INTERFERENCES OF RECTANGULAR PATHS
Jen-Shiun Chiang, Mi Lu and Yu-Ying J. Leung
Department of Electrical Engineering
Texas A&M University
1. Introduction

The interferences between a set of rectangular paths are considered in this paper. A path consists of a set of vertical and horizontal segments (v- and h-segments) where a vertical segment is connected to a horizontal segment. In VLSI design or PC board layout, it is desired to connect pairs of points by minimum length wires that do not interfere with each other. The interferences between the rectangular paths should be calculated first and then the interference-free paths can be extracted.

An interference between two rectangular paths can be divided into three categories [1]: a) overlap along a horizontal segment (h-overlap), b) overlap along a vertical segment (v-overlap), c) intersection between horizontal and vertical segments. Since the rectangular paths are shown in grid and have integer coordinates, we can use the coordinates to investigate overlaps and intersections. In the following sections, we present a parallel algorithm executing on a PRAM for solving the interference problem. The only known algorithm for solving this problem is sequential and is difficult to be parallelized [1].

2. General Approach

In our algorithm, a divide-and-conquer technique is applied. Consider a set of N h-segments which make up K given paths, where K is a power of 2 for illustration. Sort all the h-segments by their y-coordinates in a nondecreasing order, and break ties by their path Id. Partition the plane into two halves by a horizontal line that divides the set of h-segments into two equal-sized subsets so that all the y-coordinates of h-segments in the lower half are no greater than those of the segments in the upper half. Recursively perform the partition until each “strip” finally contains one h-segment.

The merge procedure is conducted in a binary tree fashion. Every two adjacent strips are merged concurrently, and thus log N iterations need to be conducted recursively. Lexicographically sort all the points in the upper strip and lower strip independently and concurrently by <X,Id,T/B>, and let the sorted list be u1,u2,...,un in the upper strip and l1,l2,...,ln in the lower strip. During the merge step, point li (ui respectively) is to be found according to point u (l) such that xj ≤ xi < xj+1 (x ≤ zj < zj+1).

To examine the horizontal overlaps, a counter HPi is maintained with each end point pi of a segment to represent the number of h-segments in a strip with the same y-coordinates, but different path Id, passing through the point to its right. In the merged strip, point li (ui) in the upper/lower strip can be provided with such information about point li (ui) in the lower/upper strip, and the number of the segments overlapping the segment ui(li) is on, HOi(HOj), should be increased by HPi(HPj). For example, in Fig. 1, HOj = 1 before the merge. Since HPi = 2, HOj = 3 after the merge. All the four segments in Fig. 1 have the same y-coordinates. To study their overlap status, we draw the one with the smaller path Id slightly lower. After all the merge steps have completed, the total number of horizontal overlaps will be given by ∑i HOi, for all the left point pi’s.

3. The Algorithm

The algorithm described here is to distribute O(2N) end points of the segments in the given paths on PE’s, with
one point per PE. The data record maintained in each PE
is as follows.

POINT record:
X integer /* x-coordinate */
Y integer /* y-coordinate */
Id integer /* Path id number */
HP integer /* Number of horizontal segments with
same y-coordinates, different path Id, passing
through to the right. */
SP integer /* Number of horizontal segments
extended to the right. */
VP integer /* Number of vertical segments passing
through extended to below. */
VT boolean /* 1 if a vertical terminal point, 0
otherwise. */
L/R boolean /* 0/1 if Left/Right point of a segment,
nil if a vertical terminal point. */
T/B boolean /* 0/1 if Top/Bottom point of a segment,
nil if a horizontal terminal point. */
C boolean /* 0/1 if the segment is
incompletely/completed. */
NT integer /* Number of top points of
incompleted segments with
smaller path Id in strip. */
NB integer /* Number of bottom points of
incompleted segments with
smaller path Id in strip. */
NC integer /* Number of completed seg-
ments with smaller path Id */
HO integer /* Number of h-overlaps. */
VO integer /* Number of v-overlaps. */
HV integer /* Number of intersections. */
TEMPn integer /* Copy of information
from another point. */

At the beginning of our algorithm, X, Y, Id, T/B, L/R,
VT, are known: HO, VO, HV, NT, NB, C, NC, HP, SP,
and VP are properly initialized in every processor. The
following algorithm will be recursively invoked to merge
the upper strip and the lower strip.

Algorithm INTERFERENCE

begin
/* Horizontal Overlap */
for each strip, in parallel do
(1.1) Sort the records POINT with VT = 0
in the strip into nondecreasing order using the
lexicographically ordered key <X,Id,T/B>.
for each point, in parallel do
begin
(1.2) Find in the other strip to be merged,
point j with z-coordinate no
greater than it and closest to it.
TEMPH = HP;
TEMPV = Y;
if Y = TEMPV then
begin
HO = HO + TEMPH;
HV = HP + TEMPH;
end /* if */
end /* for */
/* Intersection */
for each strip, in parallel do
(1.3) Sort all the records POINT in the
strip into nondecreasing order using the
lexicographically ordered key <X,Id,T/B>.
for each point, in parallel do
begin
(1.4) Find in the other strip to be merged,
point j with z-coordinate no greater than it and closest to it.
TEMPH = Xj; TEMPV = Tj/B;
TEMPV = Sp; TEMPd = Idj;
TEMPd = NTj; TEMPd = NBj;
end /* for */
for each point in a lower strip, in parallel do
/* check for completed segment */
begin
if (X = TEMPRX) ∧ (T/B = 1)
∧ (TEMPTj/B = 0) ∧ (Id = TEMPRd)
then /* find partner */
begin
C = 1;
RAW C = 1;
(2.1) NC = Prefix of Cj, Vi ∈
{i|X = Xj ∧ (T/Bi = 1)}
end /* then */
end /* for */
do step 2.2 and 2.3 in parallel
(2.2) for each point in an upper strip, in parallel do
if (C = 0) ∧ (T/B = 0) /* top point
of incompletely segment */
HV = HV + TEMPSP
end /* for */
(2.3) for each point in a lower strip, in parallel do
if (C = 0) ∧ (T/B = 1) /* bottom
point of incompletely segment */
HV = HV + TEMPSP
for each point, in parallel do
SP = SP + TEMPSP
/* Vertical Overlap */
if (C = 0) ∧ (X = TEMPRX) then
begin
do step 3.1 and 3.2 in parallel
(3.1) for each point in the upper strip, in parallel do
if T/B = 0 /* top point */ then
VO = VO + TEMPN + TEMPN + NC
else /* bottom point */
VO = VO + TEMPN + NC
end /* if */
(3.2) for each point in the lower strip, in parallel do
if T/B = 0 /* top point */ then
VO = VO + TEMPN + NC
else /* bottom point */
VO = VO + TEMPN + TEMPN + NC
end /* if */
else (C = 1) ∧ (X = TEMPRX)
for each point in the lower strip
if T/B = 1 /* bottom point of completed
segment */
VO = VO + TEMPN + NC
for each point, in parallel do
begin
NT = NT + TEMPN;
NB = NB + TEMPN;
end
end /* Algorithm INTERFERENCE */

In the above algorithm, for a strip containing k points,
the sorting executed in line (1.1) and (1.3) requires O(log k)
time. The binary search performed in line (1.2) and (1.4)
needs O(log k) time. The prefix computing involved in line
(1.2) takes O(log k) time. Other steps need constant time
only. Since the maximal size of k in a strip is O(N), and
log N iterations are needed to complete the merge, the time
performance of our algorithm is O(log^2 N), for N segments
in the given paths.

5. Conclusions
We have presented a method to calculate the interfer-
ences between the rectangular paths on 2N PE's in parallel.
Given a set of N horizontal segments of paths, our algo-

References
1983.