IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2024VLP0003

Publicized:2024/09/05

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on VLSI Design and CAD Algorithms (Submitted)*

Gridless Gap Channel Routing with Variable-width Wires

Masayuki SHIMODA[†a)], ***Student Member and*** **Atsushi TAKAHASHI**[†b)], ***Fellow***

SUMMARY In this paper, a routing problem for advanced chip designs is modeled as Gridless Gap Channel Routing (GGCR). GGCR is a routing problem to allocate variable-width trunks of nets to as fewer gaps as possible where a gap is defined horizontally between obstacles arranged regularly in the routing region. We propose Ceiling-and-Packing algorithm (CAP) for GGCR. CAP allocates the trunk of a net repeatedly so that each gap is filled as much as possible by adopting an appropriate order of allocation, and uses fewer gaps to complete the routing compared with conventional algorithms.

key words: *Gridless channel routing, Variable-width wire*

1. Introduction

With the recent technological advances, some types of advanced chips contain a number of components such as sensors, LCDs, and controllers for them which are scattered as regular pattern in routing layers and treated as obstacles during routing phase. In critical routing layers in such advanced chips [1, 2], gaps that are defined between obstacles, and a gap channel is defined. It tends to be a bottleneck of routing since many nets need to use them for their connections [3, 4]. To accomplish routing with a limited number of gaps, it is necessary to use them efficiently. Pins of nets are placed even inside the routing area since various modules are scattered throughout the chip in addition to obstacles. The variety of wire widths of nets needed inhibits efficient use of gaps. A gap channel is treated as gridless, and it is preferred to complete routing using a single horizontal wire segment for each net, i.e., dogleg is not preferred.

An example of critical routing layers in such chips is shown in Fig. 1. The areas horizontally traversing between the obstacles are routing areas (gray rectangles) called *gaps*. A gridless gap channel is defined as the union of these gaps.

Channel routing [5–7] is one of the important processes in automated layout design, especially for cell-based design, that significantly affects the quality of the chip, and many studies on it have been performed so far. Conventional channels have fixed pins of nets on its upper and lower boundaries and floating pins of nets on its left and right boundaries (Fig. 2).

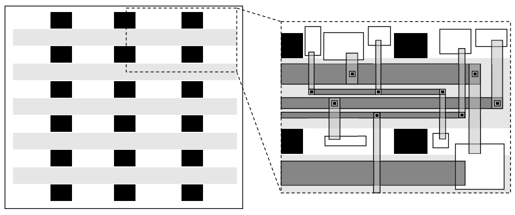


Fig. 1: Gridless gap channel. Gaps (gray) are defined horizontally between obstacles (black) regularly arranged. The pins of nets are defined on modules (while rectangles) inside area. The width of horizontal wires (dark gray) of nets varies.

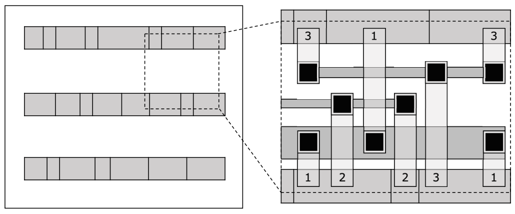


Fig. 2: Conventional channel.

In this paper, the aforementioned routing problem, especially focusing on horizontal wire segments of nets, is modeled as Gridless Gap Channel Routing (GGCR). In GGCR, the channel consists of vertically stacked fixed width gaps, and each net is connected by using a single horizontal wire segment, called the trunk of the net, with a specified width. The trunk of a net is allocated to a gap of the channel so that no overlaps with other trunks occur. The objective of GGCR in this paper is the number of gaps to complete the allocation of trunks of nets to gaps.

Even though vertical wire segments of a net have to connect the trunk and pins of the net to complete routing, the allocation of trunks of nets to gaps are focused in this paper, and no vertical constraint among nets is assumed. The optimization of vertical wires is out of concern in this paper, and it will be optimized in future works as in [1, 2].

Most of conventional channel routing algorithms, including Left-Edge (LE) [5], which is popular in conventional channel routing, neither consider the divided routing area (gaps) nor take the widths of wires into

†Tokyo Institute of Technology
a) E-mail: shimoda@eda.ict.e.titech.ac.jp
b) E-mail: atsushi@ict.e.titech.ac.jp

account, and cannot obtain good solutions in GGCR. Therefore, the development of a dedicated routing algorithm especially for GGCR is desired.

In this paper, we propose Ceiling-and-Packing algorithm (CAP) to obtain a good solution efficiently in GGCR. CAP allocates trunks repeatedly to gaps according to the priority defined on trunks. CAP gives a higher priority to a trunk with wider width. In addition, CAP considers the maximum density zones where the demand for the routing resources to allocate trunks of the remaining nets are the highest. A remaining net whose trunk does not cover the maximum density zones is postponed to allocate. Also, CAP allocates nets to gaps on a gap-by-gap basis to utilize a gap as much as possible.

The key contributions are outlined as follows:

- Recent routing problem is modeled as Gridless Gap Channel Routing (GGCR). In GGCR, variable-width trunks without vertical constraint are allocated to separated routing areas, called gaps. GGCR plays an important role in routing especially for critical routing layers which are often found in modern chips.
- Ceiling-and-Packing algorithm (CAP) is proposed for GGCR. CAP allocates the trunk of a net repeatedly so that each gap is filled as much as possible by adopting an appropriate order of allocation.

2. Gridless Gap Channel Routing (GGCR)

2.1 Background

This paper addresses Gridless Gap Channel Routing (GGCR).

In critical routing layers in advanced designs, a number of regularly pre-placed components that act as obstacles during the routing process exist. It is requested to connect the pins of nets by utilizing critical routing layers as much as possible. In GGCR, gaps are defined between obstacles, and form a *gap channel*. It tends to be a bottleneck of routing since many nets need to use these gaps for their connections. Due to the lack of horizontal routing resources in GGCR, the pins of a net in GGCR are connected by using a single horizontal wire segment in a critical layer as well as vertical wire segments in other layers.

In GGCR, the pins of nets exist at any location inside the channel, unlike conventional channel routing where pins exist only on the boundary of the channel. They are defined on modules placed inside the channel or on the boundary of the channel, and their coordinates are not absolute, as there is flexibility in module design and location. Since there is room to make adjustments of pin locations, in GGCR formulation in this paper, it is assumed that no vertical constraint among horizontal wire segments of nets exits. In case that conflicts among vertical wires occur, they will be resolved by utilizing flexibility in module design and location in followed detailed placement and routing [1].

In order to satisfy signal integrity constraints, the width of a wire used to connect pins of a net and the minimum spacing to adjacent wires are specified for each net. A wider width and a wider spacing are typically requested for a net that connect distant pins. Also, shielded wires are requested to put on both sides of wire for each sensitive net. These special wires are paid attention in EDA tools and they are sometimes routed in advance by hand in small designs.

2.2 Formulation

In GGCR, a critical routing layer is modeled as *gridless gap channel*, and the trunks of nets are allocated to gaps in the channel.

A gridless gap channel consists of gaps. Each gap is a rectangle area that spans the gap channel horizontally, and is defined to avoid obstacles, and contains no obstacle in it. The width (vertical length) and length (horizontal length) of the gap channel are denoted by C_v and C_h, respectively. The x-coordinate and y-coordinate of the lower left corner of gap g are denoted by $x(g)$ and $y(g)$. Also, the width (vertical length) and length (horizontal length) of gap g are denoted by $w(g)$ and $l(g)$, respectively, where $l(g) = C_\mathrm{h}$.

A net n consists of two or more pins to be connected by wires. The coordinate of a pin p is denoted by $(x(p), y(p))$. The minimum and the maximum x-coordinates of pins of net n are denoted by $x_{\min}(n) = \min_{p\in n} x(p)$ and $x_{\max}(n) = \max_{p\in n} x(p)$, respectively. For each net n, the *trunk* $T(n)$ of n is defined as the dedicated area for the horizontal wire segment of the net, including enough spacing to adjacent wires as well as shielded wires for the net if necessary. The width (vertical length) of the trunk of net n is denoted by $w(n)$. For a set of nets N, the set of trunks of nets in N is denoted by $T(N)$.

As an input, a set of gaps $G_\mathrm{in} = \{g_i\}_{i=1}^{k}$ where $x(g_i) = 0$, $l(g_i) = C_\mathrm{h}$, $w(g_i) > 0$, and $\mathrm{sum}_{g\in G_\mathrm{in}} w(g) \leq C_\mathrm{v}$ is given. Also, a set of nets $N_\mathrm{in} = \{n_i\}_{i=1}^{m}$, where $0 \leq x_{\min}(n_i) < x_{\max}(n_i) \leq C_\mathrm{h}$ and $w(n_i) \leq \min_{g\in G_\mathrm{in}} w(g)$ is given. It is assumed that the number of gaps are enough to accommodate to allocate trunks of all nets and that the y-coordinates of gaps are given so that gaps are not overlapped.

Let $a\colon T(N_\mathrm{in}) \to (G_\mathrm{in}, [0, W_g])$ be the allocation function of trunks to gaps where $a(n) = (g, y)$ represents that $T(n)$ is allocated to gap $g \in G_\mathrm{in}$ with offset y ($y \geq 0$) from the bottom of g, and where $W_g = \max_{g\in G_\mathrm{in}} w(g)$. It is also denoted by $a_\mathrm{g}(n) = g$ and $a_\mathrm{y}(n) = y$. The horizontal range of $T(n)$ is defined as $I_\mathrm{x}(n) = [x_{\min}(n), x_{\max}(n)]$. The vertical range of $T(n)$ by $a(n)$ is given as $I_\mathrm{y}(n) = [y(a_\mathrm{g}(n)) +$

Algorithm 1 First-Fit-Decreasing (FFD)

```
Require: set of items $U$, set of bins $G$
 1: $U \leftarrow$ DescendingWidthSort($U$)
 2: for all $b \leftarrow G$ do
 3:     $b \leftarrow \emptyset$
 4: end for
 5: while $U \neq \emptyset$ do                         ▷ descending order of size
 6:     $n \leftarrow$ delete($U$)
 7:     for all $b \leftarrow G$ do                         ▷ search fit bin
 8:         if $w(n) + \sum_{n' \in b} w(n') \geq w(b)$ then
 9:             continue
10:         end if
11:         $b \leftarrow b \cup \{n\}$             ▷ allocate $n$ to the first fit bin $b$
12:     end for
13: end while
```

$a_y(n), y(a_g(n)) + a_y(n) + w(n)]$.

The trunk $T(n)$ has to be allocated within a gap, that is, $a_y(n) + w(n) \leq w(a_g(n))$ has to be satisfied for all trunks $T(N_{\text{in}})$. Also, trunks have to be allocated to gaps without overlap, that is, for all trunks $T(n), T(n') \in T(N_{\text{in}})$, either $I_x(n) \cap I_x(n') = \emptyset$ or $I_y(n) \cap I_y(n') = \emptyset$ have to be satisfied.

Density of set of nets N at x is the sum of widths of trunks of nets N whose horizontal range contain x, and is denoted by $d(x, N) = \sum_{n \in \{n \in N | x \in I_x(n)\}} w(n)$. The maximum density of N is defined as $D(N) = \max_{x \in [0, C_h]} d(x, N)$. In order to complete the allocation of $T(N_{\text{in}})$ to G_{in}, $D(N_{\text{in}}) \leq \sum_{g \in G_{\text{in}}} w(g)$ should be satisfied. In case that $w(g) = W$ for all g in G_{in}, $\lceil D(N_{\text{in}})/W \rceil$ gives a lower bound of the number of gaps to complete the allocation.

The set of nets in N that contain x is denoted by $M(x, N) = \{n \in N \mid x \in I_x(n)\}$. The set of nets in N that horizontally overlap with $I_x(n)$ is denoted by $M(n, N) = \{n' \in N \mid I_x(n') \cap I_x(n) \neq \emptyset\}$.

The GGCR problem is to allocate trunks of nets to gaps so as to obtain the minimum number of used gaps, for given a net set N_{in} and a gap set G_{in}, as represented by

$$\begin{aligned} \min \quad & |\{a_g(n) \mid n \in N_{\text{in}}\}|, \\ \text{s.t.} \quad & a_g(n) \in G_{\text{in}}, a_y(n) + w(n) \leq w(a_g(n)), \forall n \in N_{\text{in}}, \\ & I_x(n) \cap I_x(n') = \emptyset \text{ or } I_y(n) \cap I_y(n') = \emptyset, \\ & \forall n, n' \in N_{\text{in}}. \end{aligned}$$

As the objective function, the minimization of the number of gaps used is adopted in this paper. It is unnecessary to use a small number of gaps when gaps are given as inputs. However, this objective function enables us to evaluate the performance of algorithms as well as the validity of chip design. Also, unused gaps can be utilized to improve the quality of chip design such as the total vertical wirelength.

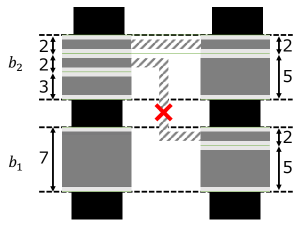


Fig. 3: An enhanced Bin Packing derived from GGCR.

2.3 Difficulty

GGCR is equivalent to Bin Packing problem when all trunks horizontally overlap with each other. There exists a trivial transformation from the decision version of Bin Packing to the decision version of GGCR. Bin packing is known to be NP-hard in general [8]. Therefore, GGCR is also NP-hard in general, and it is difficult to obtain an optimal solution efficiently in general.

Bin Packing is defined formally as follows:

BIN PACKING (BP)

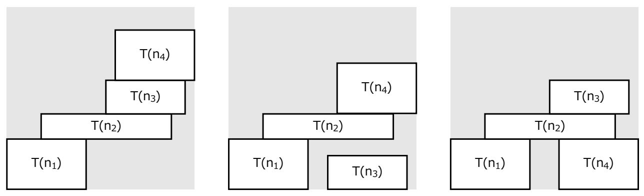
Instance: Finite set U of items, a size $s(u) \in \mathcal{Z}^+$ for each $u \in U$, a positive integer bin capacity B, and a positive integer K.

Question: Is there a partition of U into disjoint sets $U_1, U_2, \ldots, U_K$ such that the sum of the sizes of the items in each U_i is B or less?

Even though BP is NP-complete in the strong sense in general, it is solvable in polynomial time by exhaustive search if bin capacity is fixed [8]. However, exhaustive search is impractical even for a small bin capacity when the number of items is large. Therefore, various heuristics have been proposed for BP [9].

One of the heuristics for BP is First-Fit Decreasing (FFD). Algorithm 1 gives the pseudo code of FFD. FFD allocates an item in descending order of size to the first fit bin to be found. FFD is a heuristics, and does not necessarily find a good allocation. For example, FFD fails to obtain an optimal allocation when the multiset $\{3, 3, 3, 3, 2, 2, 2, 2\}$ is given as item sizes and bin size $B = 10$ [10, 11].

Examples of bin packing where bin size $B = 7$ are given in Fig. 3. There are two sets of items whose sizes are given as multisets $\{7, 3, 2, 2\}$ and $\{5, 5, 2, 2\}$ where items in each set are allocated to a bin in its corresponding set of bins. Let us assume that each set corresponds to the set of widths of trunks to be allocated to a portion of the channel in GGCR. FFD packs them successfully as $b_1 = \{7\}$, $b_2 = \{3, 2, 2\}$ and $b_1 = \{5, 2\}$, $b_2 = \{5, 2\}$, respectively. On the other hand, in GGCR, a trunk (item) has to be allocated to bins with the same index if it spans multiple portions of the channel. In case that two items with size 2 span two portion of the channel, it is impossible to allocate them in two gaps. Thus, GGCR is more complicated because GGCR is



(a) (n_1, n_2, n_3, n_4) (b) (n_1, n_3, n_2, n_4) (c) (n_1, n_4, n_2, n_3)

Fig. 4: Allocations by different orders.

a problem of accommodating 2-dimensional items with x-coordinates instead of 1-dimensional items. Therefore, an algorithm to find a better solution efficiently is required.

3. Related Works

The gridless channel routing [12,13] has been a subject of researches for a long time in conventional channels. The gridless approach allows an arbitrary location of terminals, nets, and vias, gaining popularity as well as approaches using fine grids in practical situations.

There are variations of gridless channel routing works: two layer [14,15], multi-layer [16,17], and routing in real geometries [18]. NLEA by B. Krishna et al. is one of the closest works that deals with various width wires without both dogleg and vertical constraint [19].

NLEA allocates trunks to lower left as much as possible with priority given to longer trunks among leftmost trunks that keep the right boundary of allocated region to the left as much as possible. However, these works are targeted to apply to problems on a single channel, whereas the target of our work is GGCR in which the channel is divided into the fixed width gaps. The difficulty of GGCR is high due to the fact that the target of allocation is multiple gaps instead of a single channel.

GGCR can be regarded as a placement problem for modules with fixed x-coordinates. Various studies have been conducted for various purposes in placement, and some of them consider not only the minimization of the area after the placement of the circuits. There have been many studies on data structure and search speed in layout design [20–22].

One of the issues in adapting the above to GGCR is how to evaluate a single solution and its high computational cost. In GGCR, it is necessary not only to pack trunks vertically but also to divide it into gaps.

4. Proposed Algorithm

4.1 Allocation Scheme

GGCR is a problem of allocating trunks on a two-dimensional plane without overlap in which x-coordinate of each trunk is fixed. Proposed algorithm allocates trunks to a gap by gap. For each gap, it allocates a trunk repeatedly as low as possible while avoiding jumping over other trunks allocated so far when it is dropped from above.

Formally, y-coordinate of trunk $T(n)$ is set to $a_{\rm y}(n) = \max_{n' \in M(n,N)}(a_{\rm y}(n') + w(n'))$ when the set of trunks $T(N)$ have been allocated to a gap and the trunk $T(n)$ is to be allocated to the gap.

Fig. 4 gives examples of allocation results of the set of four trunks $T(N_{\rm in}) = \{T(n_i)\}_{i=1}^{4}$ in different allocation orders. In Fig. 4(a), $T(n_3)$ is allocated on the top of $T(n_2)$ even if there is enough space below $T(n_2)$. This is not an optimal in terms of width used in the allocation. An optimal result is shown in Fig. 4(c). Note that there are various optimal results in terms of width used and that allocation orders that derive an optimal result are not unique.

The order of gaps to allocate trunks has no effect on the evaluation of the proposed algorithm. The gap order to improve the quality of other metrics is in our future works, and the order from the lowest gap among gaps not used so far is assumed in this paper for simplicity, i.e., $g = \arg\min_{g \in G}\ y(g)$ is selected as the gap to allocate, where G is the set of gaps not used so far.

4.2 Baseline: Left-Edge (LE)

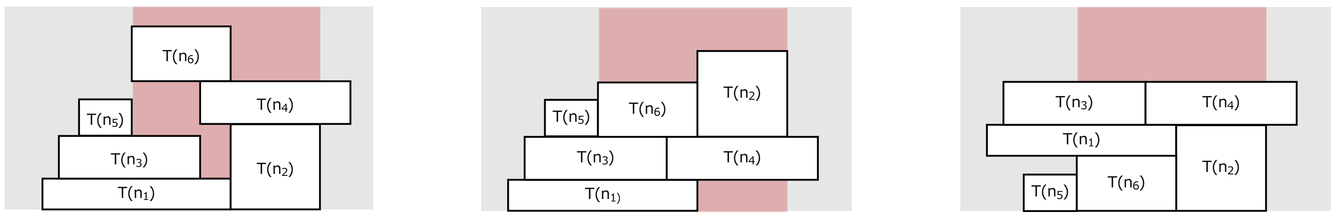
The proposed algorithm uses Left-Edge (LE) [5] as a baseline. LE is a well-known greedy algorithm for conventional channel routing where horizontal tracks are defined to which trunks are allocated. For each track, a trunk not allocated so far is allocated to satisfy horizontal constraints according to the priority defined by the leftmost principal from left to right if vertical constraints are not violated. LE uses the minimum number of tracks required if there is no vertical constraint.

Although LE allocates unit-width trunks to a track by track in conventional grid-based channels, tracks are not defined in GGCR since various width trunks exist. In the proposed algorithm, the iteration of allocation of trunks along horizontal direction from left to right is referred by *round*.

In the following, unless otherwise specified, LE refers an extended LE for GGCR. The pseudo code of LE for GGCR without vertical constraint is given in Algorithm 2. In each round, a trunk whose left end is the leftmost among the trunks not allocated so far is repeatedly allocated according to the allocation scheme if no violation occurs.

The function "LeftEdgeSort($N_{\rm in}$)" sorts trunks according to the leftmost principal. LE skips the allocation of a higher priority trunk if horizontal constraint (HC) is violated, that is, if an overlap of trunks occurs. Also, LE skips it if gap width constraint (WC) is violated, that is, if the trunk cannot fit to the gap.

The function "ceiling_width(n, N')", where $T(n)$ is to be allocated when the trunks $T(N')$ have been



(a) LE $(n_1, n_2, n_3, n_4, n_5, n_6)$ (b) NLEA$(n_1, n_3, n_5, n_4, n_6, n_2)$ (c) CAP $(n_6, n_2, n_5, n_1, n_3, n_4)$

Fig. 5: Examples of allocations. Red zone indicates the range of the maximum density of N_{in}.

Algorithm 2 Left-Edge (LE) for GGCR.

```
Require: set of nets $N_{\text{in}}$, set of gaps $G_{\text{in}}$
 1: $N \leftarrow$ LeftEdgeSort($N_{\text{in}}$), $G \leftarrow G_{\text{in}}$
 2: while $N \neq \emptyset$ do                         ▷ next gap
 3:     $g \leftarrow$ delete($G$)
 4:     $N' \leftarrow \emptyset$, $f \leftarrow$ T
 5:     while $f =$ T do                             ▷ next round
 6:         $x \leftarrow -\infty$, $U \leftarrow N$, $f \leftarrow$ F
 7:         while $U \neq \emptyset$ do              ▷ next allocation
 8:             $n \leftarrow$ delete($U$)
 9:             if $x_{\min}(n) \leq x$ then         ▷ violate HC
10:                 continue
11:             end if
12:             $y \leftarrow$ ceiling_width($n, N'$)
13:             if $y + w(n) > w(g)$ then            ▷ violate WC
14:                 continue
15:             end if
16:             $a(n) \leftarrow (g, y)$, $N' \leftarrow N' \cup \{n\}$   ▷ allocate $n$ to $g$
17:             $x \leftarrow x_{\max}(n)$, $f \leftarrow$ T
18:             $N \leftarrow$ delete($n, N$)
19:         end while
20:     end while
21: end while
```

allocated, gives the y-coordinate of trunk $T(n)$ according to the scheme, that is, $a_{\text{y}}(n)$ is set to $\max_{n' \in M(n,N')}(a_{\text{y}}(n') + w(n'))$. The rounds for a gap are repeated until no more trunk is allocated to the gap.

Note that, in GGCR, there is no guarantee that LE achieves the minimum number of gaps used in allocation, unlike conventional channel routing for uniform-width wires. The result obtained by LE is shown in Fig. 5(a) where the allocation order is $(n_1, n_2, n_3, n_4, n_5, n_6)$. The result obtained by NLEA is shown in Fig. 5(b) where the allocation order is $(n_1, n_3, n_5, n_4, n_6, n_2)$. An optimal result in terms of the width used is shown in Fig. 5(c).

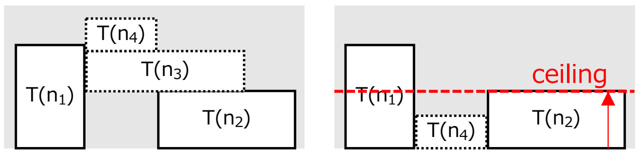
4.3 Proposal: Ceiling-and-Packing Algorithm (CAP)

We propose Ceiling-and-Packing algorithm (CAP) to use gaps efficiently in GGCR. CAP gives a higher priority to wider trunks in allocation as FFD adopts in BP. CAP allocates trunks repeatedly as LE does while trying to cover the maximum density zone as much as possible during allocation. Also, CAP adopts ceiling during allocation to allocate trunks as low as possible.

Algorithm 3 Ceiling-and-Packing (CAP)

```
Require: set of nets $N_{\text{in}}$, set of gaps $G_{\text{in}}$
 1: $N \leftarrow$ CAPSort($N_{\text{in}}$), $G \leftarrow G_{\text{in}}$
 2: while $N \neq \emptyset$ do                         ▷ next gap
 3:     $g \leftarrow$ delete($G$)
 4:     $N' \leftarrow \emptyset$, $C \leftarrow (w(g))$
 5:     while $C \neq \emptyset$ do                  ▷ next round
 6:         $Z \leftarrow \{x \mid d(x, N) = D(N)\}$
 7:         $x \leftarrow -\infty$, $U \leftarrow N$, $c \leftarrow$ top($C$)
 8:         while $U \neq \emptyset$ do              ▷ next allocation
 9:             $n \leftarrow$ delete($U$)
10:             if $x_{\min}(n) \leq x$ then         ▷ violate HC
11:                 continue
12:             end if
13:             if $(x, x_{\min}(n)) \cap Z \neq \emptyset$ then   ▷ violate ZC
14:                 continue
15:             end if
16:             $y \leftarrow$ ceiling_width($n, N'$)
17:             if $y + w(n) > c$ then               ▷ violate CC
18:                 continue
19:             end if
20:             $a(n) \leftarrow (g, y)$, $N' \leftarrow N' \cup \{n\}$   ▷ allocate $n$ to $g$
21:             $x \leftarrow x_{\max}(n)$
22:             $C \leftarrow$ insert($y + w(n), C$)
23:             $N \leftarrow$ delete($n, N$), $U \leftarrow N$
24:         end while
25:         if $x = -\infty$ then                    ▷ no allocation in round
26:             $C \leftarrow$ delete($c, C$)
27:         end if
28:     end while
29: end while
```

The pseudo code of CAP is given in Algorithm 3. The function "CAPSort(N_{in})" sorts trunks according to the width of trunk, and then ties are broken by the leftmost principal. CAP skips allocation of a higher priority trunk if horizontal constraint (HC) is violated. Even though the trunk is not actually overlapped with trunks allocated in the round by allocation scheme, it is skipped if the left end of the trunk is to the left of the rightmost end of trunks allocated in the round. CAP also skips allocation if either zone constraints (ZC) or ceiling constraint (CC) is violated. The details of ZC



(a) No CC (b) $T(n_4)$ meets CC

Fig. 6: Ceiling constraint (CC).

and CC are explained in the following subsections.

The result obtained by CAP is shown in Fig. 5(c) where the allocation order is $(n_6, n_2, n_5, n_1, n_3, n_4)$, while the priority of trunks used in CAP is $(n_2, n_6, n_3, n_4, n_5, n_1)$.

4.4 Zone Constraint (ZC)

For a set of trunks $T(N)$ not allocated so far, the density of N at any location gives a lower bound of the width of the channel that are used for allocation of $T(N)$. In each round, CAP allocates trunks so that this lower bound is reduced as much as possible. At the beginning of each round, CAP defines the maximum density zone of the channel which is defined as the set of x-coordinates at which the density of nets not allocated is the maximum. That is, the maximum density zone Z is defined as $Z = \{x \mid d(x, N) = D(N)\}$. CAP allocates a trunk only when all the maximum density zone to the left of the trunk is covered by trunks allocated in the round so far. CAP skips the allocation of a trunk if there is the maximum density zone to the left of the trunk that is not covered by trunks allocated in the round, that is, if zone constraint (ZC) is violated.

For example, in Fig. 5(c), the priority of trunks used in CAP is $(n_2, n_6, n_3, n_4, n_5, n_1)$ where the first criteria is the the descending order of widths of trunks. At the beginning, the maximum density of the channel ranges from the left end of $T(n_6)$ to the right end of $T(n_2)$. CAP tries to allocate $T(n_2)$ first in the first round, but it is skipped since the maximum density zone to the left of $T(n_2)$ that is not covered exists. CAP allocates $T(n_6)$ first, then allocates $T(n_2)$ second, and terminates the first round.

4.5 Ceiling Constraint (CC)

CAP sets to the ceiling of allocation in each round to pack as many trunks as possible into a low location. A ceiling gives the upper limit of y-coordinate of trunks to be allocated in the round. CAP skips the allocation of a trunk if the location of the trunk exceeds the ceiling when the trunk is allocated according to the allocation scheme, that is, if ceiling constraint (CC) is violated.

In each round, the minimum y-coordinate among ceiling candidates is selected as the ceiling. A candidate of ceiling is the location of the top boundary of each trunk allocated. A candidate is selected as the ceiling of a round among the set of candidates C, and it is removed from C if nothing is allocated in the round. A new candidate is inserted to C when a trunk is allocated, but no duplicated y-coordinates are contained in C. The function "insert$(y + w(n), C)$" maintains C when a trunk is allocated. At the first round and the final round of a gap, the ceiling is set to the width of the gap.

Fig. 6 illustrates the effects of CC. Assume that the priority of trunks is given as (n_1, n_2, n_3, n_4), and that the trunks $T(n_1)$ and $T(n_2)$ have already been allocated, but $T(n_3)$ and $T(n_4)$ are not allocated yet. If CC is not enforced, the allocation shown in Fig. 6(a) is obtained where $T(n_4)$ is allocated above $T(n_3)$. While, if the top boundary of $T(n_2)$ is set to the ceiling, $T(n_3)$ violates CC, and $T(n_4)$ is allocated before $T(n_3)$ as shown in Fig. 6(b). CC gives a priority to a trunk if it can be allocated below the ceiling, and enable us to utilize the routing area effectively.

Table 1: Setting (w indicates trunk width).

scenario (prefix)	Pr(w) $w=1$	$w=2$	$w=3$	$w=4$
Sparse (c1)	0.80	0.10	0.08	0.02
Dense (c2)	0.50	0.30	0.15	0.05
FFD-killer (c3)	0	0.50	0.50	0

5. Experimental Results

In order to evaluate the proposed algorithm, three types of instances that consist of trunks of four different widths in different probabilities are prepared. They are used as inputs for multiple gaps as well as for a single gap.

The single gap has an infinite width virtually, and the total width is used to evaluate algorithms. In multiple gaps, it is requested to complete the allocation by using given gaps in practice. To evaluate algorithms, a gap set whose size is beyond the lower bound (LB) is used, assuming enough gaps are given as an input.

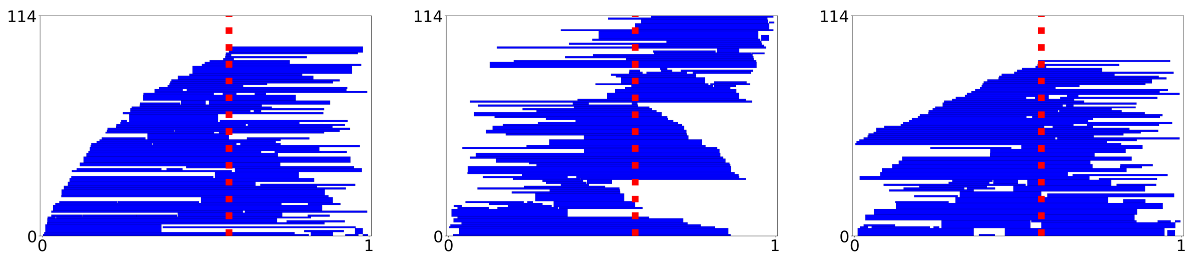
The benchmarks used in experiments were randomly generated where the minimum and maximum x-coordinates and width of each trunk are defined. The minimum and maximum x-coordinates of each trunk were generated uniformly from range $[0, 1]$, that is, $x_{\min}, x_{\max} \sim \mathcal{U}(0, 1)$. The width of each trunk w was selected among 1, 2, 3, and 4 according to the probabilities given in Table 1.

For comparisons, LE [5], NLEA [19], CAP, and derivations of CAP: CAP without allocation skip by ZC and CC, CAP without CC, CAP without ZC are used. They were implemented by Python 3.10.9, and executed on Apple M2 CPU. The execution time of LE, NLEA, and CAP was 0.3, 5.2, and 180.1 seconds, respectively, for the largest benchmark.

The results for the single gap are shown in Ta-

Table 2: The additional width for $w(g) = \infty$ in single gap.

Benchmark	Density	The additional width beyond Density (LB)					
		LE	NLEA	CAP (ours)			
(#trunk)	$D(N_{\text{in}})$			w/o ZC & CC	w/o CC	w/o ZC	CAP
c1-exp1 (10)	13	0	0	0	0	0	0
c1-exp2 (100)	91	7	23	1	1	0	0
c1-exp3 (500)	377	51	191	10	10	2	2
c1-exp4 (1000)	716	91	461	14	13	8	3
c2-exp1 (10)	16	0	0	0	0	0	0
c2-exp2 (100)	121	11	29	1	1	0	0
c2-exp3 (500)	497	71	261	4	4	1	1
c2-exp4 (1000)	941	135	602	17	14	7	5



(a) LE ($W = 98$) (b) NLEA ($W = 114$) (c) CAP (ours) ($W = 91$(optimal)).

Fig. 7: Results on c1-exp2 in single gap. Trunks (blue rectangles) and the maximum density zone (red dotted line).

Table 3: The additional gaps for $w(g) = 10$ in multiple gaps.

Benchmark	Density	LB	The number of gaps used beyond LB					
			LE	NLEA	CAP (ours)			
(#trunk)	$D(N_{\text{in}})$	$\lceil D(N_{\text{in}})/w(g) \rceil$			w/o ZC & CC	w/o CC	w/o ZC	CAP
c1-exp1 (10)	13	2	0	0	0	0	0	0
c1-exp2 (100)	91	10	0	2	0	0	0	0
c1-exp3 (500)	377	38	4	23	1	1	0	0
c1-exp4 (1000)	716	72	8	51	1	1	1	0
c2-exp1 (10)	16	2	0	0	0	0	0	0
c2-exp2 (100)	121	13	0	3	0	0	0	0
c2-exp3 (500)	497	50	5	31	1	0	0	0
c2-exp4 (1000)	941	95	11	68	1	1	0	0
c3-exp1 (10)	19	2	0	0	0	0	0	0
c3-exp2 (100)	161	17	0	6	1	1	1	1
c3-exp3 (500)	704	71	5	47	5	5	5	5
c3-exp4 (1000)	1343	135	11	112	9	9	9	9

ble 2. All algorithms achieve LB for 10 trunks. For LE and NLEA, the differences from LB increases as the number of trunks increases, and the difference are more than 10% of LB. On the other hand, the proposed algorithm achieves LB for 100 trunks, and the differences of widths are within 5 for 1000 trunks.

The results with the benchmark c1-exp2 are shown in Fig. 7. The red dotted line is drawn in the zone of the maximum density of N_{in}. It is confirmed that LE (Fig. 7(a)) and NLEA (Fig. 7(b)) are unable to cover all the maximum density zone, and that need more widths than LB to complete the allocation. On the other hand, the proposed CAP (Fig. 7(c)) covers all the zone, and the total widths used is the same as LB. Even when the routing area is not divided, the proposed CAP completes routing with less total widths, compared with conventional ones.

The results for the multiple gaps are shown in Table 3. In benchmark settings of c1 and c2, only CAP allocates trunks in the minimum number of gaps.

In the setting of FFD-killer c3, where FFD fails to achieve the minimum number of bins in most cases when the size of a bin is 10, CAP requires more number of gaps than LB, and even more than LE in c3-exp2. As mentioned in Section 2.3, like FFD, CAP will not be able to find certain combinations of trunks to fit a gap. However, as the number of trunks increases, LE also gradually increases the number of excess gaps, which is larger than that of CAP. Since the amount of trunks in an actual chip design is large, CAP is more appropriate than conventional algorithms.

6. Conclusions

In this paper, modern routing problem was modeled as GGCR, a variable-width routing problem with multiple gaps, and CAP for GGCR was proposed. GGCR has multiple routing areas, and the routing should be done so as to minimize the number of gaps used. CAP uses fewer gaps compared with conventional algorithms by prioritizing wide trunks and filling in the gaps as much as possible. Experimental results show that CAP achieves the least number of gaps compared to LE and NLE. Since an efficient solution of GGCR problem is necessary to realize a high-performance chip design, CAP contributes to these chip designs.

References

[1] Z. Wang, M. Shimoda, and A. Takahashi, "BCA channel routing to minimize wirelength for generalized channel problem," Proc. IEEE International Symposium on Circuits and Systems (ISCAS), 2024.

[2] Z. Wang, M. Shimoda, and A. Takahashi, "SDG channel routing to minimize wirelength for generalized channel," IEICE Trans. Fundamentals, vol.108, 2025. (submitted).

[3] K. Taniguchi, S. Tayu, A. Takahashi, M. Molongo, M. Minami, and K. Nishioka, "Two-layer bottleneck channel track assignment for analog VLSI," IPSJ Transactions on System and LSI Design Methodology, vol.17, pp.67–76, 2024.

[4] K. Taniguchi, S. Tayu, A. Takahashi, M. Molongo, M. Minami, and K. Nishioka, "A fast three-layer one-side bottleneck channel routing with layout constraints using ILP," IEICE Trans. Fundamentals, vol.108, 2025. (submitted).

[5] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," Proc. 8th Design Automation Workshop (DAC), pp.155–169, 1971.

[6] A.B. Kahng, J. Lienig, I.L. Markov, and J. Hu, Detailed Routing, pp.171–194, Springer International Publishing, Cham, 2022.

[7] T. Yoshimura and E. Kuh, "Efficient algorithms for channel routing," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.1, no.1, pp.25–35, 1982.

[8] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman and Co., New York, 1979.

[9] E.G. Coffman, J. Csirik, G. Galambos, S. Martello, D. Vigo, *et al.*, "Bin packing approximation algorithms: Survey and classification.," in Handbook of combinatorial optimization, pp.455–531, Springer, 2013.

[10] T. Yokomaru, T. Izumi, A. Takahashi, and Y. Kajitani, "Solution of integer bin packing problem with fixed capacity by FFD," IPSJ SIG Technical Reports (95-DA-76), vol.95, no.72, pp.1–8, 1995. (in Japanese).

[11] T. Yokomaru, T. Izumi, and Y. Kajitani, "The FFD bin-packing algorithm and its extension for VLSI circuit partitioning," IEICE Trans. Fundamentals, vol.J80-A, no.9, pp.1452–1459, 1997. (in Japanese).

[12] D. Hightower and R. Boyd, "A generalized channel router," Proc. 17th Design Automation Conference (DAC), pp.12–21, 1980.

[13] K. Sato, H. Shimoyama, T. Nagai, M. Ozaki, and T. Yahara, "A "grid-free" channel router," Proc. 17th Design Automation Conference (DAC), pp.22–31, 1980.

[14] H. Chen and E. Kuh, "Glitter: A gridless variable-width channel router," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.5, no.4, pp.459–465, 1986.

[15] C.H. Ng, "A "gridless" variable-width channel router for marco cell design," Proc. 24th Design Automation Conference (DAC), pp.633–636, 1987.

[16] P. Groeneveld, "A multiple layer contour-based gridless channel router," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.9, no.12, pp.1278–1288, 1990.

[17] D.B. Polkl, "A three-layer gridless channel router with compaction," Proc. 24th Design Automation Conference (DAC), pp.146–151, 1987.

[18] H.J. Rothermel and D. Mlynski, "Automatic variable-width routing for VLSI," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.2, no.4, pp.271–284, 1983.

[19] B. Krishna, C. Chen, and N. Sehgal, "Routing wires with non-uniform width and spacing in data paths," Proc. 11th International Conference on Microelectronics (ICM), pp.85–88, 1999.

[20] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," Proc. International Conference on Computer Aided Design (ICCAD), pp.472–479, 1995.

[21] P.N. Guo, C.K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," Proc. Design Automation Conference (DAC), pp.268–273, 1999.

[22] G.M. Wu, Y.C. Chang, and Y.W. Chang, "Rectilinear block placement using B*-trees," ACM Trans. Des. Autom. Electron. Syst., vol.8, no.2, pp.188–202, 2003.

Masayuki Shimoda received the B.E. and M.E. degree in engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2018 and 2020, respectively. He is currently a Doctoral student with the Department of Information and Communications Engineering of Tokyo Institute of Technology. His current research interests include machine learning and VLSI layout design.

Atsushi Takahashi received the B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He was at the Tokyo Institute of Technology as a Research Associate from 1991 to 1997, and as an Associate Professor from 1997 to 2009 and from 2012 to 2015. From 2009 to 2012, he was at Osaka University, Suita, Japan, as an Associate Professor. He is currently a Professor with Department of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology. His current research interests include VLSI layout design and combinational algorithms. He is a senior member of IEEE and IPSJ, and a member of ACM.