A Theoretical Study on Wire Length Estimation Algorithms for Placement with Opaque Blocks

Tan Yan*, Shuting Li, Yasuhiro Takashima, Hiroshi Murata
Graduate School of Environmental Engineering
The University of Kitakyushu
1-1, Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan

Abstract—How to estimate the shortest routing length when certain blocks are considered as routing obstacles is becoming an essential problem for block placement because HPWL is no longer valid in this case. Although this problem is well studied in computational geometry [6], the research results are neither well-known to the CAD community nor presented in a way easy for CAD researchers to ultilize their establishment. With the help of some recent notions in block placement, this paper interprets the research result in [1,8], which gives the best algorithm for this problem as we know, in a way more concise and more friendly to CAD researchers. Besides, we also tailor its algorithm to VLSI CAD application. As the result, we present a method that estimates the shortest obstacle-avoiding routing length in $O(M^2 + N)$ time for a placement with M blocks and N 2-pin nets.

I. Introduction

In an SA-based block placer, the total wire length of a placement has to be estimated millions of times. The HPWL (half perimeter wire length) model is a standard estimation model for such placers because of its fast speed. This model is fairly accurate when the blocks in the placement are regarded "transparent" to interconnections, i.e., they allow wires to go over them. However, as [4] points out, with the increasing use of IP blocks, more and more blocks in modern design are becoming "opaque" to interconnections (e.g., memory modules and noise-sensitive mixed-signal/analog/RF blocks do not allow exotic wires over them). Such blocks are regarded as obstacles for the routing and need to be considered in wire length estimation. Still using the HPWL model will greatly underestimate the wire length of actual routing (see Fig. 1) and may subsequently lead to serious problems such as timing violation, unroutable nets, etc., in later design stages.

In [4], the statistical distribution of wire length in the presence of obstacles is discussed. In our work, we provide a theoretical study on the exact wire length estimation, i.e., the shortest path length estimation for 2-pin nets in a placement configuration with the blocks as obstacles. Actually, this is already a well formulated problem in computational geometry where its full name is "rectilinear shortest path query with axis-aligned rectangular obstacles" and there exist many works on it. Mitchell [6] gives a survey on those works. However, those works are not well-known to the CAD community. The presentation in those works is also very computational-geometry-oriented and thus needs some interpretation to be applicable to VLSI design automation.

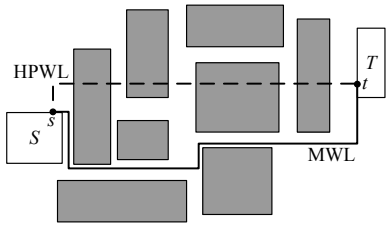


Fig. 1. The HPWL may underestimate the minimal wire length (MWL) of an actual routing.

In this paper, we present the theoretical results in [1, 8] in a way more compatible to the SA-based block placement framework. Our contribution lies in the following aspects:

- By introducing some recent notions from block placement research, (e.g., ABLR relation [10]) into the discussion, we greatly simplify the theory as well as the algorithm.
- The discussion in our paper is CAD-oriented and is therefore much more friendly to CAD researchers than the discussion in [1, 8].
- We tailor the theory in [1] so that it is applicable to VLSI design. In [1], only two extreme cases are discussed: the pins are located on the four corner vertices of the blocks and the pins can be located anywhere on the routing space. The later case leads to an increase in time complexity. We extend the discussion to the case when pins are located along the peripheries of the blocks, which is more realistic for VLSI CAD application, without increasing the time complexity. As the result, we present a method that estimates the shortest obstacle-avoiding routing length in $O(M^2 + N)$ time for a placement with M blocks and N 2-pin nets.

In the rest of this paper, we will restate the ideas in [1, 8] in a unique way, although some lemmas and theorems may find their origins in [1,8]. We will first present the problem formulation and some assumptions in Section II. Then we will present a graph based approach which runs in linear time in Section III. A further improvement to constant time will be shown in Section IV. Finally, we will conclude the paper in Section V.

*The author is now with University of Illinois at Urbana-Champaign, Email: tanyan2@uiuc.edu

1-4244-0630-7/07/$20.00 ©2007 IEEE.

II. Problem Setup

Before we start our discussion, we enforce several assumptions on the problem we will discuss:

Assumption 1. 1. The placement, i.e., the locations of the blocks, is known. 2. The blocks are non-overlapping rectangles. 3. The pins are located at the peripheries of the blocks. 4. The routing adopts the Manhattan geometry. 5. All the blocks are obstacles for routing. 6. The ABLR relations [10] of the blocks are already known.

The first four assumptions are quite reasonable and practical for VLSI design. The fifth is for the simplicity of discussion. It will be shown later that the algorithm can be easily extended to handle the problem in which only selected blocks are obstacles. The sixth assumption is special for our discussion, part of our theory is based on it. It is also a reason why our discussion is much more concise than that in [1] and [8]. The ABLR relation is universal for relation-based representations, e.g., Sequence-Pair [7]. It indicates whether a block A is located *above* or *below* or *left-to* or *right-to* another block B. If we say A is above B, then we mean that A's lower boundary is above B's upper boundary. The ABLR relation has the following properties: 1. *Uniqueness rule*, i.e., one and only one relation applies for a pair of blocks. 2. *Reversed-symmetry rule*, e.g., if A is above B, then B is below A. 3. *Transitive law*, e.g., if A is above B, B is above C, then A is above C.

If other representations, e.g., B*-tree [3], are used, we can translate the placement into a sequence-pair in $O(M \log M)$ time [5] and then obtain the ABLR relation.

Under these assumptions, we formulate the wire length estimation problem for a 2-pin net:

Definition 1. The *wire length estimation with opaque blocks* is to estimate the length of the shortest routing of a 2-pin net (s, t) that avoids all the blocks. This length is called the *minimal wire length (MWL)* of this net and a routing with this length is called an *MWL routing*.

Here we use (s, t) to denote a 2-pin net with s as the source pin and t as the target pin. In the rest of this paper, the term "the/a net" refers to the net (s, t) and "s"/"t" refers to the pins of this net. In some later situations, we will also use the symbol (u, v) to denote a routing or path between vertices u and v without introducing any confusion.

Without loss of generality, we assume the net satisfies:

Assumption 2. s is on block S and t is on block T, $S \neq T$. S is *left-to* T. $y_s \leq y_t$ (s is located lower than t).

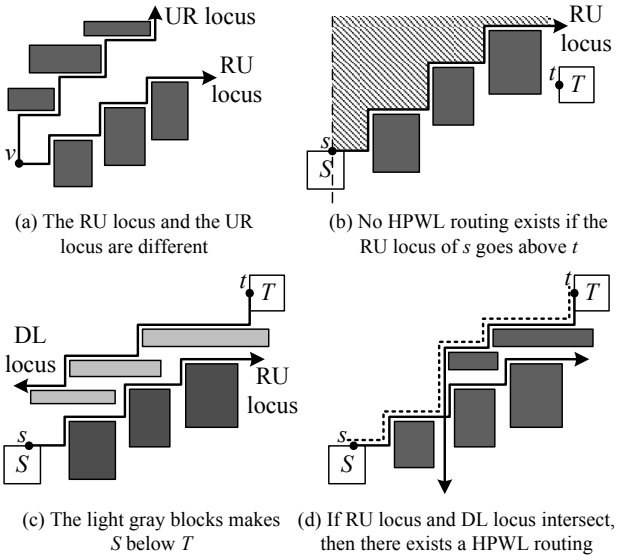
Fig. 1 gives an example of such a net. This paper discusses only the situation that satisfies Assumption 2. Other situations such as S is above T or $y_s > y_t$ can be discussed by vertically/horizontally flipping the placement or rotating the placement by $90°$.

III. Linear Time MWL Estimation

In this section, we will first give some theoretical results and then present the estimation algorithm whose correctness is guaranteed by the theory.

A. The Theory

Definition 2. Suppose an ant starts traveling from a point v and goes right. When it hits a block, it goes up until it reaches the top-left corner of that block and then it goes right again. The ant repeats such moves until it goes beyond the outline of this placement. The trace of this ant is defined as the *Right-Up (RU) locus* of v. The *Right-Down (RD), Left-Up (LU), Left-Down (LD), Up-Left (UL), Up-Right (UR), Down-Left (DL)* and *Down-Right (DR)* locus of v are defined similarly.



(a) The RU locus and the UR locus are different

(b) No HPWL routing exists if the RU locus of s goes above t

(c) The light gray blocks makes S below T

(d) If RU locus and DL locus intersect, then there exists a HPWL routing

Fig. 2. The definition and the proof of the theorem about locus.

Notice that different orders of the initials denote different loci. For example, the RU locus and the UR locus have different traces as in Fig. 2(a).

Definition 3. A routing is called x-*monotonic* if it intersects with any vertical line at most once (the intersection can be a vertical segment or a point). y-monotonic is defined similarly.

See Fig. 1 for an example, the solid routing (marked with "MWL") is x-monotonic but not y-monotonic. It is natural that:

Lemma 1. *The length of a routing is the same as the HPWL of the net (we call this routing* HPWL routing*)* $\Leftrightarrow$ *this routing is both x-monotonic and y-monotonic.*

With this, the following theorem holds:

Theorem 1. *The MWL of a net is its HPWL* $\Leftrightarrow$ *the RU locus of s goes below or through t.*

Proof. If the locus goes through t, the theorem naturally holds. Here we discuss the case when it goes below t:

$\Leftarrow$: We draw the DL locus of t. It must intersect with the RU locus of s. (Otherwise, S must be *below* T because of the transitive law of the ABLR relation as in Fig. 2(c). This violates Assumption 2.) By merging the two loci at the intersection, we find an HPWL routing (the dashed line in Fig. 2(d)).

$\Rightarrow$: If the RU locus goes above t as shown in Fig. 2(b), it divides the area to the right of s into two parts. Since the HPWL

routing must be both x and y monotonic, it must be in the upper part (the shaded area in Fig. 2(b)) and therefore cannot reach t. Thus, the RU locus must be below t. □

This theorem shows how to obtain the MWL when the RU locus of s goes below or through t. In the rest of *this section*, we will discuss about the other case, i.e., the case when the following assumption is true:

Assumption 3. The RU locus of s goes *above* t.

Definition 4. The *AB-region* of a net is the region enclosed by the RD and RU locus of s and the LD and LU locus of t. The boundary of this region forms two paths from s to t, the upper one is called *Path-A(bove)* and the lower one is called *Path-B(elow)*.

Fig. 3 gives an illustration of the definition. Notice that Path-A and Path-B are also included in the AB-region.

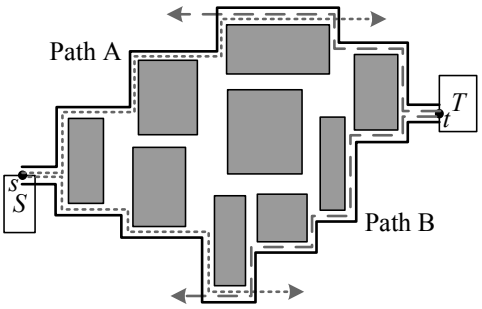


Fig. 3. The AB-region of a net.

Lemma 2. *There exists an MWL routing completely inside the AB-region.*

Proof. If the MWL routing is not within the AB region, there are two cases:

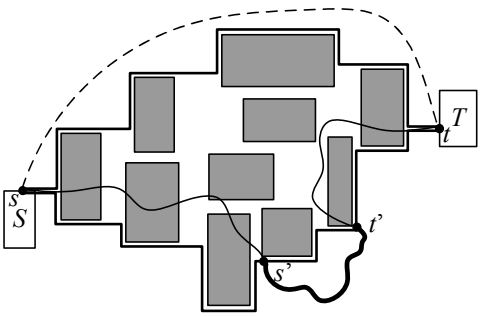


Fig. 4. There exists an MWL routing inside the AB-region.

• It is completely outside the AB region (see the upper dashed curve in Fig. 4). Then, it goes either above Path-A or below Path-B. In either case, its length cannot be shorter than that of Path-A or Path-B.

• Only part of it is outside the AB-region (see the thick curve from s' to t' in Fig. 4). Then we can replace the exceeding part by the routing from s' to t' along Path-A or Path-B and obtain a routing completely inside the AB-region. The new routing should not be longer than the MWL routing. □

Now our task is to find out this MWL routing inside the AB-region. In the following discussion, we show that it can be obtained by finding the shortest path on a DAG.

Definition 5. For each block in the placement, put a vertex at the center of its upper (lower) boundary and call it the *upper (lower) vertex* of this block. From a point p in the placement, draw a horizontal line to its right. The first block this line hits, if any, is called the *Right Adjacent Block (RAB)* of p (see Fig. 5(b)). If p is on the left boundary of a block, then its RAB is the block itself. From p to the two vertices on its RAB, define two directed edges. The one to the upper vertex is called the *Right-Up (RU) edge* of p and the other one is called the *Right-Down (RD) edge* of p (see Fig. 5(a)). The directions of these edges are from left to right (*rightward*). The length of each edge is the Manhattan distance between its two ends. Similarly, the *Left Adjacent Block (LAB)* and the *Left-Up (LU)* and *Left-Down (LD) edges* of a point (see the dashed edges of q in Fig. 5(a)) are defined. However, here we let the directions of the LU and LD edges *still be rightward*. If the LU/LD edge is identical with a RU/UD edge, the LU/LD edge is not defined. That is, we do not allow multiple edges between two vertices.

Comparing Fig. 5(a) with (b), one can see that:

Remark 1. Each edge corresponds to one actual HPWL routing. A path of such edges also corresponds to a continuous routing (but not necessarily with HPWL).

In the rest of this paper, the term "path" refers to the path composed of such edges and the term "routing" refers to an actual routing.

Due to this correspondence between the edge/path in the graph and the actual routing, we define the intersection of edges/paths as follows:

Definition 6. Two edges/paths are said to intersect with each other if and only if their corresponding routings intersect with each other.

We classify the edges into two categories:

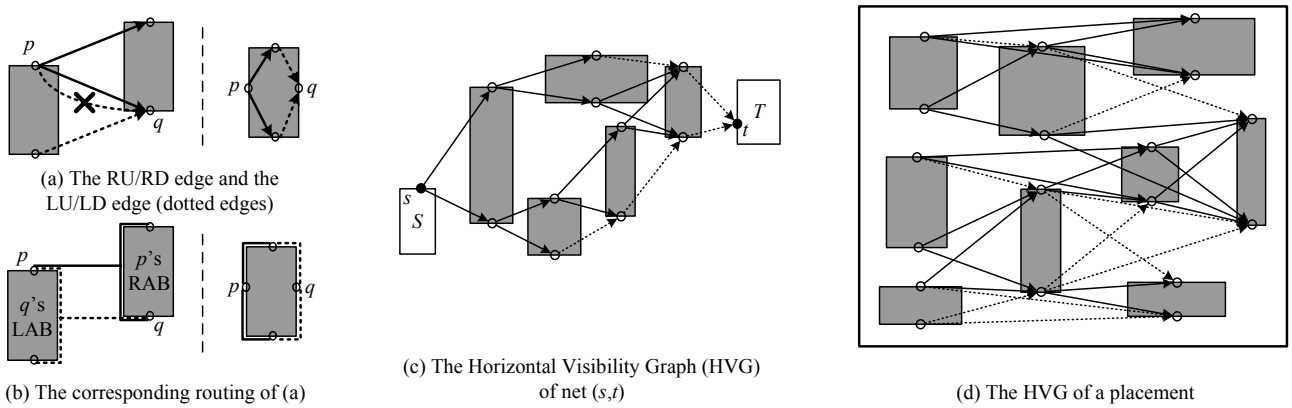
Definition 7. An edge is called *up-going* if the y-coordinate increases along its direction. Otherwise, it is called *down-going*.

We know that RU and LD edges are up-going and RD and LU edges are down-going. Following fact can be easily examined:

Lemma 3. *An up-going edge cannot intersect with another up-going edge.*

Since a locus corresponds to a path of edges of the same type (e.g., a RU locus corresponds to a path of RU edges), this Lemma also applies for up-going (RU or LD) loci. We call the corresponding path of the RU/RD/LU/LD locus of a vertex v the RU/RD/LU/LD path of v.

We now define a graph based on these edges (Fig. 5(c)):



(a) The RU/RD edge and the LU/LD edge (dotted edges)

(b) The corresponding routing of (a)

(c) The Horizontal Visibility Graph (HVG) of net (s,t)

(d) The HVG of a placement

Fig. 5. Routings, their corresponding edges and the Horizontal Visibility Graph (HVG).

Definition 8. The *Horizontal Visibility Graph (HVG)*[1] *of net* (s,t) is $G_{(s,t)} = (V,E)$, $V = \{$the upper and lower vertices of all the blocks in the AB-region$\} \cup \{s,t\}$, $E = \{$all the RU and RD edges between the vertices in $V\} \cup \{$the RU and RD edge of $s\} \cup \{$the edges belonging to the LU and LD path of $t\}$.

The following lemma ensures that the MWL routing is embedded in this graph:

Lemma 4. *There exists a path p from s to t on $G_{(s,t)}$ that corresponds to an MWL routing.*

Proof. Take arbitrary MWL routing r. Suppose it deviates from $G_{(s,t)}$ at vertex v which we call the *branching vertex* (see Fig. 6, the gray routing differs from $G_{(s,t)}$ at vertex v). There are two cases:

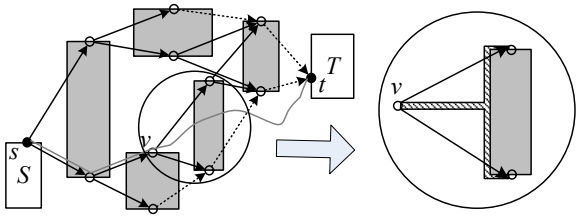


Fig. 6. An MWL routing is embedded in the HVG.

• v is on the LU or LD path of t (along the dotted edges in Fig. 6). Notice that the LU/LD path itself is an HPWL path, the path $s \rightarrow v \rightarrow$ LU/LD path of $t \rightarrow t$ is on $G_{(s,t)}$ and corresponds to an MWL routing.

• v is not on the LU/LD path of t (see v in Fig. 6). Then it must have RU and RD edges. By Lemma 2, there exists an MWL routing r' inside the AB-region of (v,t). We focus on the part of the AB-region between v and v's RAB (see the part inside the circle in Fig. 6). Notice that this part of the AB-region (the shaded region) has an area of zero, the MWL routing inside it must take either the RU or the RD edge of v. Then we find an MWL routing that follows $G_{(s,t)}$ at v. In other words, v is no longer a branching vertex. By repeating this, we could eliminate all the branching vertices and find a path on $G_{(s,t)}$ with MWL. □

[1]The concept of "visibility graph" is widely used in computational geometry [6]. However, the graph we define here is only a subgraph of that in [6].

The path p must be the shortest path in $G_{(s,t)}$. (Otherwise, the routing corresponding to the shortest path is shorter than the MWL routing.) Therefore, we have the following theorem:

Theorem 2. *The length of the shortest path from s to t in $G_{(s,t)}$ is MWL.*

Also notice that any path on the HVG corresponds to an x-monotonic routing because the directions of all the edges are rightward and each edge corresponds to an x-monotonic routing. Thus, the MWL routing is x-monotonic. Furthermore, by Lemma 1, Theorem 1 together with a discussion similar to the proof of Lemma 4, we can prove that:

Corollary 1. *Any MWL routing (no matter the RU locus of s goes above or below t) is x-monotonic.*

Obviously, $G_{(s,t)}$ is a DAG and the number of its edges $|E| = O(M)$. Therefore, the shortest path on it can be found in $O(M)$ time.

However, the time to build the graph overwhelms the estimation time. To solve the problem, we build one common HVG for all the nets (see Fig. 5(d)) instead of separate HVGs for different nets:

Definition 9. The *Horizontal Visibility Graph (HVG)* of a placement P is $G_P = (V',E')$, $V' = \{$the upper and lower vertices of all the blocks in $P\}$, $E' = \{$all the RU/RD/LU/LD edges of the vertices in $V'\}$.

Building G_P can be a preprocess before the estimation. When estimating a net, we add the RU/RD edges of s and LU/LD edges of t to G_P and it becomes a super-graph of $G_{(s,t)}$. Therefore, the shortest path in it must also be the shortest path of $G_{(s,t)}$.

B. The Algorithm

Now with all the theoretical discussion, we can build an algorithm (see Algo. 1) that calculates the MWL of (s,t). In the algorithm, tracing down the RU locus from s (line 4) can be done in $O(M)$ time by tracing along the RU path of s on the HVG. Finding the RAB or LAB (line 8) takes at most $O(M)$ time (by checking the blocks one by one). Finding the shortest path (line 10) takes $O(M)$ time because the HVG is a DAG.

Algo. 1 Estimate the MWL

```
1: Build the HVG G_P of the placement P
2: TotalWL = 0
3: for each net (s,t) do
4:    Trace down the RU locus of s
5:    if the locus goes below t then
6:       TotalWL = TotalWL + HPWL
7:    else
8:       Find RAB of s and LAB of t
9:       Add RU/RD edges of s and LU/LD edges of t to G_P
10:      Find shortest path from s to t on G_P
11:      TotalWL = TotalWL + ShortestPathLength
12:      Remove RU/RD edges of s and LU/LD edges of t from G_P
13:   end if
14: end for
```

Building the HVG of the placement (line 1) is done as a preprocess before wire length estimation. We use a plane sweep method (Algo. 2) to do this.

Algo. 2 Build the HVG $G_P = (V, E)$

```
1: Sort the blocks according to their x coordinates.
2: contour = φ, V = φ, E = φ
3: for the i'th leftmost block B_i do
4:    V = V + {v_i, v_j} // v_i, v_j = B_i's upper and lower vertices
5:    while can find vertex u in contour such that y(v_i) < y(u) <
      y(v_j) do
6:       E = E + {(u, v_i), (u, v_j)}
7:       contour = contour + {v_i, v_j} − {u}
8:    end while
9: end for
```

Since the operations (finding, adding and deleting vertices) on the *contour* set takes $O(\log M)$ time if *contour* is implemented by a height-balanced tree, the time complexity of Algo. 2 is $O(M \cdot \log M)$. Therefore, the total time complexity of Algo. 1 is $O(M \cdot \log M + N \cdot M)$. Notice that although the x-coordinate is used to order the blocks in Algo. 2, Γ^+ or Γ^- in Sequence-Pair [7] can also be used for the ordering because they follow the topological order in G_P.

IV. Constant Time Estimation using LUT

It can be seen that if multiple nets share the same RAB and LAB, repeating the shortest path computation is a waste of time (e.g., in Fig. 7, net (s_1, t_1) and net (s_2, t_2) share the same shortest path from block A to block B). An idea is

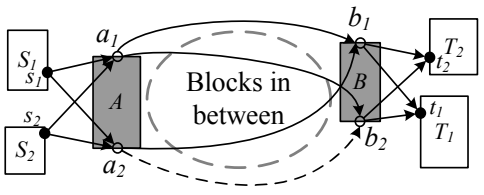


Fig. 7. The estimation could be constant time if the shortest paths (curved edges) between blocks are pre-calculated.

to pre-calculate all vertex-to-vertex distances in G_P and store them in a lookup table (LUT) before we estimate the nets. The RU/RD/LU/LD edges of s and t of all the nets are also pre-calculated and stored into a LUT. When estimating a net (say, (s_1, t_1)), we just select the shortest path among the four choices ($s_1 \to a_1 \to b_1 \to t_1$, $s_1 \to a_1 \to b_2 \to t_1$, $s_1 \to a_2 \to b_1 \to t_1$ and $s_1 \to a_2 \to b_2 \to t_1$). Since all the distances and the edges are pre-calculated, this takes only constant time.

To prove the correctness of the above idea, following points must be clarified (notice that in this section, Assumption 3 does not necessarily hold):

1. What to do if there exist no path between two vertices in the HVG (see the dashed edge in Fig. 7).
2. How to check whether the RU locus is above or below t within constant time.
3. How to obtain the RAB/LAB in the preprocess.

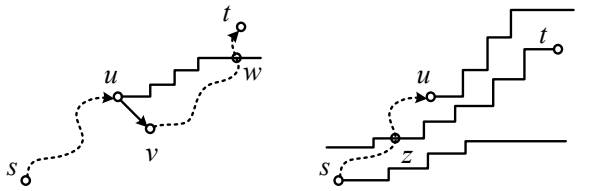
For the first point, we need the following lemma:

Lemma 5. *For two vertices s and t on G_P, if there is no path between them, then $MWL = HPWL$.*

Proof. If MWL>HPWL, then the RU locus of s goes above t (Theorem 1). There must exist an MWL path in the HVG (Lemma 4) which violates the assumption. □

Lemma 6. *For two vertices s and t in G_P, suppose t is above the RU locus of s. If there exists a shortest path between the two vertices, then its length must be HPWL.*

Proof. Suppose the shortest path p is longer than the HPWL. Then it can't be y-monotonic and thus has at least one downgoing edge (by Lemma 1 and Corollary 1). Suppose the source and target of that edge are u and v. We draw the RU locus of u. There are essentially two cases:



(a) u's RU locus goes below t (b) u's RU locus goes above t

Fig. 8. The proof of Lemma 6.

- This locus goes below t (Fig. 8(a)). Then the path (v, t) must intersect with the RU locus at *a certain vertex w* (the intersection cannot happen between the edges because of Lemma 3). Then there is a path $s \to u \to$ RU locus of $u \to w \to t$ on G_P whose length is shorter than p.
- This locus is above t (Fig. 8(b)). Then similarly we can find a path $s \to z \to$ LD locus of $t \to t$ on G_P whose length is shorter than p. □

With Lemma 5, Lemma 6 and Theorem 2, we can see that checking whether the locus goes above or below t is not necessary and the second point is clarified:

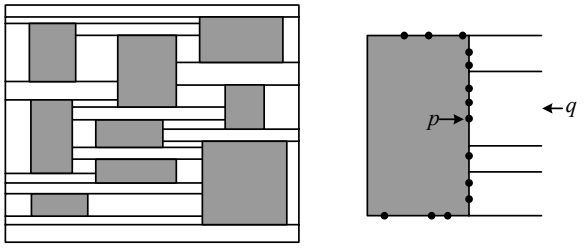
Theorem 3. *The MWL of any two vertices on G_P can be obtained by finding the shortest path on the graph. If the shortest*

path exists, then the MWL is its length. Otherwise, the MWL is the HPWL.

Finally, we deal with the last point – finding the RAB/LAB of each pin. Here we only describe how to find the RAB. Finding the LAB is similar.

If the pin is on the left boundary of a block, then its RAB is the block itself. Otherwise, we obtain its RAB by a two step process: first we build the horizontal stripping that provides channels to link the pins on its left boundary to the block on its right boundary, then we link the pins to their RABs.

The Horizontal Stripping (see Fig. 9) is defined as the *Maximal Horizontal Strips* in the corner stitching structure (Chapter 4.4.5 in [9]). The construction of the horizontal stripping is



(a) The horizontal striping (b) Link pins to strips

Fig. 9. Use horizontal stripping to find the RAB of each pin.

similar to the construction of G_P (see Algo. 2) and it takes $O(M \cdot \log M)$ time.

After the strips are constructed, we link all the pins to their RABs by scanning the pins and strips once. With the pins on each block sorted by their y-coordinates, this process can be done in $O(M+N)$ time (the number of strips is $O(M)$). Sorting all the pins takes $O(M \cdot N \cdot \log N)$ time. But since we need to do it only once before the SA process, this pin-sorting process is excluded from the estimation for one placement configuration.

With all the theorems and descriptions above, we now present the LUT based MWL estimation algorithm (see Algo. 3). In the algorithm, line 1 takes $O(M \cdot \log M + M + N)$ time, line 2 takes $O(M \cdot \log M)$ time. The *for* loop from line 3 to line 10 takes $O(M^2)$ time. The loop from line 11 to line 13 takes $O(N)$ time. The total time complexity of Algo. 3 is therefore $O(N + M^2)$.

Algo. 3 MWL calculation using LUT

```
1:  Get the RAB/LAB of all the pins
2:  Build the HVG of the placement
3:  for each vertex-pair (u, v) do
4:      Find shortest path between u and v
5:      if there is no shortest path then
6:          dist(u, v) =HPWL
7:      else
8:          dist(u, v) =shortest path length
9:      end if
10: end for
11: for each net do
12:     Select the shortest path among the four choices
13: end for
```

The space complexity of the lookup tables is $O(N+M^2)$ because the lookup table of vertex-to-vertex distances consumes $O(M^2)$ space and the lookup table for RAB/LAB of each pin consumes $O(N)$ space.

V. Conclusions and Future Works

In this paper, we revisit the theoretical result in [1, 8] and interpret it in a CAD-friendly presentation. We also make an extension on it so that it is more realistic to VLSI design. As the result, what presented is an algorithm that estimtes the obstacle-avoiding minimum wire length in $O(M^2 + N)$ time for a placement with M blocks and N 2-pin nets. Considering that $M \ll N$ in many practical cases, the implication is significant: the obstacle-avoiding minimum wire length can be estimated in constant time per net, just the same as the HPWL model.

In the paper, we only discuss over the case in which all the blocks are obstacles. It is easy to extend our method to handle the case in which only some of the blocks are opaque, by building the HVG based on those opaque blocks only.

Future works include: 1. Integrating routing congestion into the graph model. 2. Extending this method to handle multi-pin nets. 3. Experimental study of this method. 4. Applying this method to build a fast global router.

References

[1] M. J. Atallah and D. Z. Chen, "Parallel rectilinear shortest paths with rectangular obstacles," *Computational Geometry: Theory and Applications*, Vol. 1, Issue 2, pp. 79–113, 1991.

[2] H. H. Chan, S. N. Adya, and I. L. Markov, "Are floorplan representations important in digital design?" ISPD'05, pp. 129-136, 2005.

[3] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: a new representation for non-slicing floorplans," DAC'00, pp. 458-463, 2000.

[4] C.-K. Cheng, A. B. Kahng, B. Liu, and D. Stroobandt, "Toward better wireload models in the presence of obstacles," ASPDAC'01, pp. 527-532, 2001.

[5] C. Kodama, K. Fujiyoshi, and T. Koga, "A novel encoding method into sequence-pair," ISCAS'04, pp. 329-332, 2004.

[6] J. S.B. Mitchell, "Geometric shortest paths and network optimization," *Handbook of Computational Geometry*, Elsevier Science, pp. 633-702, 2000.

[7] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair," *IEEE TCAD*, Vol.15, No. 12, pp. 1518-1524, 1996.

[8] P. J. de Rezende, D. T. Lee and Y. F. Wu, "Rectilinear shortest paths with rectangular barriers," SCG'85, pp. 204-213, 1985.

[9] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1999.

[10] X. Zhang and Y. Kajitani. "Space-planning: Placement of modules with controlled empty area by single-sequence," ASPDAC'04, pp. 25-30, 2004.