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

Derong Liu[1], Vinicius Livramento[2], Salim Chowdhury , Duo Ding[3], Huy Vo[3], Akshay Sharma[3], and David Z. Pan[1]

[1]University of Texas at Austin, Austin, TX, USA
[2]Federal University of Santa Catarina, Brazil
[3]Oracle Corp., Austin, TX, USA

ABSTRACT

As VLSI technology scales to deep sub-micron, design for interconnections becomes increasingly challenging. The traditional bus routing follows a sequential bit-by-bit order, and few works explicitly target inter-bit regularity for signal groups via multi-layer 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. Experimental results using industrial benchmarks demonstrate the effectiveness of the proposed technique.

1. INTRODUCTION AND RELATED WORK

As VLSI technology scales to deep sub-micron 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 Figure 1. Observe that there are three signal groups marked with different colors. The signal bits may have different numbers of pins and have to be routed in a regular type. 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 those performance-critical signal bits, the routability and wire-length should also be optimized to avoid functional inaccuracy and timing issues. Therefore, an advanced synergistic router should be able to not only reach optimal routability and wire-length but also guide each bit routing intelligently for design regularity.

To realize these requirements, we prefer to design 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 for those performance-critical signals, 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 for regularity. Meanwhile, instead of a bit-by-bit routing, signal bits can be clustered based on their possible route styles, as seen in Figure 1, where two styles exist in *Group*1, and each can 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 stringent. During the whole procedure, all these constraints should be taken into accounts carefully.

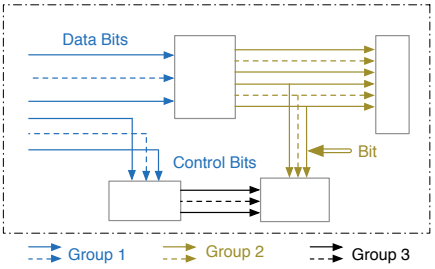


Figure 1: On-chip signal groups example.

There are few previous works focusing on bus architecture synthesis for on-chip designs. Some bus-oriented works incorporate with floorplanning to satisfy the timing constraints [4], minimize total bus area [5], or improve dead space [6]. Especially, multi-bend shapes are considered in [6] for providing more topology candidates through simulated annealing. And a bus thermal analyzer models the potential hot spots on chips [7]. There are also some works about escape routing on printed circuit board (PCB) design: such as pin ordering and untangling [8], layer resource minimization [9], and an automatic planning flow in [10] including bus decomposition, escape routing, layer assignment and global routing. Compared with these previous works, our synthesis tool provides a more extensive view to deal with bundled signal groups with more possibilities.

Very few of previous routing works target at synergistic topology generation and routing synthesis of signal groups with multi-pin connections. For current industrial designs, regular topologies with parallel routes are highly preferred to reduce inter-bit 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.

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

- An automatic framework directs synergistic routing and synthesis for bundled groups with multi-pin connections.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC'17, June 18-22, 2017, Austin, TX, USA

© 2017 ACM. ISBN 978-1-4503-4927-7/17/06…$15.00

DOI: http://dx.doi.org/10.1145/3061639.3062321

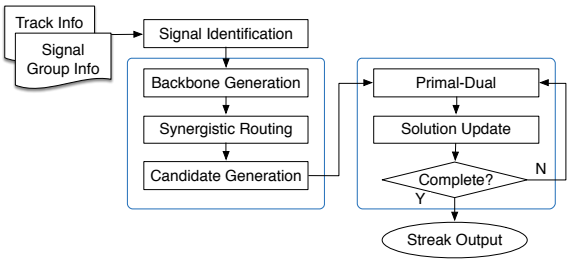


Figure 2: Overall Streak flow.

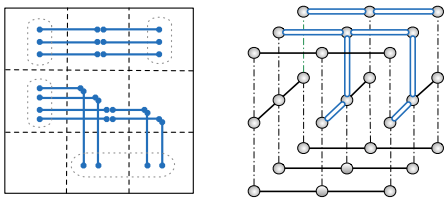


Figure 3: Signal routing model. (a) 2-D Routing; (b) 3-D Routing.

- An identification stage partitions signal groups into a set of objects where each bit has an equivalent topology.
- A mathematical formulation improves routability and wire-length while handling the topology similarity.
- A primal-dual flow benefits the runtime while keeping very comparable performance.

The remainder of this paper is organized as follows. Section 2 presents the overview of our framework and adopted models. Section 3 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 4 reports the experimental results, and followed by conclusion in Section 5.

2. 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.

2.1 Streak Flow

To provide an explicit view of Streak framework, the overall flow is illustrated in Figure 2. Initially, the information of specified track allocation and pin locations from bits bundled in signal groups is provided. Considering that the bits in a group may require various routing types, as shown in Figure 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. Then we construct a set of backbones for each object and derive equivalent topologies for each bit in an object. Since our framework targets at multi-layer structure, 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. The details of each step in the flow will be given in Section 3.

2.2 Proposed Model Description

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 sub-optimality 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**, shown as a vertex in Figure 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 $Group1$ from Figure 1 on a 2-D grid, as shown in Figure 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 parallel in a horizontal direction, and their corresponding 3-D solutions are provided in Figure 3(b), where the horizontal trunks are assigned on the same metal layer.

From this example, we present a novel model to distinguish 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 [11] 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-Cells, we prefer to use the relative direction rather than distance to describe each pin's location. As shown in Figure 4(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 8 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 in Equation (1),

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

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 Figure 4(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}. With the SV, these pins belonging to various bits of 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 bundled in groups.

2.3 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 topologies 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.*

3. ALGORITHMS

In this section, we present the technique details adopted through Streak flow. A pre-processing stage partitions each signal group

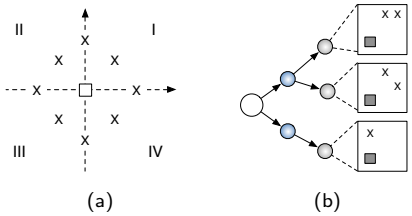


Figure 4: Example of signal identification: (a) Quadrant-based similarity vector; (b) Hierarchical isomorphic identification.

into a set of routing objects; a set of backbone structures are constructed and equivalent topologies are developed; a mathematical formulation selects the appropriate topology and assigns penalties to control irregular topologies; and a primal-dual flow is presented finally for speed-up.

3.1 Signal Isomorphic Identification

Besides covering general bus routing, our framework provides more feasibilities to handle groups of signals, which can be from messages or control information. A pre-processing stage provides a set of bits clustered in different groups, which can belong to multiple buses and share a similar topology. In comparison to general bus planning, the binding signals possess a different number 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 sub-groups, and deal with each sub-group 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 as shown in Figure 4(b), where each leaf node colored in grey represents an object containing a set of bits owning the same similarity vectors for all the pins. The methodology is intuitive but naive by calculating the similarity vector for each net pin with considerable calculation overhead. Therefore, we adopt a hierarchical strategy based on the premise that the driver pins of various bits in the same group can be mapped mutually. In this way, we calculate the similarity vector of the driver for each bit at first, and those with different vectors are separated as blue nodes in Figure 4(b). 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 for each bit due to the fact that the number of pins in each direction in comparison to the driver is quite limited. Then, for those bits with multiple pins in one direction to the driver, as shown in Figure 4(b), we only need to evaluate those pins located in one direction. Finally, those bits with the same SV for all the pins are combined as an object, and common topologies are encouraged for the objects from one signal group. In industrial designs, the signal groups are user-defined by referencing the specifications from many aspects, such as signal shielding, cell connection, etc.

3.2 Topology Generation and Evaluation

Before solving the signal routing problem, an efficient topology generation procedure is essential to provide promising alternatives. In this section, we propose a synergistic topology strategy for multi-pin connections which require equivalent topologies in one object and sharing topologies among objects in one group. It consists of backbone generation for objects, equivalent routes for bits and regularity evaluation among objects.

3.2.1 *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. To construct a set of backbones, we select one bit located in the center region of all the signal bits in an object and take its pins for backbone generation. Since the identification stage distinguishes the bits sufficiently, a selected bit can be representative of all the bits in one object.

For the topology generation, we extend the Batched Iterated 1-Steiner (BI1S) algorithm [12] based on an industrial flow. Since 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 form Steiner trees without increasing the wire-length. Through the combination of pins and inserted points with rectilinear connections, a Steiner tree can be obtained. Then we select a non-inserted point from the queue with the highest priority to construct another tree. For each tree, at least one different point is adopted. After visiting all the promising points at least once, we obtain an appropriate set of backbones on 2-D plane for post-processing.

3.2.2 *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 3.1 describes how to partition a signal group into a set of objects, and a set of backbones are constructed for each object in Section 3.2.1. 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 2.2. Through making sure the corresponding pin in backbone for each bit, we are able to generate a topology same as backbone.

After the identification stage, all the bits in one object have the same SV for each pin. Thus, it is direct 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. For a backbone, its pins have been traversed to record their SVs produced during the identification stage. In this way, a look-up table (LUT) is built by matching the SV with its corresponding pin in the backbone. Meanwhile, bending points in the backbone are also located with the same X/Y coordinates as its pins, based on Hanan grids, so each bending point can also be recognized by its corresponding pins. Then we traverse each bit for topology generation in reference to each given backbone.

For each bit, each pin can be mapped to its reflection in the backbone according to its SV through the LUT. With this corresponding relationship, we start to build the topology by calibrating the bending points in each signal bit. A non-visited bending point is selected arbitrarily from the backbone, where each point

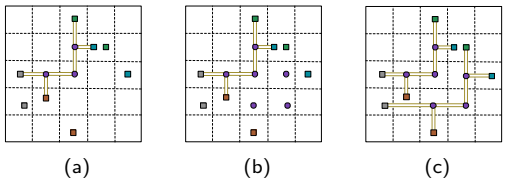


Figure 5: Equivalent topology generation example. (a) Pin(square) mapping through similarity vector; (b) Bending points(circles) aligning; (c) Topology generation by connecting mapping pins and points.

has both horizontal and vertical connections to the pins, and these pins are taken as the reference location of the point. Based on these pins, the corresponding horizontal and vertical pins in the signal bit can be acquired from the LUT. Then, the bending point in the bit can be located with the same X coordinate as the vertical pin, and Y coordinate as the horizontal pin. Through connecting the bending point with these neighboring pins with the same X/Y coordinates, an equivalent topology is able to be obtained. 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 each bit.

An explicit example is illustrated with three phases in Figure 5. Figure 5(a) provides the backbone and the mapped pins through LUT, while each corresponding pin is identified with the same color. With these pins, the internal bending points connecting pins are determined and aligned as shown in Figure 5(b). Finally, an equivalent topology is given through connecting the pins and inserted points for the specified bit in Figure 5(c).

Additionally, considering the existing multi-layer 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.

3.2.3 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 3.1, the pins in two bits can be mapped reciprocally according to their SVs when these two bits have the same number of pins, which could also be achieved through vertical sequences in FLUTE. However, for the bits 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 Equation (2). It is equal to the number of mapped rectilinear connections (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 Figure 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.

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

3.3 Mathematical Formulation

The mathematical formulation of Streak is provided in Formula (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 sub-optimality. The second item is to enforce the routing of objects and M is a large penalty for those non-routed 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 after identification. To minimize the topology variance, we add the third item in Formula (3). It helps to quantify the topology irregularity of objects in one group g, which is equal to the reciprocal of the regularity rate. For those topologies which do not share any rectilinear connections, a large number is to be assigned that should be smaller than M to guarantee routability. Meanwhile, for x_{ij} and x_{pq}, if they share any rectilinear connections but their assigned layers are not adjacent, a penalty proportional to the layer difference will also be assigned.

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

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 Figure 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).

3.4 Primal-Dual Algorithm

Although an ILP solver can be utilized to solve Formula (3), in real design it is not preferable due to its prohibitive runtime when a significant number of variables exist. We thus design a primal-dual algorithm to provide an efficient solution. With the generated topologies for each object, a fast and efficient 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 [13], which could also be applied in routing flow by incrementing the dual variables accordingly. This section provides the details how to solve the current routing flow through a primal-dual algorithm.

At first, we prefer to linearize the quadratic terms in For-

mula (3) for a primal formulation. Some previous works predefine one of these two variables as a known value through an iteration-based framework [14,15], while [16] takes the quadratic terms through an extensive Semidefinite Programming strategy for more accuracy. Considering the properties of primal-dual, we search for the allowable minimum value of each term based on the current states of x_{ij} and x_{pq}. Therefore, Equation (4) is utilized to provide a relatively accurate approximation:

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

where

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

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 combining topologies of x_{ij} and x_{pq} can still satisfy the current edge capacities. If not, this combining topology will be removed from the solution set. With this linear approximation, a dual problem ($\mathcal{DP}$) can be acquired as in Formula (6).

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

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

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

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

Formula (6) provides the dual form of Formula (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 Formula (3) can be determined by satisfying the constraints in Formula (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.

The outline of primal-dual algorithm is described in Algorithm 1, where the input is a set of routing objects with their candidate topologies. The initial primal solutions are set to 0 while keeping the dual variables also to 0 for their feasibilities (lines 1–2). 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 5–6). Notably, here x_{ij} should be able to satisfy the current edge capacity constraints without any potential violations. Then the value of x_{ij} increases to 1 while s_i is kept as 0 due to the primal constraint. With the integration of solution x_{ij}, we update the available routing tracks of each edge passed by x_{ij} (line 8). Meanwhile, considering the decreasing usable tracks, some x_{pq}s become infeasible and their values are not allowed to raise. 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 Formula (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 combining topologies of x_{ij} and x_{pq}, $c'(p,q)$ should be recalculated when some combining topologies are not available due to the reduced edge 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.

Algorithm 1 Primal-Dual Algorithm

Input: A set of routing objects with its candidate set.
1: Initiate primal solutions x_{ij}, s_i to 0;
2: Initiate dual solutions α_{ij}, β_{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**

4. EXPERIMENTAL RESULTS

We implemented the proposed Streak framework in C++, and tested it on a Linux machine with eight 3.3GHz CPUs. Meanwhile, we selected GUROBI [17] as our ILP solver. To evaluate its performance, we adopt seven industrial benchmarks with $10nm$ 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 1. Here column "**#SG**" provides the number of signal groups, and column "**#Net**" corresponds to the total net number. With the existence of multi-pins, the maximum pin number of all the nets is listed in column "$\mathbf{Np_{max}}$", and the maximum bit number in each benchmark is also listed in column "$\mathbf{W_{max}}$".

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 1. 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 groups in a whole benchmark. Equation (7) explains how to calculate **Reg** for each group,

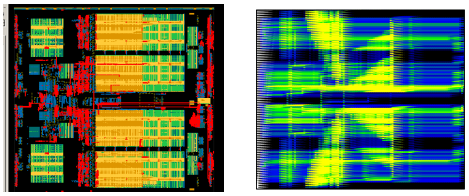
$$Reg=\frac{2\cdot\sum_{t_i,t_p\in g} Ratio(t_i,t_p)}{N_o\cdot(N_o-1)}, \quad (7)$$

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, designers prefer to guide a general routing of the major part while finishing the residual detailed connections through a commercial tool, so the resulting regularity rate 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, only around 1% wire-length overheads exist in average for seven benchmarks from ILP, where Primal-Dual provides a slightly better value due to its lower routability. 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 multi-pin signal bits. Considering

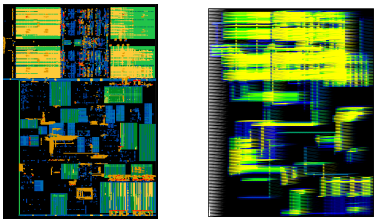
Table 1: Performance Comparisons on $10nm$ Industry Benchmarks

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%	6.80	98.70%	5.7	99.13%	6.80	97.25%	0.8
Industry2	492	12239	2	136	100%	17.24	99.59%	17.57	98.54%	107.6	99.59%	17.57	97.93%	2.0
Industry3	234	4402	2	70	100%	7.41	98.72%	7.05	95.66%	> 3600	98.72%	7.05	95.66%	1.2
Industry4	146	3446	2	147	100%	7.82	100.00%	7.79	97.72%	5.0	100.00%	7.79	97.72%	0.6
Industry5	587	11185	14	77	100%	15.00	99.32%	16.11	89.36%	> 3600	98.64%	15.50	88.95%	149.5
Industry6	409	7278	9	256	100%	11.25	99.27%	11.11	90.98%	> 3600	99.27%	11.11	90.18%	143.1
Industry7	171	4087	7	147	100%	12.40	100.00%	12.47	96.40%	54.7	100.00%	12.47	95.61%	1.2
average	-	-	-	-	100%	11.16	99.43%	11.27	95.34%	> 1567.6	99.34%	11.19	94.76%	42.6
ratio	-	-	-	-	**1.00**	**1.00**	**0.9943**	**1.010**	–	–	**0.9934**	**1.002**	–	–



(a) (b)

Figure 6: Routing congestion map for Industry7: (a) Manual design result; (b) Streak result.



(a) (b)

Figure 7: Routing congestion map for Industry6: (a) Manual design result; (b) Streak result.

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 edge capacity violation for all the benchmarks.

Additionally, it shows that this problem becomes more complicated with the increase of both multi-pins and routing congestion. For a multi-pin design with low congestion density, such as Industry7 for example, ILP provides a good performance in short runtime. Nevertheless, for those designs with serious congestions, the runtime of ILP is prohibitively long, so we terminate the flow by setting a timing limit to 3600s. Compared with ILP, the Primal-Dual flow is able to achieve comparable wire-length, routability and regularity rate much faster.

To provide a more detailed comparison, we show the congestion density for Industry7 in Figure 6 and Industry6 in Figure 7. Figure 6(a) gives the congestion map from manual design, where the red regions indicate the capacity overflows and lighter regions indicate more congested routing conditions. Both with 100% as the routability, the result from Streak in Figure 6(b) allocates the routes in a balanced manner without any overflows. Meanwhile, regular routes can be observed from Streak with concurrent bending points. For a more congested benchmark Industry6 in Figure 7, it is seen that the routes become more 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 design proves the effectiveness of our tool to handle signal groups with synergistic routing styles.

5. CONCLUSION

In this paper, we have proposed a set of algorithms to generate synergistic topology 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, while a fast flow matches a close performance with manual design and ILP results. The results show that our synthesis tool is able to provide efficient routing solutions with full legality and reasonable congestion map. In the future work, we plan to take pin accessibility into consideration for more detailed exploration of signal routing.

Acknowledgement

The authors would like to thank Bei Yu from CUHK, Yue Xu and Yu-Yen Mo from Oracle for helpful comments and discussions.

6. REFERENCES

[1] G. Persky and L. V. Tran, "Topological routing of multi-bit data buses," in *Proc. DAC*, 1984, pp. 679–682.

[2] J. H. Y. Law and E. F. Y. Young, "Multi-bend bus driven floorplanning," in *Proc. ISPD*, 2005, pp. 113–120.

[3] F. Mo and R. K. Brayton, "A simultaneous bus orientation and buses pin flipping algorithm," in *Proc. ICCAD*, 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. DAC*, 2005, pp. 565–570.

[5] H. Xiang, X. Tang, and M. D. F. Wong, "Bus-driven floorplanning," *IEEE TCAD*, vol. 23, no. 11, pp. 1522–1530, november 2004.

[6] T. Ma and E. F. Y. Young, "TCG-based multi-bend bus driven floorplanning," in *Proc. ASPDAC*, 2008, pp. 192–197.

[7] D. H. Kim and S. K. Lim, "Bus-aware microarchitectural floorplanning," in *Proc. ASPDAC*, 2008, pp. 204–208.

[8] T. Yan and M. D. F. Wong, "Untangling twisted nets for bus routing," in *Proc. ICCAD*, 2007, pp. 396–400.

[9] J.-T. Yan, "Efficient layer assignment of bus-oriented nets in high-speed PCB designs," *IEEE TCAD*, vol. 35, no. 8, pp. 1332–1344, 2016.

[10] P.-C. Wu, Q. Ma, and M. D. F. Wong, "An ILP-based automatic bus planner for dense PCBs," in *Proc. ASPDAC*, 2013, pp. 181–186.

[11] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," *IEEE TCAD*, vol. 27, no. 1, pp. 70–83, 2008.

[12] A. B. Kahng and G. Robins, "A new class of iterative steiner tree heuristics with good performance," *IEEE TCAD*, vol. 11, no. 7, pp. 893–902, 1992.

[13] Y. Yang, W.-S. Luk, D. Z. Pan, H. Zhou, C. Yan, D. Zhou, and X. Zeng, "Layout decomposition co-optimization for hybrid e-beam and multiple patterning lithography," *IEEE TCAD*, vol. 35, no. 9, pp. 1532–1545, 2016.

[14] B. Yu, D. Liu, S. Chowdhury, and D. Z. Pan, "TILA: Timing-driven incremental layer assignment," in *Proc. ICCAD*, 2015, pp. 110–117.

[15] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, "TILA-S: Timing-driven incremental layer assignment avoiding slew violations," *IEEE TCAD*, 2017.

[16] ——, "Incremental layer assignment for critical path timing," in *Proc. DAC*, 2016, pp. 85:1–85:6.

[17] Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com, 2016.