LEMAR: A Novel Length Matching Routing Algorithm for Analog and Mixed Signal Circuits*

Hailong Yao   Yici Cai   Qiang Gao

EDA Lab., Department of Computer Science and Technology
Tsinghua University, Beijing 100084, P.R. China
E-mail: {hailongyao,caiyc}@tsinghua.edu.cn, gaoq04@gmail.com

Abstract— Enabled by the heterogeneous integration in modern System-On-Chips (SOCs), the design automation for analog and mixed signal circuit components in SOCs is attracting increasing interests. Matching constraints for specific analog signals are critical for correct functionalities. This paper presents a novel single-layer detailed routing algorithm with the length matching constraint, called LEMAR. LEMAR features an innovative routing model for partitioning the routing layout for wire detouring, effective detouring patterns according to the geometric shapes of the partitioned tiles, an enhanced A*-search algorithm along with the backtrack technique for finding the routing path, and an iterative rip-up and reroute procedure for finding the feasible routing solution with the matching constraint. Experimental results are promising and show that LEMAR is both effective and efficient.

I. Introduction

As System-On-Chips (SOCs) are getting widely used in real-life applications, the design automation for analog and mixed signal circuit components in SOCs are attracting more and more concerns. However, due to sensitive analog signals and hence complicated analog design constraints, analog design automation has not made the same progress with the digital design counterpart. Therefore, analog design automation has become the bottleneck and brought difficult challenges in the automated SOCs design flow [1, 2].

Matching constraint is a prevalent geometric constraint in analog and mixed signal designs, such as analog circuit amplifier. Mismatches in electrical characteristics (e.g., parasitic capacitance and resistance) between two matching nets will cause offsets and even circuit malfunctions [3]. Therefore, it is very important to enforce the matching constraint on certain analog signals. According to the specific electronic requirements for different analog devices, the matching constraints can be classified into three types: symmetric constraint [7], exact matching constraint [9, 10], and length matching constraint [6]. Symmetric constraint and exact matching constraint both satisfy the length matching constraint, but are stricter with better electrical performance. However, due to the limited routing resource, routing two nets with symmetric or exact matching constraint is difficult. For routing on a single layer, length matching routing can obtain similar electrical performance for the given nets. For this reason, "Equal Length Route" is supported in commercial automatic analog routing tools [11]. Besides, a single-layer length matching routing algorithm can be easily extended to electrical matching due to the linear relationship between parasitic resistance/capacitance and wire length/bends. In this paper, we focus on the single-layer length matching routing problem, which can also be extended to multi-layer routing. However, multi-layer length matching routing is not suggested, as it obtains worse electrical performance due to different wire widths and vias than the exact matching multi-layer routing [9]. Figure 1 shows an example of length matching routing solution for two nets on a single layer.

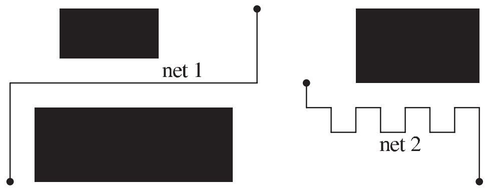


Fig. 1. Two nets with the length matching constraint.

There has been several works on the routing problem with the matching constraint. Carragher et al. present a linear programming based algorithm for computing the optimal Steiner point positions to equalize wire lengths and achieve zero skew [4]. In [5], a Lagrangian relaxation based routing algorithm is proposed to route nets within tight minimum and maximum length bounds. Yan et al. propose a routing algorithm based on Bounded-Sliceline Grid (BSG) to solve length matching problem [6]. The above-mentioned works do not consider routing obstacles in the layout. Besides, the time complexity of the above algorithms is high. Therefore, the above methods may not be applicable for industrial analog designs. In [7], Du et al. propose a detailed routing algorithm with symmetric constraint, which is a stricter constraint than the length matching constraint. Schreiner et al. propose a length-compensation routing algorithm for bus nets [8]. The algorithm adjusts the shorter nets slightly in the local neighborhood, and may not find the feasible routing solution if one net is much longer than the others. Recently, Ozdal et al. propose a dynamic-programming algorithm to plan the global routing topologies for the exact matching routing problem [9]. In [10] a detailed routing algorithm is proposed for routing the given nets with the exact matching constraint. However, the exact matching routing algorithms cannot be easily extended for the length matching constraint, and the time complexity of the algorithms

*This work is supported by the National Natural Science Foundation of China (60976035) and Specialized Research Fund for the Doctoral Program of Higher Education:SRFDP-200800030026.

is high due to the strict exact matching constraint. Therefore, the existing works may not be suitable for fast single-layer length matching routing in modern SOC designs.

In this paper, we present a novel detailed routing algorithm named "LEMAR" for the length matching constraint. Besides obtaining the single-layer length matching routes for the given nets, LEMAR can also detour/snake the route for a given net to any feasible length. This is similar to the Lengthening command "Equal Length Wire" in commercial automatic analog routing tool [11]. Major contributions of this paper are as follows:

- A new routing model is presented to partition the routing area into rectangular routing tiles, which conforms to the detouring requirement for length matching.
- We design different types of the detouring patterns according to the geometric shapes of the tiles. Following these patterns, a net can detour as long as possible in each free routing tile.
- An enhanced A*-search algorithm along with the backtrack technique is presented, which searches for the maximum detouring length during the expanding stage for the shorter net.
- An iterative rip-up and reroute process is adopted if one round of length matching routing does not find a feasible routing solution with good matching results.

The rest of the paper is organized as follows. Section II gives the length matching routing problem formulation and the overall routing flow. In Section III, we introduce the method for partitioning the routing area into rectangular tiles and the different types of detouring patterns within the free tiles. Section IV presents the enhanced A*-search algorithm along with the backtrack technique to obtain a detouring path, and the rip-up and reroute technique. Experimental results are presented and discussed in Section V. Finally, the paper is concluded with future research directions in Section VI.

II. Problem Formulation and Overall Flow

In this paper, we focus on the single-layer length matching routing problem for two nets. For multi-layer routing problem, exact matching is more appropriate for better electrical performance. The single-layer length matching routing problem can be stated as follows.

Given: A single-layer routing area R with a set of routing obstacles O, and two nets n_1 and n_2.

Find: Routing paths p_1 and p_2 for nets n_1 and n_2 to minimize the total wire length.

Subject to: (i) The routing obstacles in O are avoided. (ii) There are no design rule violations. (iii) p_1 and p_2 are of the same length.

In [10], a multi-layer detailed routing algorithm is presented for the exact matching routing constraint, which routes the given nets simultaneously. As mentioned above, exact matching constraint is much stricter than length matching constraint. As a result, the exact matching routing algorithm has high time complexity. For single-layer routing, the length matching constraint is enough for obtaining similar electrical performance as the exact matching. But the length matching routing algorithm runs faster with the relaxed constraints.

Figure 2 shows the overall flow of our length matching routing algorithm LEMAR. First, the routing information is loaded,

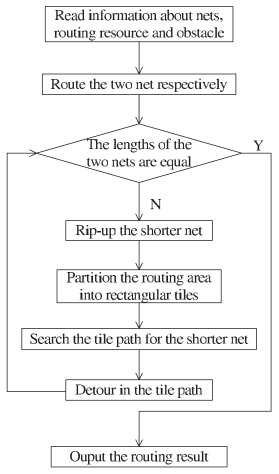

Read information about nets, routing resource and obstacle
Route the two net respectively
The lengths of the two nets are equal
Y
N
Rip-up the shorter net
Partition the routing area into rectangular tiles
Search the tile path for the shorter net
Detour in the tile path
Ouput the routing result

Fig. 2. Flow of LEMAR.

including the routing area, the design rules, the routing obstacles, and the nets to be routed. Second, the given two nets are routed sequentially without considering the length matching constraint. Here any classic routing algorithm can be used for the initial routing of the two nets, provided that the minimum routing wire length can be obtained. Third, the wire lengths of the two nets are compared. If the two nets are of the same length, the length matching routing is finished. Otherwise, a detouring iteration is started. In the detouring stage, the shorted net is ripped up and rerouted by the presented length-matching routing algorithm. If the rerouting process outputs a routing solution with the same length as the longer net, then the detouring stage is finished. Otherwise, a routing solution with a longer wire length than the original longer path will be used, if there exists one. Then another round of detouring iteration is started with the rip-up and reroute of the original longer path. The detouring iteration will be finished when valid routing solutions of the same length are obtained or no longer paths for the two nets can be found. For analog and mixed signal designs, the routing resource is more abundant than in the digital counterparts. Therefore, the presented algorithm almost always guarantee a feasible solution for a given pair of nets.

III. Tile-based Detouring

A. Partitioning of the Routing Area

In [12], an implicit connection graph model is presented, which implicitly partitions the routing area into non-uniform routing grids. To satisfy the design rule, obstacles are first expanded by $WS_{min} + WW_{min}/2$, where WS_{min} and WW_{min} are minimum wire spacing and minimum wire width, respectly (see Figure 3 (a)). Then, the routing area is partitioned into non-uniform grids by the expanded obstacle boundaries along with the source and target grids. Nets are routed on the non-uniform routing grids. Figure 4 (a) shows the partitioned routing area using the implicit connection graph model.

By contrast, in our length matching routing algorithm, nets need to be detoured in the unoccupied routing area. So the above implicit connection graph model is not suitable in this case. We present a new model for partitioning the routing area as shown in Figure 3 (b). As shown in Figure 3 (b), we expand obstacles by $WS_{min}/2$, where the expanded obstacles are called *obstacle tiles*. Besides, we represent the source and target pins as rectangular tiles called *pin tiles*. Then, the routing area is partitioned by the boundaries of all the pin tiles and obstacle tiles. In the partitioned layout, all rectangular tiles other than pin/obstacle tiles are called *free tiles*. When routing a net, a free tile can be used for making detours for longer wire length. The constraint for routing paths in the free tiles is that routing paths cannot be within distance $WS_{min}/2$ from the tile boundaries. Figure 4 (b) shows a partitioned routing area using our method and a possible detouring result of a given net. From Figure 4, our tile partitioning method can ensure the minimum space design rule be satisfied between the routing path and the obstacles.

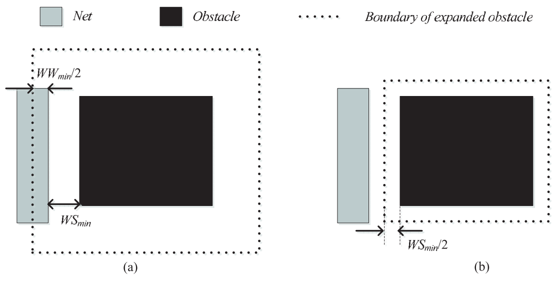


Fig. 3. Different methods for obstacle expansion.

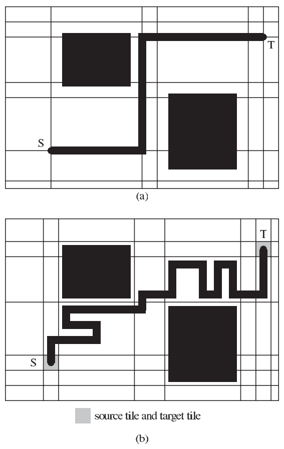


Fig. 4. Partitioned routing area and its detouring result.

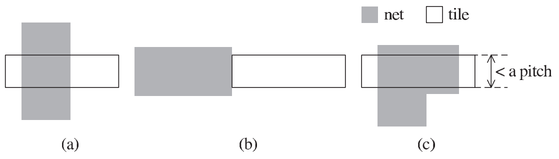


Fig. 5. Net expansion in a tiny tile.

Note that there is a special type of space tiles called *tiny tiles*, whose width or height is smaller than a *pitch*[1]. For the tiny tile, the expanding direction of a net is limited. As shown in Figure 5, assume width of the tile is smaller than a *pitch*. Net can only expand in vertical direction across the tile as shown in Figure 5 (a). It cannot expand in the horizontal direction (see Figure 5 (b)) nor make a turn inside the tile (Figure 5 (c)), which would violate the design rule. During the partitioning of the routing area, a heuristic algorithm is adopted to merge the tiny tiles into the adjacent tiles to avoid the above problem.

B. Estimation of the Detouring Length

Nets can make detours in the free tiles, and the detouring lengths are related to the area of the tiles. As shown in Figure 6 (a), a net detours in a free tile. We can partition the tile into regions along the dotted lines, i.e., the 5 regions marked from 1 to 5. Each region contains a part of the net. As shown in Figure 6 (b), the partitioned 5 regions can be recombined into a new rectangular tile, whose width and height are a *pitch* and the detouring length of the net, respectively. The black region does not appear in the new tiles in Figure 6 (b), and this type of region is called *wasted region*. Assume that the area of the original tile is S_a, and the area of the wasted region is ΔS, the detouring length L_d can be calculated as follows:

$$L_d = (S_a - \Delta S)/pitch \tag{1}$$

To increase the detouring length, the routing algorithm should choose the tiles in the routing path with large area and should minimize the area of wasted regions as much as possible.

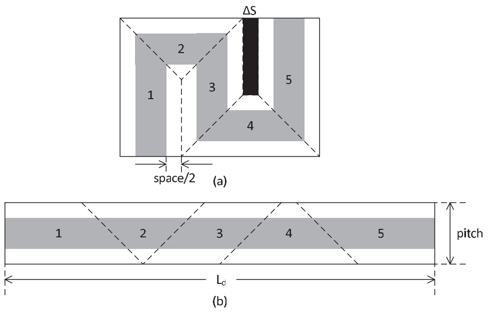


Fig. 6. The Relation between area of tile and detouring length.

C. Transformation of Different Tiles

According to the entrance and exit points within the tile, free tiles can be classified into two categories: (1) entrance and exit are on the diagonal corners, and (2) entrance and exit are on the same edge. The two different cases can be transformed into a single type of tile by some simple operations. In our routing algorithm, all the free tiles are transformed into a single case, where entrance and exit are on the bottom left and top right corners of tile, respectively. Figure 7 shows the methods for the transforming operation.

- If entrance and exit are on bottom left and top right corners of a tile (Figure 7 (a)), no transforming operations are needed.
- If entrance and exit are on bottom right and top left corners of a tile (Figure 7 (b)), after doing *symmetric mapping* operation according to the dotted line, the tile can be transformed into the same type as Figure 7 (a).
- If entrance and exit are on the top edge of the tile (Figure 7 (c)), the tile can be divided into two tiles t_1 and t_2: t_1's width is a *pitch*, and a net only expands from top to bottom, and t_2 is of the same type as Figure 7 (a).
- If entrance and exit are on the right edge of the tile (Figure 7 (d)), after doing *flip* operation according to the 45°-diagonal dotted line, the tile can be transformed into the same type as Figure 7 (c). Then it can be divided into two tiles through the same operation as in Figure 7 (c).
- If entrance and exit are on other edges of the tile, similar operations can be performed.

Based on the above analysis of different cases for the entrance and exit points, we only need to consider a single case for computing the detouring path in a free tile, i.e., when the entrance and exit are on the bottom left and top right corners of the free tile, respectively. Next, we present the detouring patterns within such free tiles.

[1] $pitch = WS_{min} + WW_{min}$.

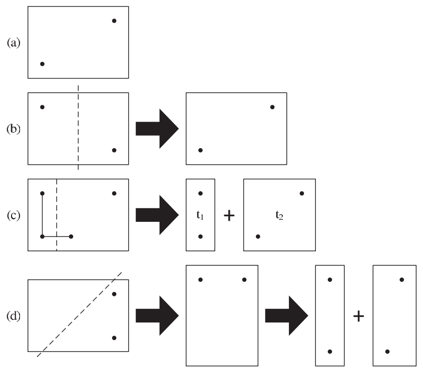


Fig. 7. Transforming operations.

D. Detouring Patterns

Detouring patterns are quite important for increasing the wire length. To obtain maximum detouring length, we adopt two detouring patterns according to the geometric shape of a free tile, i.e., *narrow pattern* and *standard pattern*. If the width or height of a free tile is smaller than $4 \times pitch$, the narrow pattern is used. As shown in Figure 8, if the height is smaller, the routing path detours in the vertical direction. Otherwise, it detours in the horizontal direction.

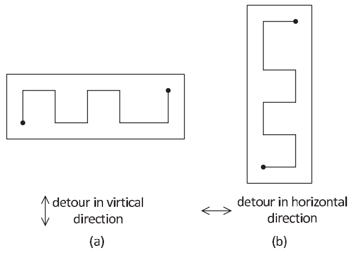


Fig. 8. Narrow pattern.

If both the height and width of a free tile are greater than $4 \times pitch$, the standard pattern is used. In standard pattern, the routing path first detours in the horizontal direction. When the height of the remaining free space is less than $4 \times pitch$, i.e., pure horizontal detour cannot occupy the whole tile, it starts to detour in the vertical direction with the narrow pattern. By detouring in two directions, the wasted region is minimized for larger detouring length. Figure 9 shows an example of the standard pattern vs. a pure vertical detour. In the figure, the black regions are wasted. It is obvious that ΔS_a is smaller than ΔS_b. According to Equation 1, the detouring length of standard pattern L_d^a is larger than L_d^b.

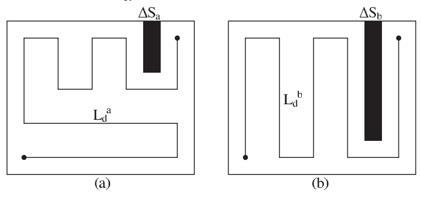


Fig. 9. Comparison between standard pattern and narrow pattern.

According to the size of the free tile and the corresponding detouring pattern, the maximum detour wire length in the free tile can be computed efficiently. The maximum detour wire length is used during the searching of the tile path in the A* algorithm.

IV. Tile Path Searching and Detouring

When the two nets are routed using certain routing algorithms, the shorter net is ripped up and rerouted using A* algorithm to obtain a routing solution with a longer wire length. First, the routing area is partitioned into the routing tiles. Then, A* algorithm is adopted to find a path of free tiles for the shorter net. Inside these free tiles, the shorter net will make detours to increase its length, such that the length matching constraint may be satisfied.

A. A*-Search Algorithm for Tile Path

In the A* search algorithm, rectangular tiles are abstracted as points. Thus, we can get a virtual 2D grid graph, where each grid point represents a tile. Table I gives the variables used in the algorithm, and Algorithm 1 gives the pseudo code of our searching algorithm.

TABLE I
Variables used in A* algorithm for searching tile path.

Variable	Definition
L_{obj}	the desired length
t_{cur}	the current tile
t_{new}	the new expanding tile
t_s	the source tile
t_t	the target tile
L_{min}	the minimum length from t_s to t_{cur}
L_{max}	the maximum length from t_s to t_{cur}
dir	the direction of tile path before last turn
$cost_G$	the total cost from t_s to t_{cur} in the virtual grids
Δg	the cost from t_{cur} to t_{new}
$cost_H$	Manhattan Distance from t_{new} to t_t in the virtual grids
$cost_F$	the sum of $cost_G$ and $cost_H$

Algorithm 1 A*-search-based algorithm for tile path.

```
Input: $t_s$, $t_t$, $route\_area$, $L_{obj}$
Output: A tile path
 1: $openlist.push\_back(t_s)$
 2: while $openlist$ is not empty do
 3:     $t_{cur}$ = the first element of $openlist$
 4:     Remove $t_{cur}$ from $openlist$
 5:     if $t_{cur} == t_t$ then
 6:         if $t_{cur}.L_{max} + pitch/2 \geq L_{obj}$ then
 7:             if $t_{cur}.L_{min} + pitch/2 \leq L_{obj}$ then
 8:                 Find the paths
 9:                 return true
10:             else
11:                 return false
12:             end if
13:         end if
14:         set $t_{cur}$ free
15:     else
16:         Expand new tiles from $t_{cur}$
17:         Obtain $cost_H$, $cost_G$, $cost_F$, $L_{min}$, $L_{max}$, $dir$ for new tiles
18:         Insert new tiles into $openlist$ and keep $openlist$ sorted according to $cost_F$ and
            $L_{max}$
19:     end if
20: end while
21: return false
```

In Step 18, tiles in *openlist* are sorted according to $cost_F$ in non-descending order. The tiles with the same $cost_F$ are sorted according to L_{max} in non-ascending order. Thus, the tile which has larger wire length from t_s to it would be expanded preferentially. This strategy will reduce the searching space and obtain the tile path with large detouring length.

The backtrack strategy [13] is used in the tile searching process. If t_{cur} is t_t, but the maximum length from t_s to t_t of the current tile path is smaller than L_{obj}, our algorithm would set t_{cur} free (Step 14 in Algorithm 1), just as it has not been expanded. Then t_{cur} would be equal to a neighbor tile of t_t and expand to t_t again. This would force tile path to detour around the t_t to increase the detouring length. Experimental results in Section V show that this backtrack strategy is quite effective.

B. Minimum Length and Maximum Length

It is very important to estimate L_{min} and L_{max} precisely. The computation of these two variables for t_{new} is affected by the following cases: (1) whether there is a turn at t_{cur}, and (2) the direction of tile path before the last turn (dir in Table I).

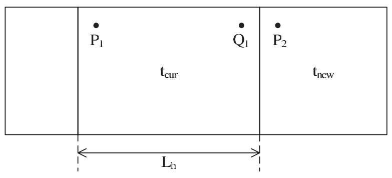


Fig. 10. Calculation of L_{min} and L_{max} if turn does not happen at t_{cur}.

If there is not a turn at t_{cur}, the relative position of source point in t_{new} is the same as t_{cur}. As shown in Figure 10, t_{cur}'s source point P_1 is on the top left corner of t_{cur}, and t_{new}'s source point P_2 is also on the top left corner of t_{new}. So t_{cur}'s target point Q_2 is on the top right corner of t_{cur}. Then, we can obtain L_{min} and L_{max} of t_{new}[2]:

$$\begin{aligned} t_{new}.L_{min} &= t_{cur}.L_{min} + t_{cur}.L_h \\ t_{new}.L_{max} &= t_{cur}.L_{max} + calMaxLen(t_{new}) \end{aligned} \quad (2)$$

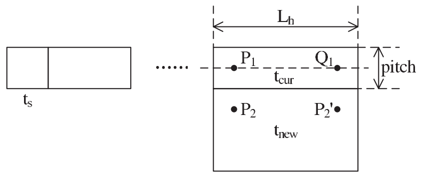


Fig. 11. Calculation of L_{min} and L_{max} if the first turn of tile is at t_{cur}.

If the first turn of tile path is at t_{cur}, according to the method for partitioning routing area, the width or height of t_{cur} is $pitch$. As shown in Figure 11, t_{cur}'s source point P_1 is near to left edge of t_{cur}. When calculating L_{min} of t_{new}, t_{new}'s source point P_2 is at top left corner of t_{new}, and t_{cur}'s target point is at the same position with P_1. When calculating L_{max} of t_{new}, t_{new}'s source point P_2' is at top right corner of t_{new}, and t_{cur}'s target point Q_1 is near to right edge of t_{cur}. Then, we can obtain L_{min} and L_{max} of t_{new}:

$$\begin{aligned} t_{new}.L_{min} &= t_{cur}.L_{min} + pitch/2 + pitch/2 \\ &= t_{cur}.L_{min} + pitch \\ t_{new}.L_{max} &= t_{cur}.L_{max} + t_{cur}.L_h - pitch/2 + pitch/2 \\ &= t_{cur}.L_{max} + t_{cur}.L_h \end{aligned}$$

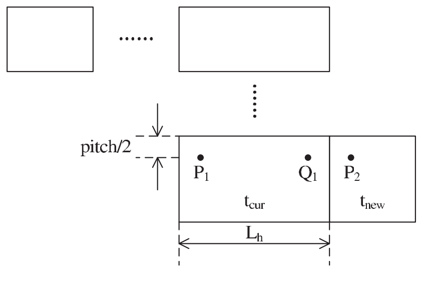


Fig. 12. Compute L_{min} and L_{max}: path turns at t_{cur} and $t_{cur}.dir$ is not $NONE$.

If tile path turns at t_{cur} and the direction before last turn ($t_{cur}.dir$) is not $NONE$, according to the previous situation, it is clear that t_{cur}'s source point is at the corner near to t_{cur}'s father tile and near to the direction before last turn. As shown in Figure 12, $t_{cur}.dir$ is $Left$, so t_{cur}'s source point P_1 is at top left corner of t_{cur}. When calculating L_{min} and L_{max} of t_{new}, t_{new}'s source point P_2 is at top left corner of t_{new}, too. Then we can determine that t_{cur}'s target point is at top right corner of t_{cur}. At last, we can obtain L_{min} and L_{max} of t_{new}:

$$\begin{aligned} t_{new}.L_{min} &= t_{cur}.L_{min} + t_{cur}.L_h \\ t_{new}.L_{max} &= t_{cur}.L_{max} + calMaxLen(t_{new}) \end{aligned} \quad (3)$$

[2]$calMaxLen()$: function for computing detouring length in a tile using the two patterns.

If tile path turns at t_{cur} in other directions, L_{min} and L_{max} of t_{new} can be calculated in a similar way.

C. Detouring after Searching

When the tile path is found, we start to detour within each tile. The routing path is detoured in the tiles one by one from target tile t_t to the source tile t_s. Since the minimum length from t_s to each tile (L_{min}) has been obtained during the tile expanding process, we use a greedy algorithm to compute the desired wire length L_w in each tile according to L_{min} and the desired wire length of the whole net L_{obj}. Assume the obtained wire length from t_t to t_{cur} is L_M, L_w can be calculated as follows:

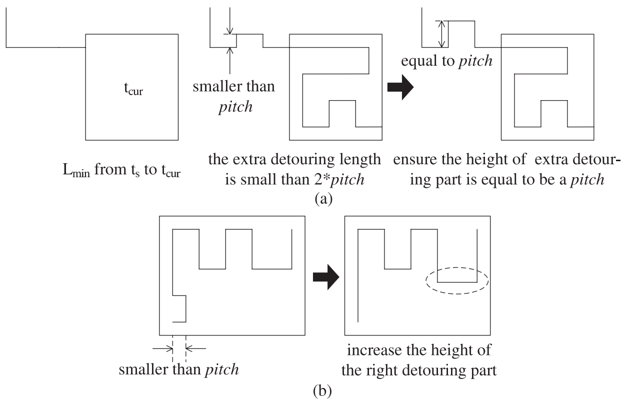
$$L_w = L_{obj} - L_M - L_{min} \quad (4)$$

The maximum detouring length L^t_{max} of each tile can be calculated using $calMaxLen()$ function. According to the relation between $wLen$ and L^t_{max}, we can obtain the detouring length L_d of each tile:

1. If $L_w \le L^t_{max}$, L_d is equal to L_w.
2. If $L_w > L^t_{max}$, L_d is equal to L^t_{max}.

When doing detour in one tile, if L_d is smaller than L^t_{max}, the height of detouring parts of net should be reduced to decrease the wire length. However, there are two special cases:

- If $L_w > L^t_{max}$ but $L_w - L^t_{max} < 2 \times pitch$, the extra detouring length in remaining tiles is smaller than $2 \times pitch$, which would violate the design rule. So the detouring length in current tile should be reduced to keep the extra detouring length equal to $2 \times pitch$ (see Figure 13 (a)).
- When reducing the height of detouring parts of net, if the height of some detouring part is smaller than a $pitch$, the design rule will also be violated. Then, we will adjust the height of other detouring parts to make the height of this detouring part to be 0 or a $pitch$ (see Figure 13 (b)).



tcur
smaller than pitch
equal to pitch
Lmin from ts to tcur
the extra detouring length is small than 2*pitch
ensure the height of extra detouring part is equal to be a pitch
(a)
smaller than pitch
increase the height of the right detouring part
(b)

Fig. 13. Adjust the height of some detouring parts.

V. Experimental Results

The length matching routing algorithm is implemented in C++ and run on Linux server with Intel(R) Xeon(TM) 2.40GHz CPU. For comparison, we implement a routing algorithm, which only searches tile path without backtrack strategies. When t_t is the first element of $openlist$, if the maximum detouring length is smaller than the desired length, the contrast algorithm set the maximum detouring length as the final length of the shorter net.

The algorithms are tested on artificially generated benchmarks with obstacles and given nets with the length matching constraint. Table II shows the experimental results of the length

TABLE II
EXPERIMENTAL RESULTS OF LEMAR.

benchmark	routing area (nm^2)	#obstacles	% obstacles/ routing area	len. of net 1 (*nm*)	original len. of net 2 (*nm*)	len of net 2 after detouring (*nm*)	runtime (s)
T1	57800×27900	3	11%	64900	38000	64900	0.01
T2	40200×23300	46	47%	36800	24500	36800	0.01
T3	29850×24700	11	41%	40500	21500	40500	0.01
T4	210200×210300	609	44%	254500	170900	254500	3.72
T5	237700×236400	635	36%	236000	155600	236000	2.52

matching algorithm, and Table III shows the results of the contrast algorithm. Since the two algorithms use the same benchmarks, Table III only gives the information about length and runtime. From Table II, the length matching routing algorithm could do detouring for *net*2 to keep the length of *net*2 equal to *net*1. From Table III, without the backtrack techniques, the contrast algorithm failed to find length matching solutions for $T2$ and $T3$.

TABLE III
EXPERIMENTAL RESULTS OF THE CONTRAST ALGORITHM.

bench-mark	len. of net 1 (*nm*)	original len. of net 2 (*nm*)	len. of net 2 after detouring (*nm*)	time (s)
T1	64900	38000	64900	0.01
T2	36800	24500	35300	0.01
T3	40500	21500	23400	0.01
T4	254500	170900	254500	3.67
T5	236000	155600	236000	2.50

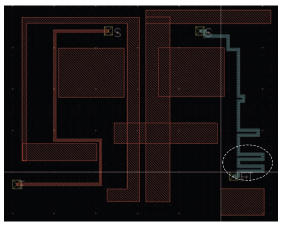


Fig. 14. Routing result of $T3$.

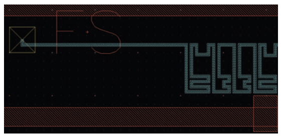


Fig. 15. Routing result of $T4$.

Figure 14 and Figure 15 gives the routing result of two benchmarks obtained from the length matching routing algorithm. Figure 16 shows the routing result of $T3$ from the contrast algorithm. In Figure 14, the region inside dotted line represents the tile path detours around the target tile, and in Figure 16, the tile path does not detour since the contrast algorithm does not use backtrack and iterative strategies. From Table III, we can find the contrast algorithm does not find the correct tile path for $T2$ and $T3$, and the length matching constraint does not satisfied in these two benchmarks.

From the experiments, we observe that if there are too many obstacles as in $T4$ and $T5$, many tiles are very small (i.e., tiny tiles) due to the partitioning along the boundaries of expanded obstacles, which makes it difficult to find suitable tile path. Tiny tiles are not suitable for either standard pattern or narrow pattern. However, the free tiny tiles can be combined into a larger tile in which nets can detour. Therefore, for those extreme cases with many obstacles, an efficient tile partitioning and merging method is needed. Fortunately, in practical analog designs the routing resources are typically abundant without so many obstacles. Therefore, our algorithm sufices in most

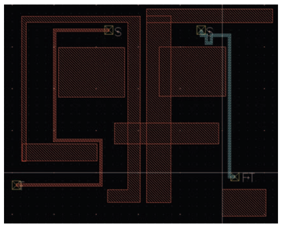


Fig. 16. Routing result of $T3$ from the contrast algorithm.

cases. Besides, from the flow of algorithm, it is easy to understand that our algorithm can be extended to route a single net with a pre-specified length, which can find its application for special-purpose routing.

VI. CONCLUSIONS

In analog and mixed signal circuits, the length matching constraint should be considered to guarantee better electrical properties. In this paper, we present a novel detailed routing algorithm for length matching problem, which guarantees two nets with the same length. The algorithm is based on tile searching and detouring in free tiles, which is also suitable for finding a path with a given range of length. Future work includes testing the presented algorithm on more industrial benchmarks, as well as extending the algorithm for matching more than two nets simultaneously and with evenly distributed detouring wires to avoid local congestion.

REFERENCES

[1] N. Weste and D. Harris, “CMOS VLSI design: a circuits and systems perspective”, Pearson Education Asia Limited and China Machine Press, 2005.

[2] R. A. Rutenbar, “Design automation for analog: the next generation of tool challenges”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2006, pp 458-460.

[3] R. A. Rutenbar and J. M. Cohn, “Layout tools for analog ICs and mixed-signal SoCs: a survey”, *Proc. International Symposium on Physical Design*, 2000, pp 76-83.

[4] R. J. Carragher, C. K. Cheng, and M. Fujita, “An efficient algorithm for the net matching problem”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1993, pp 640-644.

[5] M. M. Ozdal and M. D. F. Wong, “A length-matching routing algorithm for high-performance printed circuit boards”, *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 12, 2006, pp 2784-2794.

[6] T. Yan and M. D. F, “BSG-route: A length-matching router for general topology”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp 499-505.

[7] C. Du, Y. Cai, X. Hong, and Q. Zhou, “A shortest-path-search algorithm with symmetric constraints for analog circuit routing”, *Proc. International Conference on ASIC*, 2005, pp 844-847.

[8] L. Schreiner, M. Olbrich, E. Barke, and V. M. zu Bexten, “Routing of analog busses with parasitic symmetry”, *Proc. International Symposium on Physical Design*, 2005, pp 14-19.

[9] M. M. Ozdal and R. F. Hentschke, “Exact route matching algorithms for analog and mixed signal integrated circuit”, *Proc. International Conference on Computer-Aided Design*, 2009, pp 231-238.

[10] Q. Gao, H. Yao, Q. Zhou and Y. Cai, “A novel detailed routing algorithm with exact matching constraint for analog and mixed signal circuits”, *Proc. International Symposium on Quality Electronic Design*, 2011.

[11] Laker User Manual, SpringSoft Inc. http://www.springsoft.com

[12] J. Cong, J. Fang, and K.-Y. Khoo, “DUNE: A multi-layer gridless routing system with wire planning”, *Proc. International Symposium on Physical Design*, 2000, pp 12-18.

[13] J. Bitner and E. Reingold, “Backtrack programming techniques”, *Communications of the ACM*, 1975, 18(11), pp 651-656.