Information Processing Letters 47 (1993) 167–172
Elsevier

27 September 1993

Single jog minimum area joining of compacted cells

Andrew Lim
Information Technology Institute, National Computed Board, 71 Science Park Drive, Singapore 0511

Yeow Meng Chee
Planning and Infrastructure Department, National Computer Board, 71 Science Park Drive, Singapore 0511

Siu-Wing Cheng
Department of Computer Science, Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong

Communicated by K. Ikeda
Received 24 April 1992
Revised 29 July 1993

Abstract

Lim, A., Y.M. Chee and S.-W. Cheng, Single jog minimum area joining of compacted cells, Information Processing Letters 47 (1993) 167–172.

In this paper, we present an $O(n^2)$ time algorithm that obtains a single jog minimum area joining of two compacted cells.

Keywords: Algorithms; VLSI design; single jog; compacted cells; cell stretching; river routing; cell joining; symbolic sticks designs

1. Introduction

A popular design style involves tiling a two-dimensional area with compacted basic cells such that all needed inter cell connections are between pairs of adjacent cells [1,10]. Furthermore, adjacent cells have the same number of pins on their common boundary and the ith pins of the two cells are to be connected for all i. In the example of Fig. 1 the adjacent cells A and B each have four pins on their common boundary (adjoining sides). Pin a_i of A is to be connected to pin b_i of B, $1 \leqslant i \leqslant 4$.

Correspondence to: A. Lim, Information Technology Institute, National Computer Board, 71 Science Park Drive, Singapore 0511.

This connecting of pin pairs of the two adjacent cells is referred to as cell joining. Adjacent cells may be joined by stretching the cells so that the pin pairs line up. If cell A is stretched by y at any point x between a_1 and a_2, then the portion of A from x up is moved by y units (Fig. 1(b)). To join the pins of A and B by cell stretching it may be necessary to stretch A and B at several points. Figure 1(c) shows the effect of stretching A and B so as to join all pin pairs. Cell stretching clearly increases the area of the layout.

River routing [2–9] is an alternative to cell stretching. Only one routing layer is available. Figure 1(d) shows how cells A and B can be joined using river routing. River routing also increases the layout area.

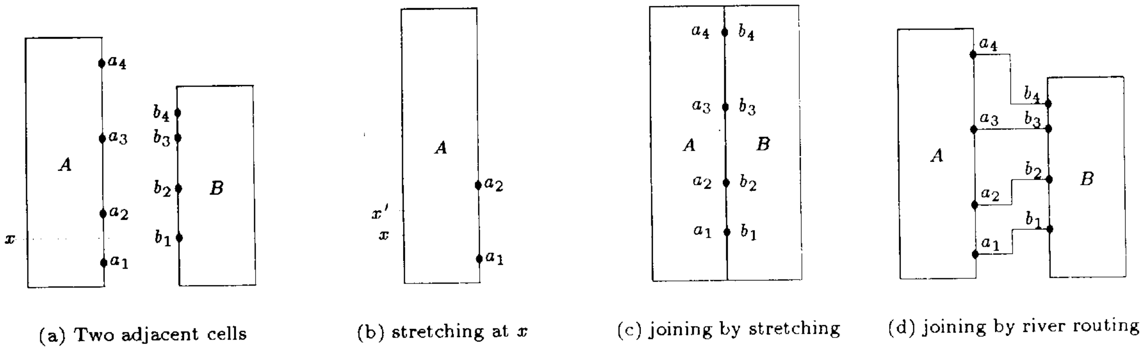

Fig. 1. Joining adjacent cells.

Lim, Cheng and Sahni [3] have studied and proposed efficient algorithms that combine cell stretching and river routing to join together pairs of adjacent cells together using unlimited number of jogs. Algorithms for the unlimited number of jogs problem cannot be used when there is a restriction on the number of jogs, i.e. when the number of jog is 1. The correctness of algorithms for the unlimited jogs case is based on results in [2] which do not hold for the case when each connection can use at most one jog.

In this paper, we show that pairs of cells can be joined optimally (i.e., with the smallest possible area) using at most one jog per connection in $O(n^2)$ time. A jog is a vertical segment of a connection. Due to this restriction, more tracks are usually needed for joining of cells than without this restriction. Single jog routing is some times preferred as the routing contains lesser jogs. Jogs decrease the manufacturing yield and reduce the reliability of the circuit. It is clear that if there are k tracks available for routing, at most k jogs are needed (see Fig. 2).

In the next section, we introduce our notation and terminology. The remainder of the paper develops the algorithm.

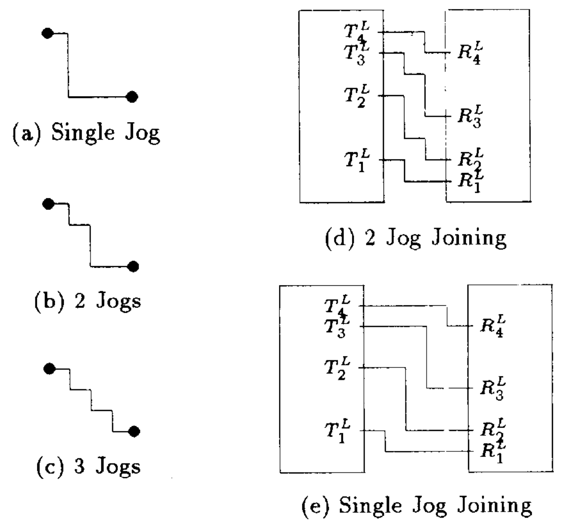

Fig. 2. Joining of cells using at most one jog per connection.

2. Notation and terminology

In this section, we formulate the joining problem for two cells L and R. Cell L is to the left of cell R and each cell has n terminals. The case when one cell is above another is symmetric to this. The terminals of L are on its right vertical boundary while those of R are on its left vertical boundary (Fig. 3(a)). Let T_i^A denote terminal i of cell A, $A \in \{L, R\}$. Let p_i^A, $A \in \{L, R\}$ be the position of T_i^A, $1 \leqslant i \leqslant n$. Let $p_0^A = 0$ and $p_{n+1}^A = h_A$ where h_A is the height of cell A. Since we assume a virtual grid system, all the $n+2$ p_i^As are integers. Let $\Delta_i^A = p_{i+1}^A - p_i^A$, $0 \leqslant i \leqslant n$. Stretching the cell A is equivalent to repositioning the terminals of A on the virtual grid line corresponding to the right edge of A in the case $A = L$ and left edge of A in case $A = R$. Let q_i^A be the position of terminal i of A following stretching, $1 \leqslant i \leqslant n$. Let $q_0^A = 0$ and $q_{n+1}^A = q_n^A + \Delta_n^A$. Then, the height of the stretched cell A is

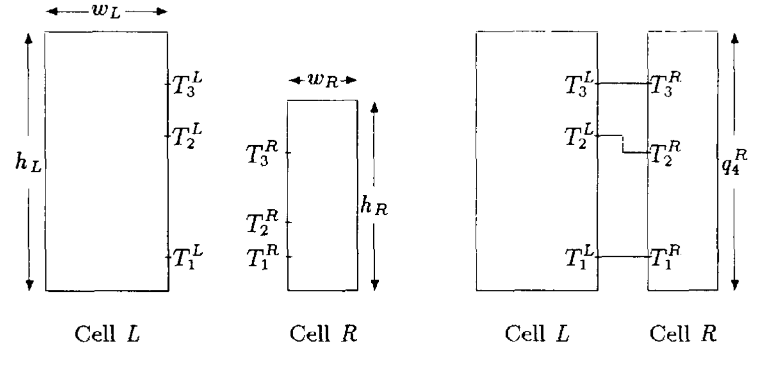

Fig. 3. Cells L and R.

q_{n+1}^A (Fig. 3(b)). The q_i^As define a *legal stretching* of cell A iff $q_{i+1}^A - q_i^A \geqslant \Delta_i^A$, $0 \leqslant i \leqslant n$.

Let s_{min} be the minimum number of tracks needed to route all terminals pairs (q_i^L, q_i^R), $1 \leqslant i \leqslant n$. The total area occupied by the stretched cells L and R as well as the inter cell routing is $(w_L + w_R + c(s_{min} + 1)) \times \max\{q_{n+1}^L, q_{n+1}^R\}$ for $s_{min} > 0$ and $(w_L + w_R) \times \max\{q_{n+1}^L, q_{n+1}^R\}$ for $s_{min} = 0$. w_L and w_R are, respectively, the widths of cells L and R, and c is a technology dependent constant that represents the minimum allowable track separation. The two cases can be compactly written as

$$(w_L + w_R + c \times \min\{s_{min}, 1\} \times (s_{min} + 1))$$
$$\times \max\{q_{n+1}^L, q_{n+1}^R\}.$$

The single jog minimum area joining (SJMAJ) problem is to find a legal stretching Q such that the total area occupied by the stretched cells as well as the inter cell routing is minimum.

```
Procedure SJMinHeight(s)
  /*Determine a minimum height legal stretching Q
    that can be routed using at most 1 jog and s tracks */
begin
  if $s = 0$ then
    for $i$:= 1 to $n$ do begin $q_i^L := q_{i-1}^L + \max\{\Delta_{i-1}^L, \Delta_{i-1}^R\}$; $q_i^R := q_i^L$; end
  else /* $s > 0$ */
    for $i$:= 1 to $n$ do begin
      $q_i^R := q_{i-1}^R + \Delta_{i-1}^R$; $q_i^L := q_{i-1}^L + \Delta_{i-1}^L$;
      if there is a violation of number of tracks used
      then begin
        Choose a $j$ s.t. overlap := $\max\{q_{j-1}^L, q_{j-1}^R\} - \min\{q_j^L, q_j^R\} + 1$
        is smallest and $i - s + 1 \leq j \leq i$;
        Push up the appropriate side by overlap and update those
        terminals affected by the push up at connection $j$;
      end; /* if */
    end; /* for */
end
```

Fig. 4. Procedure to find minimum height legal stretch using one jog.

```
Procedure FastSJMinHeight(s);
  /*Determine a minimum height legal stretching $Q$
    that can be routed using at most 1 jog and $s$ tracks */
begin
  if $s = 0$ then
    for $i$:= 1 to $n$ do begin $q_i^L := q_{i-1}^L + \max\{\Delta_{i-1}^L, \Delta_{i-1}^R\}$; $q_i^R := q_i^L$; end
  else begin /* $s > 0$ */
    set_empty($Q$); set_empty($\mathcal{L}$); $offsetL := 0$; $offsetR := 0$; $trackused := 0$;
    for $i$:=1 to $n$ do begin
      $q_i^L := q_{i-1}^L + \Delta_{i-1}^L$; $q_i^R := q_{i-1}^R + \Delta_{i-1}^R$;
      $t^L := q_i^L + offsetL$; $t^R := q_i^R + offsetR$;
      $overlapL := \max\{0, q_{i-1}^L - t^R + 1\}$; $overlapR := \max\{0, q_{i-1}^R - t^L + 1\}$;
      if $overlapL \le 0$ and $overlapR \le 0$
      then begin /* No overlap */
        clear_queue($Q$); $q_i^L := t^L$; $q_i^R := t^R$;
        $offsetL := 0$; $offsetR := 0$; addtolist($\mathcal{L}$,$i$);
        if $(q_i^L = q_i^R)$
          then $trackused := 0$
          else $trackused := 1$;
      end;
      else /* there is overlap */
      begin
        $overlapL := \max\{0, q_{i-1}^L - q_i^R + 1\}$; $overlapR := \max\{0, q_{i-1}^R - q_i^L + 1\}$;
        $overlap := \max\{overlapL, overlapR\}$;
        special_add($Q$,$q_i^L$, $q_i^R$, $i$, $overlap$, $overlapL$, $overlapR$);
    /*  special_add record to the queue $Q$ s.t. it will not allow
        any record just in front of it to have a bigger or equal $overlap$
        and will remove the record that violates the condition from the
        queue. This is applied recursively. */
        $trackused := trackused + 1$;
        if $trackused > s$ then begin /*track violation*/
          remove_front($Q$, $t^L$, $t^R$, $k$, $overlap$, $overlapL$, $overlapR$);
          $q_k^L := t^L + overlapL$; $q_k^R := t^R + overlapR$;
          $offsetL := offsetL + overlapL$; $offsetR := offsetR + overlapR$;
          addtolist($\mathcal{L}$,$k$);
          if $q_k^L = q_k^R$
            then $trackused := i - k$
            else $trackused := i - k + 1$;
          if $k = i$ then begin $offsetL := 0$; $offsetR := 0$; end
        end; /* track violation */
      end; /* else */
    end; /* for */
    for $i$:= 1 to length_of_list($\mathcal{L}$) do
      for $j$:= $L[i] + 1$ to $L[i+1] - 1$ do begin
        $q_j^L := q_{j-1}^L + \Delta_{j-1}^L$; $q_j^R := q_{j-1}^R + \Delta_{j-1}^R$;
      end;
end;
```

Fig. 5. A fast procedure to find minimum height legal stretch using at most one jog.

The SJMAJ problem can be solved in $O(n^2)$ time. This algorithm is based on an $O(n)$ algorithm to obtain a minimum height legal stretching when the number of tracks available for river routing is fixed.

3. The algorithm

Our single jog minimum area joining is based on an algorithm that obtains a minimum height legal stretching Q that can be routed using s tracks for any given $s \geqslant 0$. The algorithm, called *SJMinHeight* is given in Fig. 4.

Theorem 1. *Let $Q = \{q_i^A \mid 1 \leqslant i \leqslant n,\ A \in \{L, R\}\}$ be the set of terminal positions computed by procedure SJMinHeight. Q satisfies the following properties:*

(a) *Q is a legal stretching of cells L and R.*

(b) *Q can be routed using s tracks.*

(c) *Q uses at most one jog per connection.*

(d) *Let $D = \{d_i^A \mid 1 \leqslant i \leqslant n,\ A \in \{L, R\}\}$ be any s track routable legal stretching of cells L and R. $d_i^L \geqslant q_i^L$ and $d_i^R \geqslant q_i^R \geqslant q_i^{\tilde{R}}$, $1 \leqslant i \leqslant n$.*

(e) *Q is a minimum height legal stretching that is s track routable.*

Proof. (a) From the algorithm, it is clear that $q_{i+1}^A - q_i^A \geqslant \Delta_i^A$, $0 \leqslant i < n$. So, Q is a legal stretching. (b) The s tracks routability is taken care of in the **for** loop of the **if–then** statement. (c) same as (b). (d) this part will be proved by induction. The basis step when $i = 1$ is obvious. Suppose that (d) is true for q_i^A. Now consider q_{i+1}^A. If q_{i+1}^A and q_{i+1}^R can be connected without any adjustment, then $q_{i+1}^L \leqslant d_{i+1}^L$ and $q_{i+1}^R \leqslant d_{i+1}^R$. Otherwise, without loss of generality, the right side of a connection has been increased by the algorithm to make the $i+1$ connection feasible. To avoid ambiguity, we denote the terminals on the right side by $q_i^{\tilde{R}}$ before this change. It can be easily verified that if q_{i+1}^L and $q_{i+1}^{\tilde{R}}$ can be connected, then $q_k^{\tilde{R}} > q_{k-1}^L$ for some $i - s + 2 \leqslant k \leqslant i + 1$. Since this connection is not feasible, our algorithm selects a j, $i - s + 2 \leqslant j \leqslant i + 1$, such that $q_{j-1}^L + 1 - q_j^{\tilde{R}}$ is minimum and then increase $q_j^{\tilde{R}}$ to $q_j^R = q_{j-1}^L + 1$. Therefore, $q_{i+1}^R = q_{i+1}^{\tilde{R}} + q_{j-1}^L + 1 - q_j^{\tilde{R}}$. We know that $d_k^R \geqslant q_k^{\tilde{R}}$, $i - s + 2 \leqslant k \leqslant i + 1$, by the induction assumption. Since q_k^L, $i - s + 1 \leqslant k \leqslant i + 1$ have not been modified by our algorithm, $q_k^L \leqslant d_k^L$ by the induction assumption. Because $d_t^R > d_{t-1}^L \geqslant q_{t-1}^L$ for some t, $i - s + 2 \leqslant t \leqslant i + 1$ (by the feasibility criteria), implies that

$$\begin{aligned} d_{i+1}^R &\geqslant q_{i+1}^{\tilde{R}} + \left(d_t^R - q_t^{\tilde{R}}\right) \geqslant q_{i+1}^{\tilde{R}} + \left(q_{t-1}^L + 1 - q_t^{\tilde{R}}\right) \\ &\geqslant q_{i+1}^{\tilde{R}} + \left(q_{j-1}^L + 1 - q_j^{\tilde{R}}\right) = q_{i+1}^R. \end{aligned}$$

(e) follows from (d) and the observation that the height is $\max\{q_n^L + \Delta_n^L,\ q_n^R + \Delta_n^R\}$. □

The procedure *SJMinHeight* takes $O(ns)$ time as there are s updates whenever any terminal is shifted up. The procedure *FastSJMinHeight*, given in Fig. 5, is an efficient version of the procedure *SJMinHeight*. The procedure takes $O(n)$ time as the number of operations on the queue $\mathscr{Q}$ is $O(n)$. This is because each terminal pair can be added into the $\mathscr{Q}$ at most once and each time a terminal pair moves up a position in $\mathscr{Q}$, because it has a smaller *overlap*, will remove another terminal pair from $\mathscr{Q}$. Hence, there cannot be more than $O(n)$ such removals and moving up in the queue $\mathscr{Q}$. Whenever a q_i^A is removed from the front of the queue, its connection will be routed horizontally (i.e. uses no vertical track) or on the leftmost or on rightmost track, and will not be modified. So, the final positions of all the terminals can later be found by remembering those terminal pairs that are routed horizontally or on the leftmost or rightmost track. This is done using the list $\mathscr{L}$.

Let $Q(s)$ denote the s track stretching computed by procedure *FastSJMinHeight*. A minimum area stretching can be obtained in $O(n^2)$ time by computing $Q(j)$, $0 \leqslant j \leqslant n$ and determining which of these gives the smallest area. Note that for any $Q(j)$ the area can be computed in $O(1)$ using the formula in Section 2. The actual computing time can be reduced somewhat by first determining the maximum number of tracks needed when there is no stretching and then computing $Q(j)$ only for j in the range $[0, s_{max}]$. Since s_{max} can be as large as n, the overall complexity remains $O(n^2)$. The procedure for

```
Procedure SJFindSmax
  /*Determine the $s_{max}$ of 2 cells */
begin
  $s := 1$; $s_{max} := 0$; $i := 1$;
  while $i \le n$ and $s_{max} = 0$ do
  begin
    if ($q_i^R <> q_i^L$) then $s_{max} := 1$;
    $i := i + 1$;
  end
  if $s_{max} = 1$ then
  begin
    for $i := 1$ to $n - 1$ do
      if $q_{i+1}^R - q_i^L \le 0$ then $s := s + 1$
      else
      begin
        if $s > s_{max}$ then $s_{max} := s$;
        $s := 1$;
        if $q_{i+1}^L = q_{i+1}^R$ then $s := 0$;
      end;
    for $i := 1$ to $n - 1$ do
      if $q_{i+1}^L - q_i^R \le 0$ then $s := s + 1$
      else
      begin
        if $s > s_{max}$ then $s_{max} := s$;
        $s := 1$;
        if $q_{i+1}^L = q_{i+1}^R$ then $s := 0$;
      end;
  end;
end
```

Fig. 6. Procedure to find s_{max} for single jog routing of two cells.

computing s_{max} is given in Fig. 6. Unfortunately, the s value that results in minimum area cannot be found by a binary search in the range of [0, n]. This is because the area is not a monotone function of s [3].

4. Conclusions

In this paper, we have proposed an O(n^2) time algorithm for the single jog minimum area joining problem.

References

[1] D. Boyer, Symbolic layout compaction review, in: *Proc. ACM/IEEE Design Automation Conf.* (1988) 383–389.

[2] C. Leiserson and R. Pinter, Optimal placement for river routing, *SIAM J. Comput.* (1983) 447–462.

[3] A. Lim, S. Cheng and S. Sahni, Optimal joining of compacted cells, *IEEE Trans. Comput.*, to appear; Extended abstract in 1992 Brown/MIT Advanced Research in VLSI and Parallel Systems, pp. 99–113.

[4] A. Mirzaian, River Routing in VLSI, *J. Comput. System Sci.* (1987) 43–54.

[5] A. Mirzaian, A minimum separation algorithm for river routing with bounded number of jogs, in: *Proc. Internat. Conf. in Computer-Aided Design* (1989) 10–13.

[6] R. Pinter, On routing two point nets across a channel, in: *Proc. ACM/IEEE Design Automation Conf.* (1982) 899–902.

[7] R. Pinter, River-routing: Methodology and analysis, in: *Proc. Third Caltech Conf. on VLSI*, 1983.

[8] T. Tuan and S. Hakimi, River routing with small number of jogs, *SIAM J. Discrete Math.* **3** (1990) 585–597.

[9] T. Tuan and K. Teo, On river routing with minimum number of jogs, *IEEE Trans. Computer-Aided Design* **10** (1991) 270–273.

[10] N. Weste, Virtual grid symbolic layout, in: *Proc. ACM/IEEE Design Automation Conf.* (1981) 225–233.