design result. (b) Streak result.

design, where the red regions indicate hotspots with overflows and lighter regions indicate more congested routing conditions. Both with 100% as the routability, Streak in Fig. 11(b) allocates the routes in a balanced manner without any overflows. Meanwhile, regular routes can be observed with concurrent bending points. For a congested benchmark `Industry6` in Fig. 12, it is seen that the routes become complex for both manual and Streak result. Still, scattered overflow hotspots can be avoided by Streak efficiently. It is seen that with the slight sacrifice of routability, no overflow exhibits in Streak. Therefore, this comparison with manual designs proves the effectiveness of our tool to handle signal groups with synergistic routing styles.

To evaluate the algorithm scalability, we generate another large multipin benchmark based on `Industry2`, which offers the largest size among all two-pin testcases. During the generation, besides the existing two-pin connections, we insert some pseudo pins for the randomly selected groups so that the complicated routing styles can be obtained. Furthermore, the pseudo bits bundled in groups are also introduced to increase

TABLE II
PERFORMANCE COMPARISONS OF POST OPTIMIZATION ON 10 nm INDUSTRIAL BENCHMARKS

Bench	ILP	ILP + Post Opt					PD	PD + Post Opt				
	Vio(dst)	Vio(dst)	Route	WL (10^5)	Avg(Reg)	CPU (s)	Vio(dst)	Vio(dst)	Route	WL (10^5)	Avg(Reg)	CPU (s)
`Industry1`	12	0	100.00%	7.32	98.97%	10.1	12	0	100.00%	7.32	97.95%	4.7
`Industry2`	11	10	99.59%	17.98	98.54%	121.0	11	10	99.59%	17.98	97.93%	4.5
`Industry3`	10	0	99.15%	7.36	95.97%	> 3600	10	0	99.15%	7.37	96.00%	2.6
`Industry4`	6	2	100.00%	7.95	97.72%	4.7	6	2	100.00%	7.95	97.72%	0.8
`Industry5`	2	1	99.66%	17.16	90.25%	> 3600	2	1	99.15%	17.27	89.60%	198.2
`Industry6`	3	2	99.51%	11.40	91.30%	> 3600	3	2	99.76%	11.40	90.59%	145.7
`Industry7`	8	2	100.00%	12.66	95.82%	61.6	8	2	100.00%	12.66	95.02%	1.6
Average	7.4	2.4	99.70%	11.69	95.51%		7.4	2.4	99.66%	11.71	94.97%	
Ratio		**1.000**	**1.000**	1.000	1.000			**1.000**	**1.000**	1.002	0.994	

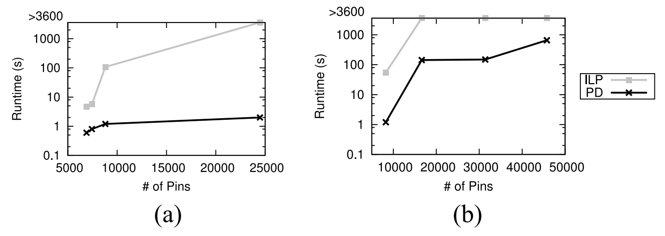

Fig. 13. Performance comparison on algorithm scalability. (a) Two-pin benchmarks. (b) Multipin benchmarks.

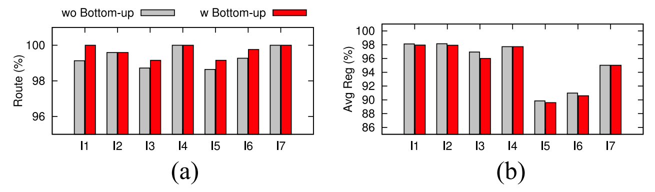

Fig. 14. Performance comparison of bottom-up clustering. (a) Impact on routability. (b) Impact on average regularity.

the potential conflicts. By conforming to the nature of signal routing, the pins of those generated bits are located in proximity. Then the scalability comparison in terms of total pins is provided in Fig. 13, where Fig. 13(a) shows the results of two-pin benchmarks, i.e., `Industry1`–`Industry4`, and Fig. 13(b) shows the results of multipin benchmarks. The number of pins in the largest benchmark is given as the rightmost point in Fig. 13(b). For two-pin benchmarks, we observe that primal-dual provides a better scalability in comparison to ILP, especially with a larger scale. Meanwhile, the runtime of primal-dual increases in a small amplitude. Comparatively, a worse scalability is seen for both ILP and primal-dual in Fig. 13(b). It is understandable because multipin connections lead to more complicated routing styles compared to two-pin connections, and the conflicts from various signal groups are also aggravated. Still, primal-dual exhibits a better scalability than ILP, as expected, which verifies the effectiveness of primal-dual for both two-pin and multipin benchmarks. To improve the scalability of ILP, we may adopt varying sizes of *G-Cell*s iteratively to solve the problem in a divide-and-conquer manner, as discussed in Section VI.

B. Effectiveness of Post Optimization

To prove the effectiveness of the post optimization, the results with and without this integration are listed in Table II. Besides the columns illustrated above, we compare another metric, i.e., "Vio(dst)," to evaluate the number of signal groups with the source-to-sink distance violation. To find an appropriate threshold value for each benchmark, here we set it to 50% of the maximum initial source-to-sink distance. The source-to-sink distance is the path length from the driver to the corresponding sink. In this setting, a few groups are marked as violated ones which exceed this given threshold and required to be adjusted through the refinement.

To demonstrate the effectiveness explicitly, we apply this post optimization to the solutions obtained from ILP and primal-dual, respectively. The number of violations before the post-optimization are listed in column 2 for ILP and column 8 for primal-dual. Both methods are with the same violation number, which indicates that primal-dual is able to achieve the similar results as ILP. Meanwhile, the other columns provide the results of violations, routability, wirelength, average regularity, and runtime for both methods after the post optimization. As shown in Table II, around 67% of violating groups can be fixed by introducing the extra detours, which verifies that the refinement stage has an efficient control of matching source-to-sink distances. Meanwhile, we observe that the routability values increase for both ILP and primal-dual, because the combination of layer prediction and clustering promotes more opportunities for signal bits to accomplish their routing. And the previous gap between ILP and primal-dual is also shrunk, which further validates its efficiency. Additionally, this post stage contributes to a slight degradation of wire-lengths compared with the initial results. Since the detours are induced to alleviate the violations during the post refinement phase, this increase of wire-length is expected and acceptable. Besides, the very similar wirelengths are seen from ILP and primal-dual, and the regularity ratio becomes slightly lower for both methods. In essence, handling every bit as an object will lead to a worse regularity ratio; however, the bottom-up clustering methodology still takes topology variance into careful consideration and our post optimization targets at complementary routing for the residual groups without distorting the global planning, so the regularity ratio is in well control. Therefore, according to the

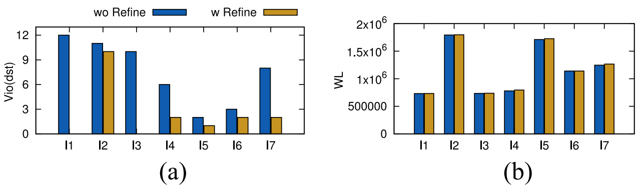

Fig. 15. Performance comparison of post refinement. (a) Impact on violations. (b) Impact on wire-length.

results, a higher routability and a slightly lower regularity ratio can be achieved due to the combination of layer prediction and bottom-up clustering, and the distance violations can be reduced greatly through the post refinement. After the post optimization, we still reach the similar performance of ILP and primal-dual, which implies the routing consequence from the global view is respected sufficiently.

Besides, we perform two experiments by excluding the bottom-up clustering and the refinement stage to prove the validity in Figs. 14 and 15. Fig. 14(a) shows that the routability can be improved by around 0.3%, consistent with the objective of the clustering strategy. Meanwhile, although we search for the solution with the consideration of regularity in Algorithm 3, it still pays a slight penalty of regularity ratio, as shown in Fig. 14(b). Considering that more routing styles are enabled to enhance the routability, as illustrated in Fig. 7, a relatively lower regularity ratio is accepted. In addition, since the refinement stage targets to reduce the wire-length deviation, we list the number of existing violations and wire-length in Fig. 15. From Fig. 15(a), the number of violations can be controlled efficiently through the proposed refinement, but it suffers from the wire-length penalty in Fig. 15(b), caused by the twisting overheads for distance matching. Because we only allow the necessary detours, the total overhead is negligible.

VI. Conclusion

In this paper, we have proposed a set of algorithms to generate synergistic topologies for on-chip signal groups. At first signal bits with distinctive connections are identified and then combined as routing objects with equivalent topologies. A mathematical formulation targets at wire-length and routability optimization while controlling the topology differences, followed by a fast flow to match a close quality with manual design and ILP results. To improve the routability of signal groups, a post-optimization stage allocates appropriate routes for each bit with the control of regularity. A post-routing refinement strategy then decreases the source-to-sink distance deviation of signal bits. The results show that our synthesis tool is able to provide efficient routing solutions with full legality and reasonable congestion map. In our future work, we plan to improve the potential scalability for large benchmarks, through a hierarchical and iterative method to divide the routing problem and solve them separately as subproblems. Meanwhile, we plan to take pin accessibility into consideration for more detailed exploration on signal routing.

Acknowledgment

The authors would like to thank Y. Xu and Y.-Y. Mo from Oracle for helpful comments and discussions.

References

[1] G. Persky and L. V. Tran, “Topological routing of multi-bit data buses,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Albuquerque, NM, USA, 1984, pp. 679–682.

[2] J. H. Y. Law and E. F. Y. Young, “Multi-bend bus driven floorplanning,” in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, San Francisco, CA, USA, 2005, pp. 113–120.

[3] F. Mo and R. K. Brayton, “A simultaneous bus orientation and bused pin flipping algorithm,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2007, pp. 386–389.

[4] S. Pasricha, N. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane, “Floorplan-aware automated synthesis of bus-based communication architectures,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Anaheim, CA, USA, 2005, pp. 565–570.

[5] H. Xiang, X. Tang, and M. D. F. Wong, “Bus-driven floorplanning,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 11, pp. 1522–1530, Nov. 2004.

[6] T. Ma and E. F. Y. Young, “TCG-based multi-bend bus driven floorplanning,” in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Seoul, South Korea, 2008, pp. 192–197.

[7] S. Pasricha, N. D. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane, “FABSYN: Floorplan-aware bus architecture synthesis,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 3, pp. 241–253, Mar. 2006.

[8] O. He, S. Dong, J. Bian, S. Goto, and C.-K. Cheng, “Bus via reduction based on floorplan revising,” in *Proc. ACM Great Lakes Symp. VLSI (GLSVLSI)*, 2010, pp. 9–14.

[9] H. Xiang, L. Deng, L.-D. Huang, and M. D. F. Wong, “OPC-friendly bus driven floorplanning,” in *Proc. IEEE Int. Symp. Qual. Electron. Design (ISQED)*, San Jose, CA, USA, 2007, pp. 847–852.

[10] P.-H. Wu and T.-Y. Ho, “Bus-driven floorplanning with bus pin assignment and deviation minimization,” *Integr. VLSI J.*, vol. 45, no. 4, pp. 405–426, 2012.

[11] D. H. Kim and S. K. Lim, “Bus-aware microarchitectural floorplanning,” in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Seoul, South Korea, 2008, pp. 204–208.

[12] T. Yan and M. D. F. Wong, “Untangling twisted nets for bus routing,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2007, pp. 396–400.

[13] J.-T. Yan, “Efficient layer assignment of bus-oriented nets in high-speed PCB designs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1332–1344, Aug. 2016.

[14] P.-C. Wu, Q. Ma, and M. D. F. Wong, “An ILP-based automatic bus planner for dense PCBs,” in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Yokohama, Japan, 2013, pp. 181–186.

[15] 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.

[16] A. B. Kahng and G. Robins, “A new class of iterative Steiner tree heuristics with good performance,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 7, pp. 893–902, Jul. 1992.

[17] Y. Yang *et al.*, “Layout decomposition co-optimization for hybrid E-beam and multiple patterning lithography,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1532–1545, Sep. 2016.

[18] B. Yu, D. Liu, S. Chowdhury, and D. Z. Pan, “TILA: Timing-driven incremental layer assignment,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 110–117.

[19] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, “TILA-S: Timing-driven incremental layer assignment avoiding slew violations,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 231–244, Jan. 2017.

[20] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, “Incremental layer assignment for critical path timing,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2016, pp. 1–6.

[21] *Gurobi Optimizer Reference Manual*, Gurobi Optim. Inc., Houston, TX, USA, 2016. [Online]. Available: http://www.gurobi.com

[22] (2012). *Synopsys IC Compiler*. [Online]. Available: http://www.synopsys.com

Derong Liu received the B.S. degree in microelectronics from Fudan University, Shanghai, China, in 2011. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

Her current research interests include physical design and design automation for logic synthesis.

Bei Yu (S'11–M'14) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

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

Dr. Yu was a recipient of the four Best Paper Awards at International Symposium on Physical Design in 2017, the SPIE Advanced Lithography Conference in 2016, the International Conference on Computer Aided Design in 2013, and the Asia and South Pacific Design Automation Conference in 2012, plus three additional Best Paper Award nominations at DAC/ICCAD/ASPDAC, and four ICCAD/ISPD contest awards. He has served in the editorial boards of *Integration*, *VLSI Journal*, and *IET Cyber-Physical Systems: Theory & Applications*.

Vinicius Livramento received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Santa Catarina, Florianópolis, Brazil, in 2013 and 2016.

He was a visiting Ph.D. student with the University of Texas at Austin, Austin, TX, USA, in 2016. He is currently a Software Design Engineer with ASML, Veldhoven, The Netherlands. His current research interests include timing and power optimization during physical design.

Dr. Livramento was a recipient of the First Place Award in the ISPD 2013 and in the ICCAD 2015 Contests.

Salim Chowdhury (M'86) received the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA, in 1986.

He taught at the University of Iowa, Iowa City, IA, USA, for some years, where he obtained multiple research grants from National Science Foundation. He had been researching with semiconductor industries since then, with Motorola, Chicago, IL, USA, Sun Microsystem, Santa Clara, CA, USA, and until recently with Oracle America, Redwood City, CA, USA.

Dr. Chowdhury was a recipient of the best paper award from DAC and holds many patents and publications.

Duo Ding received the Ph.D. degree with the University of Texas at Austin, Austin, TX, USA.

He is a Senior Staff Engineer with Samsung Austin Research Center, Austin, TX, USA, with a focus on EDA and machine learning/deep learning. He holds a number of publications and international awards in the above areas.

Huy Vo received the B.S. and M.S. degrees from the University of California at Berkeley, Berkeley, CA, USA.

He is a Professional Software Engineer with several years of industry experience. He has previously researched with other researchers in the design and implementation of HDL and microelectronic CAD tools. His current research interests include cloud computing and large scale data processing.

Akshay Sharma received the bachelor's degree in electronics and communication engineering with the Delhi College of Engineering, New Delhi, India, in 1999, and the Ph.D. degree in electrical engineering from the University of Washington (UW), Seattle, WA, USA, in 2005.

He has researched on the architecture and implementation of CAD algorithms that enable FPGA, high-speed processor, SRAM, and mixed-signal physical design during his Ph.D. studies. He also served a brief stint as an Engineering Lecturer with UW in the from 2009 to 2010. His current research interests include development of EDA algorithms, embedded systems, and software development methodology.

Dr. Sharma was a recipient of UW Electrical Engineering's Oustanding Research Assistant Award in 2005.

David Z. Pan (S'97–M'00–SM'06–F'14) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, CA, USA.

He is currently the Engineering Foundation Professor with the University of Texas at Austin, Austin, TX, USA. He has published over 280 refereed technical papers, and holds eight U.S. patents. He has graduated over 20 Ph.D. students, who are currently holding key academic and industry positions. His current research interests include cross-layer nanometer IC design for manufacturability, reliability, security, physical design, analog design automation, and CAD for emerging technologies.

Prof. Pan was a recipient of a number of awards for his research contributions, including the SRC 2013 Technical Excellence Award, the DAC Top ten Author in Fifth Decade, the ASP-DAC Frequently Cited Author Award, and the 14 best paper awards. He has served as a Senior Associate Editor for *ACM Transactions on Design Automation of Electronic Systems*, an associate editor for a number of other journals. He has served in the executive and program committees of many major conferences, including ASPDAC 2017 Program Chair and ICCAD 2018 Program Chair. He is a fellow of SPIE.