Exact Route Matching Algorithms for Analog and Mixed Signal Integrated Circuits

Muhammet Mustafa Ozdal
Intel Corporation
Hillsboro, OR 97124
mustafa.ozdal@intel.com

Renato Fernandes Hentschke
Intel Corporation
Hillsboro, OR 97124
renato.f.hentschke@intel.com

ABSTRACT

As SOC designs are getting more popular, the importance of design automation for analog and mixed-signal ICs is increasing. In this paper, we study the problem of exact route matching, which is an important physical design constraint commonly imposed on specific analog signals for the purpose of correct analog functionality. For this, we first propose a mathematical formulation that models the route matching problem exactly. Based on this formulation, we derive important theoretical conclusions, and propose dynamic-programming algorithms to solve the problem. We also discuss how to use heuristic search techniques to enable faster computations. Our experimental results show the effectiveness of our algorithms.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits**]: Design Aids

General Terms

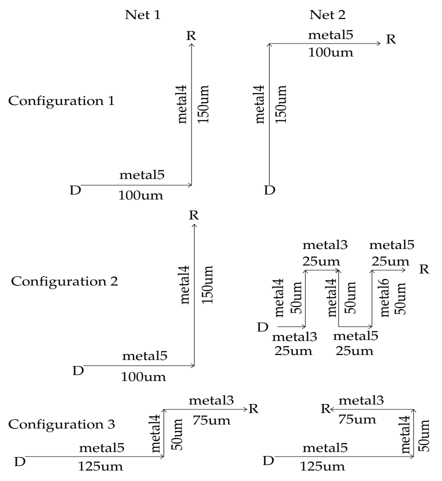
Algorithms, Design

Keywords

Analog and mixed-signal design, routing, length matching

1. INTRODUCTION

As system-on-chips (SOCs) are getting increasingly popular, the need for heterogeneous integration is becoming more apparent in modern VLSI designs. A typical SOC design today has different components with analog and digital functionalities. Digital design automation has been studied extensively in the past few decades, and matured solutions exist for most of the digital physical design problems. However, the analog design automation has not kept in pace with its digital counterpart, mainly because: 1) the analog design process is much more complicated and error-prone, and 2) the analog design sizes used to be small enough for manual design. However, with today's increasing circuit sizes, more complicated circuit functionalities, and the time-to-market pressures, analog parts can easily become the bottleneck in the overall SOC design process. Improvements in the analog physical design algorithms can increase the design productivity significantly by enabling automation of manual tasks that are complex, time-consuming, and error-prone. Hence, there has been renewed interest in analog design automation recently [4, 7, 15].



	Rise delay Net 1	Rise delay Net 2	Rise delay Mismatch
Configuration 1	1.44 ps	2.06 ps	0.62 ps
Configuration 2	1.44 ps	2.19 ps	0.75 ps
Configuration 3	1.34 ps	1.34 ps	0 ps

Figure 1: Simulation results of different routing configurations, all of which have $250\mu m$ total length.

Matching constraints are commonly imposed on specific devices and interconnects in analog and mixed-signal designs. While matching constraints are imposed on digital designs mainly for skew and performance considerations, there are more inherent reasons for the matching constraints in the analog designs [5]. One reason for this is the fact that some circuit functionalities depend directly on matching (such as pipelined analog/digital converters). Furthermore, common-mode rejection and supply noise rejection characteristics are impacted by how good the devices and interconnects are matched [5]. Hence, it is an important requirement for analog circuits to satisfy the matching constraints, and deviations in the signal waveforms can lead to incorrect functionality.

Typically, the matching requirements for nets have been imposed as *length matching* constraints in the literature, especially for board level routing. However, matching the wirelengths exactly does not guarantee that the interconnects have identical electrical properties. The routing topologies, layer assignment, via locations, etc. are all factors that can have impact on the electrical characteristics of the interconnects.

For IC-level routing, each metal layer is defined to have a horizontal or vertical orientation, and the routes need to switch between layers accordingly. For modern IC technologies, different layers may have different characteristics. As an example, for a *65nm* Intel logic technology, the metal layer pitches were reported to be *220nm, 280nm, 330nm, 480nm, 720nm,* and *1080nm* for metal3 to metal8, respectively [1]. It is common in the industry to have lower metal layers having

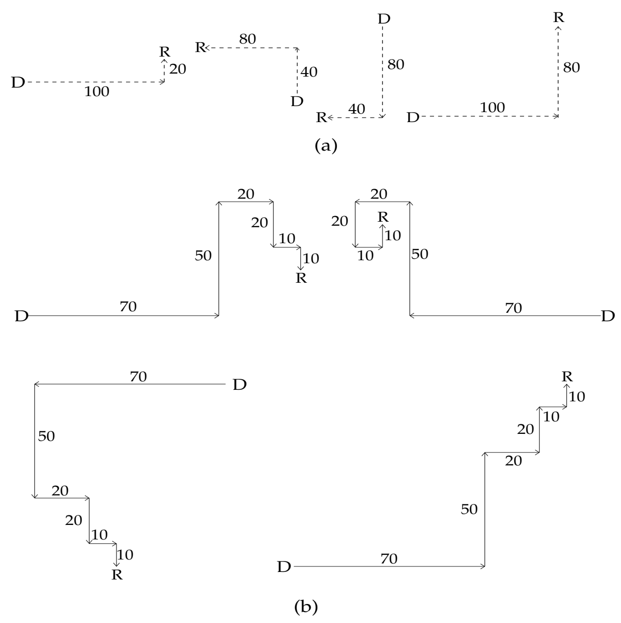


Figure 2: (a) A sample problem with 4 nets to be matched. (b) Exactly matched routing solutions, where each segment has identical length for all nets.

finer pitches (and hence higher resistivities) than the upper layers. So, a wire segment on an upper layer may have lower resistance than another wire segment on a lower layer. Hence, matching the lengths of the routes is not enough to guarantee identical signal waveforms.

We have performed some Spice simulations to demonstrate the fact that length matching is not always enough to have identical interconnect characteristics. In these simulations, metal5 and metal6 are set to have the same wide pitch, while the lower layers are set to have the same narrow pitch. Further details of our experimental setup are given in Section 5. In Figure 1, we have considered a number of routes all of which have total wirelengths equal to $250\mu m$. The driver and receiver terminals (denoted as D and R in the figure) are assumed to be on metal1, and vias are inserted accordingly between layers (though not shown here for clarity of the figure).

In the first configuration of Figure 1, *net1* and *net2* have identical lengths on metal4 and metal5. However, the delay mismatch between them is as high as 30%. The main reason is that the first segment connected to the driver has higher resistance for *net2* compared to *net1*, which leads to a larger delay value. Hence, when the length matching is done without considering routing topologies and layer assignment, the interconnect characteristics do not necessarily match each other. In the second configuration of Figure 1, snaking is done for *net2* similar to [9] to match the lengths. Again, the delays of net1 and net2 fail to be matched mainly because of different layer and via usage.

In the third configuration of Figure 1, routes are matched *exactly*. For both nets, the first segment connected to the driver is a $125\mu m$ metal5 segment, followed by a $50\mu m$ metal4 segment, and followed by a $75\mu m$ metal3 segment. Since the routes are exactly the same for *net1* and *net2*, we can say that they have electrically equivalent routes.

Furthermore, exact route matching makes sure that the interconnect properties change consistently in the presence of process variations. For example, if the waveforms are matched for a particular (Vdd, temperature) corner, they will still be matched for different corners, because the routes are exactly identical.

In this paper, we focus on the problem of exact route matching for a given set of 2 terminal nets. The number of nets that need to be matched with each other is usually small (typically less than 5). However, even for small number of nets, this can be a difficult problem. Figure 2(a) illustrates an example problem where routes of the given 4 nets need to be matched exactly. Figure 2(b) shows the solution where all 4 nets have exactly the same routing solutions. For example, a horizontal segment of length $70\mu m$ is connected to all drivers, followed by a vertical segment of length $50\mu m$, etc. Layer assignment is not shown here for clarity of the figure, but layers need to be identical, too. Furthermore, while determining these routes, we also need to consider the routing costs, and we need to choose the routes with minimum total cost for all nets.

The problem of length matching has been studied in the literature especially for board level routing. In [9], a general purpose Lagrangian relaxation (LR)-based algorithm is proposed. In our experiments, we compare our algorithms with the LR-based routing, and demonstrate that [9] is not sufficient for exact delay matching. In [8] and [10], length matching algorithms are proposed for the cases where all terminals align with each other, which cannot be generalized to solve more general problems. More recently, Yan and Wong [18] proposed bounded sliceline grid (BSG)-based length matching algorithms for single-layer routing. However, these algorithms are not applicable to IC-level routing, where each layer has either horizontal or vertical orientation. In another recent work, Lin et al. [4] proposed a matching based placement and routing system for analog designs. Although they propose a pattern-based matching algorithm for transistor-level routing, they do not discuss the details of how to compute matching routes between different devices, which can be in different locations of the designs.

In the rest of the paper, we first formulate the problem formally in Section 2. Then, we provide a theoretical study of the exact route matching problem in Section 3, and we derive important theoretical conclusions about this problem. Based on the models proposed in this section, we propose dynamic-programming and A*-based algorithms in Section 4 to solve the route matching problem. Our experiments in Section 5 demonstrate the effectiveness of these algorithms. Due to page limitations, complete proofs of the lemmas and theorems will be omitted in this paper.

2. PROBLEM FORMULATION

We can formulate the IC-level exact route matching problem as follows. Let $\mathcal{D}$ be the given design with alternating horizontal and vertical layers and a global routing grid as defined in the public benchmarks [6]. Here, each grid edge is defined to have a predefined cost, which may be based on overflow as in [6]. In the rest of the notations, a net is assumed to have one driver terminal T_D and one receiver terminal T_R with coordinates (x_D, y_D) and (x_R, y_R), respectively. For simplicity of presentation, all terminals are assumed to be in the lowest metal layer, as in [6].

A horizontal (vertical) routing segment s_i is defined to be a directional wire from point (x_1, y) to (x_2, y) (from point (x, y_1) to (x, y_2)). Each routing segment is defined to be maximal in the 2-D space (e.g. a horizontal segment cannot be adjacent to another horizontal segment). Furthermore, let z_i denote the layer assignment for segment s_i, where z_i can correspond to one or more layers (in case parts of the s_i are assigned to different layers). The route R_n of net n is defined to be an ordered sequence of alternating horizontal and vertical segments: $R_n : [s_1, ...s_i, ...s_k]$ and layer assignment for each segment. For such R_n, it must be the case that each s_i and s_{i+1} $(1 \leq i < k)$ have opposite orientations (horizontal or vertical), and the end-point of s_i must have the same (x, y) coordinate as the first point of s_{i+1}. Each transition from s_i to s_{i+1} is denoted as *bend*, and the bend count of R_n with k segments is defined to be equal to $k-1$. Similarly, the number of vias can be computed by summing up the layer transitions at the terminals, at the bends, and possibly within segments (if a segment is assigned to more than one layers). The *cost* of segment s_i is defined as the sum of the costs of the grid edges it is passing through.

Definition 2.1. *Given a set of nets $\mathcal{N}$, where each net $n \in \mathcal{N}$ has route $R_n = [s_1^n, ..., s_i^n, ..., s_{k_n}^n]$ with k_n segments and layer assignment $Z_n = [z_1^n, ..., x_i^n, ..., z_{k_n}^n]$, the routes of all the nets in $\mathcal{N}$ are defined to be* exactly matching *if and only if:*

$\mathcal{N}$	the set of input nets
N	number of nets
t_D^n	driver terminal of net n
t_R^n	receiver terminal of net n
R_n	route of net n
D_H	vector of horizontal distances between terminals (Defn 3.1)
D_V	vector of vertical distances between terminals (Defn 3.1)
ud_h	number of unique entries in D_H
ud_v	number of unique entries in D_V
L_H	vector of horizontal segment lengths (Defn 3.2)
L_V	vector of vertical segment lengths (Defn 3.2)
k_h	number of entries in L_H
k_v	number of entries in L_V
C_H	configuration matrix for horizontal segments (Defn 3.4)
C_V	configuration matrix for vertical segments (Defn 3.4)
C_{mH}	matching configuration matrix for horizontal segments (Defn 3.5)
C_{mV}	matching configuration matrix for vertical segments (Defn 3.5)

Figure 3: Common notations used in the paper

1. *The segment counts of all routes are equal, i.e. $k_n = k_{n+1} = k$, for each $1 \leq n < N$.*
2. *The lengths of the respective routing segments are equal*[1]*, i.e. $|s_i^n| = |s_i^{n+1}|$, for each $1 \leq i \leq k$ and $1 \leq n < N$.*
3. *The layer assignment of the respective routing segments are identical, i.e. $z_i^n = z_{i+1}^n$ for each $1 \leq i \leq k$ and $1 \leq n < N$.*

Definition 2.2. *We can formulate the basic problem in this paper as follows: Given a set of nets $\mathcal{N}$, compute a route $R_n = [s_1^n, ...s_k^n]$ for each net $n \in \mathcal{N}$ such that all routes* exactly match *each other (as defined in Definition 2.1), and the following objectives are realized in lexicographic order:*

1. *Minimize total wirelength $\sum_{n=1}^{N}\sum_{i=1}^{k}|s_i^n|$*
2. *Minimize total segment cost $\sum_{n=1}^{N}\sum_{i=1}^{k} cost(s_i^n)$*
3. *Minimize number of vias $\sum_{n=1}^{N} \#vias(R_n)$*

The algorithms we propose in this paper will be based on bounded-bend pattern routing, which already restricts the maximum number of vias that can be in the routing solutions. Hence, the cost function above puts more emphasis on segment cost minimization. It is well known that pattern routing is effective especially for critical nets because of its predictability [3] in addition to the efficiency of computations [11, 12]. Furthermore, it is typically undesirable to increase the wirelengths of the critical nets to reduce their costs.

By routing a set of nets in an identical manner, we make sure that the signal waveforms arriving at the receiver terminals match each other very closely, which is an important requirement for analog routing. Note that, in this paper, we do not focus on crosstalk issues. These crosstalk issues can be avoided by shielding the most critical nets by power lines. It is also possible to incorporate crosstalk considerations into our cost models and metrics, which will not be discussed in this paper for brevity of presentation.

The algorithms presented in this paper are presented using a global routing model for the purpose of planning the routing topologies. Since the nets with matching requirements are critical nets, these nets are expected to be detailed routed before non-critical nets. This way, the final detailed routes of these nets are expected to highly correlate with their respective global routes.

The common notations used in the rest of the paper are listed in Figure 3.

[1]In general, if some of the horizontal (vertical) distances between the driver and receiver terminals are even, and some are odd, then matching all the segment lengths *exactly* may not be possible. Although this issue can be dealt with in practice with small modifications, we will assume that for each terminal T with coordinate (x, y) that x and y are even numbers for clarity. This can easily be achieved by increasing the granularity of the input global routing grid by a factor of 2.

3. THEORETICAL STUDY

In this section, we study the exact route matching problem theoretically. The algorithms we propose in Section 4 will be based on the results of this theoretical study. In the following subsections, we first propose a mathematical model to represent matching routing solutions. Then, we derive and prove upper and lower bounds for the number of bends needed for exactly matched routing solutions. After that, we focus on the cost minimization aspect, and we prove that the optimal min-bend min-cost exact route matching solution can be found in polynomial time for an asymptotically constant number of nets.

For the purposes of the theoretical study, we define a routing solution as *feasible* if the exact matching requirement in Definition 2.1 is satisfied. In this feasibility definition, we do not consider overlaps with hard blockages or chip boundaries. These issues are considered during cost minimization by setting the costs of such routes as infinite.

3.1 Solution Modeling

Let $\mathcal{N}$ be the given set of N nets of which routes are to be matched exactly. Let k_h (k_v) denote the number of horizontal (vertical) segments in the routing solution R_n of a net $n \in \mathcal{N}$. Based on the definitions given in Section 2, k_h and k_v values must be identical for all nets due to matching requirements. Furthermore, we can make the following observation.

Remark 3.1. *By definition, a valid routing solution must have $|k_h - k_v| \leq 1$.*

Since each segment is defined to be maximal in the 2-D space, a horizontal segment must follow a vertical segment (and vice versa) in any routing solution R_n.

Definition 3.1. *D_H (D_V) is defined to be the vector of horizontal (vertical) distances between driver and receiver terminals. In other words, $D_H[n]$ ($D_V[n]$) is equal to $t_R.x - t_D.x$ ($t_R.y - t_D.y$), for $1 \leq n \leq N$. Note that this value can be positive or negative depending on the relative locations of the driver and receiver terminals.*

Definition 3.2. *L_H (L_V) is defined to be the horizontal (vertical) segment length vector of size k_h (k_v), where $L_H[i]$ ($L_V[i]$) is the length of the i^{th} horizontal (vertical) segment in R_n.*

Definition 3.3. *The direction of segment s_i^n is defined based on the directional path from driver terminal t_D^n to receiver terminal t_R^n. Assume that s_i^n is from point (x_1, y_1) to point (x_2, y_2) in the path from t_D^n to t_R^n. If s_i^n is a horizontal (vertical) segment, it is defined to have positive direction iff $x_2 > x_1$ ($y_2 > y_1$), and vice versa.*

Definition 3.4. *C_H is defined to be the horizontal configuration matrix of size $(N \times k_h)$, where:*

$$C_H[n,i] = \begin{cases} 1 & ;\ i^{th} \text{ horizontal seg. in } R_n \text{ has positive direction} \\ -1 & ;\ i^{th} \text{ horizontal seg. in } R_n \text{ has negative direction} \end{cases} \quad (1)$$

Let $C_V[n,i]$ be the vertical configuration matrix of size $(N \times k_v)$ defined similarly.

Intuitively, we make sure that all nets have *exact matching* routes by using a single L_H (L_V) vector corresponding to all horizontal (vertical) segment lengths in all nets. On the other hand, the different coefficients in C_H (C_V) matrix corresponding to different nets *customize* the routes based on relative locations of driver and receiver terminals of different nets. Figure 4 illustrates an example for matching horizontal segments of 3 nets.

Theorem 3.2. *For a given set of nets $\mathcal{N}$, an exact matching routing solution must satisfy $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$. Furthermore, if $|k_h - k_v| = 1$, then the quartet (C_H, L_H, C_V, L_V) defines one unique matching 2-D solution without layer assignment. Otherwise, if $k_h = k_v$, then the quartet (C_H, L_H, C_V, L_V) defines two* *unique matching 2-D routing solutions without layer assignment. Different 3-D routing solutions can be generated out of these 2-D solutions through layer assignment.*

PROOF. The details of the proof are omitted due to page limitations. Intuitively, if $k_h = k_v + 1$, then the quartet (C_H, L_H, C_V, L_V) defines a unique 2-D route which starts with $L_H[1]$ and ends with $L_H[k_h]$, alternating between horizontal and vertical segments in the middle. If $k_h = k_v$, then there are two 2-D routes, which 1) starts with $L_H[1]$ and ends with $L_V[k_v]$, and (2) starts with $L_V[1]$ and ends with $L_H[k_h]$. □

3.2 Theoretical Bounds for Bend Counts

In the previous subsection, we have shown that 2-D route matching problem can be represented as a set of equations corresponding to the quartet (C_H, L_H, C_V, L_V). In this subsection, we derive theoretical upper and lower bounds for the number of bends required for exactly matching routes. Note that the number of bends is equal to $k_h + k_v - 1$ based on the definitions given before.

For the purpose of bend count analysis, we will assume in this subsection that all elements of D_H (D_V) are unique. Intuitively, if there are multiple nets with equal horizontal (vertical) distances, it is sufficient to consider a single entry in D_H (D_V) corresponding to these nets to minimize the number of bends. Let ud_h (ud_v) denote the number of unique values in D_H (D_V).

Lemma 3.3. *If the equations $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$ both have feasible solutions, then exact route matching problem has a feasible solution with bend count less than or equal to $(2 * max(k_h, k_v) - 1)$.*

PROOF. If $|k_h - k_v| \leq 1$, then the quartet (C_H, L_H, C_V, L_V) corresponds to a set of feasible matching solutions as stated in Theorem 3.2. Otherwise, assume that $k_h < k_v - 1$ without loss of generality. In this case, the number of horizontal segments k_h can be increased by segment splitting (as will be described in Section 4.2) while still satisfying the equation $C_H \times L_H = D_H$. Full proof of the lemma is omitted due to page limitations. □

Lemma 3.4. *There exists at least one feasible solution for the equation $C_H \times L_H = D_H$ with $k_h = ud_h$. Similarly, there exists at least one feasible solution for the equation $C_V \times L_V = D_V$ with $k_v = ud_v$.*

The proof of this lemma is based on defining configuration matrices C_{mH} and C_{mV} that correspond to a class of feasible solutions for L_H and L_V as follows.

Definition 3.5. *Given a D_H vector with unique elements, C_{mH} is defined to be the matching configuration matrix of size $(ud_h \times ud_h)$ such that:*

$$C_{mH}[i,j] = \begin{cases} sign(D_H[i]) & ; i \geq j \\ -sign(D_H[i]) & ; i < j \end{cases} \qquad (2)$$

Similarly, C_{mV} is defined to be the matching configuration matrix corresponding to D_V vector.

Lemma 3.5. *Without loss of generality, assume that the elements of D_H are sorted in increasing order of their absolute values, i.e. $|D_H[i]| < |D_H[j]|$ iff $i < j$. The solution to the equation $C_{mH} \times L_H = D_H$ (where C_{mH} is as defined in Definition 3.5) is:*

$$\begin{aligned} L_H[1] &= |D_H[1]| + \sum_{i=2}^{ud_h} \frac{|D_H[i]| - |D_H[i-1]|}{2} \\ L_H[2] &= \frac{|D_H[2]| - |D_H[1]|}{2} \\ &\ldots \\ L_H[i] &= \frac{|D_H[i]| - |D_H[i-1]|}{2} \\ &\ldots \\ L_H[ud_h] &= \frac{|D_H[ud_h]| - |D_H[ud_h - 1]|}{2} \end{aligned} \qquad (3)$$

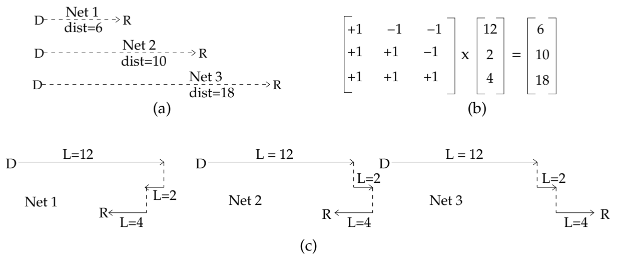


Figure 4: (a) 3 nets to match with different horizontal distances between terminals. (b) The solution to $C_{mH} \times L_H = D_H$. (c) The horizontal segments corresponding to L_H, where segment lengths are identical for each net. The vertical segments are not shown for clarity.

The elements of L_V vector can be computed in a similar way.

PROOF. This solution to L_H is obtained through Gaussian elimination. The proof is straightforward by solution substitution in the linear equation $C_{mH} \times L_H = D_H$, and each $L_H[i]$ is guaranteed to be positive because of sorted D_H. □

An example is illustrated in Figure 4 to demonstrate the physical correspondence of Lemma 3.5. In part (a), 3 nets are shown with different horizontal distances between driver (D) and receiver (R) terminals. The C_{mH} matrix in part (b) is set based on Definition 3.5, and D_H matrix is set based on horizontal distances between the terminals. When $C_{mH} \times L_H = D_H$ is solved for L_H (using Lemma 3.5), the segment lengths are obtained as: $[12; 2; 4]$, as given in Figure 4(b). The corresponding horizontal segments are illustrated in part (c). Observe that all nets have identical horizontal segments, and the coefficients in C_{mH} determine the directions of the segments.

Theorem 3.6. *For any set of N nets, there exists at least one exactly matching routing solution with optimal total wirelength and with bend count less than or equal to $2 \times max(ud_h, ud_v) - 1$. Furthermore, all segment sizes in this routing solution are guaranteed to be integers if all elements of D_H and D_V vectors are even integers.*

PROOF. The proof is due to Lemma 3.3 and Lemma 3.4. The integer guarantee and wirelength optimality are due to Lemma 3.5. □

The integer guarantee in Theorem 3.6 is important especially when routing is done on a coarse-grain grid (such as global routing grid). As mentioned in Section 2, we make sure that all D_H and D_V values are even integers by increasing the granularity of the input global routing grid by a factor of 2. This guarantees that there is an integer solution with exact matching. Alternatively, if changing the granularity of the routing grid is not feasible, we can perform rounding to map the segment lengths onto the grid.

Theorem 3.7. *The bend count stated in Theorem 3.6 is the minimum bend count that guarantees a feasible solution for any arbitrary route matching problem with N nets.*

PROOF. $C_H \times L_H = D_H$ is a system of linear equations. Consider a vector L_H with size less than ud_h. Since there are ud_h number of equations, the number of variables in the linear system will be less than the number of equations. For $C_H \times L_H = D_H$ to have a feasible solution, the coefficients in D_H need to be *consistent*. Although there may be special D_H vectors that allow L_H to have less than ud_h elements, the size of L_H needs to be at least ud_h to guarantee a solution for any D_H vector. □

3.3 Cost Minimization Through Column Permutation and Layer Assignment

As stated in Theorem 3.2, the solution to the equations $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$ corresponds to one or two unique 2-D matching routing solution without layer assignment. Given a configuration matrix C_H, and a segment length vector L_H that satisfies $C_H \times L_H = D_H$, we can permute the columns of C_H and the elements of L_H while still maintaining the equation. For example, consider the $C_H \times L_H = D_H$ equation given in Figure 4(b). We can swap the first two columns of C_H and the first two elements of L_H, and the equation $C_H \times L_H = D_H$ would still be satisfied. However, the order of horizontal segments would be changed from $[12; 2; 4]$ to $[2; 12; 4]$, which would lead to a different routing solution.

In general, assume that there are k_h horizontal segments and k_v vertical segments that satisfy the equations $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$. By permuting these horizontal and vertical segments, we can obtain $(k_h!) \times (k_v!)$ different 2-D routing solutions. Furthermore, each segment can be assigned to a different layer, which would further expand the solution space. We propose a dynamic programming algorithm in Section 4.1 to find the minimum cost routing solution among all these possible routing configurations.

3.4 Cost Minimization Through Continuous Space Exploration

Note that the number of elements in D_H (and hence the number of rows in C_H) is fixed for a given problem, since D_H stores the horizontal distances between input nets. However, the size of L_H (and hence the number of columns in C_V) can vary depending on the number of segments in the solution. In Section 3.2, we have derived upper and lower bounds for the minimum number of segments needed to satisfy the equations $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$.

On the other hand, there may be cases where more segments are needed than the minimum possible. For example, we may want to increase the number of bends to obtain a routing solution with a lower cost. Although our focus is on min-bend routing in this paper, it is straightforward to extend the algorithms proposed to explore solutions with more bends than the minimum. Furthermore, even for min-bend routing, we may need an L_H (L_V) vector that has more elements than minimum. The reason is due to Remark 3.1, which states that for a valid routing solution, we must have $|k_h - k_v| \le 1$.

In general, consider the equation $C_V \times L_V = D_V$ for a fixed C_V matrix and a fixed D_V vector (similar discussions also apply for $C_H \times L_H = D_H$). This corresponds to a system of linear equations, where the number of variables is equal to $sizeof(L_V)$, and the number equations is equal to $sizeof(D_V)$. If this system of equations is *linearly independent*, then there will be a unique solution for each element of L_V. Otherwise, the solution to L_V can be represented as one or more linear equations. An obvious example is when the number of equations is less than the number of variables. For example, consider a linear system with a single equation and two variables: $y_1 + y_2 = 8$. In this case, the solution to (y_1, y_2) is not unique, and it can be represented as a line segment in the 2-D space. Figure 7(b) illustrates an example for this case, where the segment sizes are not fixed, but function of each other. To find the optimal routing solution, the continuous solution space needs to be explored accordingly.

Further details about how to explore the continuous solution space to find the min-cost routing solution for a fixed configuration matrix C_H or C_V are given in Section 4.2.

3.5 Optimality Study

Theorem 3.8. *The general min-cost exact route matching problem for N nets is NP-complete even for $N = 2$.*

PROOF. The proof is based on reduction from the NP-complete problem of finding the path with maximum length. Details are omitted due to lack of space. □

Theorem 3.9. *The K-bend min-cost exact route matching problem for N nets is polynomial-time solvable if K and N are asymptotically constant.*

PROOF. The size of the solution space is polynomial in the number of grid points. Even the brute force enumeration of all solutions will lead to a polynomial-time algorithm. Since such a brute force approach would be too expensive even for small values of K and N, we propose more efficient algorithms in Section 4. □

Theorem 3.10. *The min-bend min-cost exact route matching problem for N nets is polynomial-time solvable if N is asymptotically constant.*

PROOF. Due to Theorem 3.6, there must be a feasible solution with bend count less than or equal to $2N$, which is constant. The proof follows due to Theorem 3.9. □

4. ALGORITHMS

Based on the theoretical study in the previous section, we propose new algorithms to solve the exact route matching problem for N nets. The algorithms proposed in this section will be based on one class of configuration matrices. We have chosen to use all permutations of the C_{mH} and C_{mV} matrices defined in Definition 3.5. As discussed in Section 3.2, using C_{mH} and C_{mV} leads to routing solutions with bend counts less than or equal to $2 \times max(ud_h, ud_v) - 1$, which is the minimum bend count that guarantees feasible solutions for any arbitrary problem. Alternatively, it is possible to explore different configuration matrices by solving the system of equations $C_H \times L_H = D_H$ and $C_V \times L_V = D_V$. However, in our experience, using one class of configuration matrices gave us a large enough solution space. Hence, this was sufficient for our practical purposes.

The basic idea of our algorithms is to explore the solution space defined by the configuration matrices C_{mH}, C_{mV}, and their permutations efficiently to find the minimum cost solution. For this, we first propose a dynamic programming (DP) algorithm that is guaranteed to find the minimum cost routing solution for a given class of configuration matrices. Then, we propose A*-based heuristics to search the solution space faster and in a more efficient way.

For ease of presentation, we will first describe the dynamic programming algorithm in Section 4.1 for segment permutation problem, assuming that $|ud_h - ud_v| \le 1$. Then, we will extend this algorithm in Section 4.2 to perform continuous space exploration by adding extra routing segments. In Section 4.3, we will propose optimal and greedy A* heuristics for faster computations.

4.1 DP Based Segment Permutation and Layer Assignment

In this subsection, we assume that $|ud_h - ud_v| \le 1$ for the ease of presentation, and we focus on solution space exploration through segment permutation only. The problem focused on in this subsection can be stated as follows. Given a set of nets $\mathcal{N}$, a set of horizontal segment lengths L_H, a set of vertical segment lengths L_V, and segment directions for each net defined by C_{mH} and C_{mV} matrices, compute the ordering and layer assignment of the segments that will lead to minimum total cost for nets in $\mathcal{N}$.

We propose a dynamic programming (DP) algorithm to solve this problem. For this, we first define independent subproblems as follows.

Definition 4.1. *A partial solution state can be defined as the triplet $\{\mathcal{S}_H, \mathcal{S}_V, last\_layer\}$, where $\mathcal{S}_H$ ($\mathcal{S}_V$) is the unordered set of horizontal (vertical) segments utilized in the partial solution.*

Lemma 4.1. *If two partial solutions A and B correspond to the same state (as defined in Definition 4.1), it is guaranteed that both A and B end exactly at the same 3-D coordinate for each net n in $\mathcal{N}$.*

PROOF. By definition, both partial solutions A and B contain the same set of horizontal and vertical segments (though possibly in different order). So the summation of horizontal segment lengths and vertical segment lengths are equal for each net in $\mathcal{N}$. Furthermore, the last layer of the partial solutions is defined in the solution state, and must be the same for all nets. □

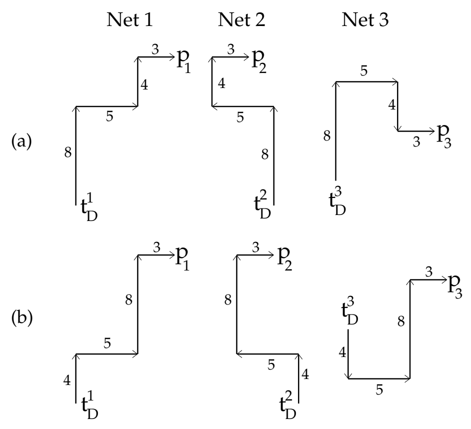


Figure 5: Two of the possible partial solutions corresponding to the state: $\{\mathcal{S}_H = \{4, 8\}, \mathcal{S}_V = \{3, 5\}, layer\}$ are illustrated in parts (a) and (b) for 3 nets. In the DP algorithm, only the partial solution with min cost will be stored in the priority queue.

Lemma 4.2. *Consider two partial solutions A and B that correspond to the same solution state, and $cost(A) < cost(B)$. It is guaranteed that B will not be part of any optimal solution.*

Figure 5 illustrates an example with two partial solutions in parts (a) and (b), corresponding to the same state. Observe that both partial solutions have the same set of horizontal and vertical segments, but in a different order. As stated in Lemma 4.1, different partial solutions for the same nets end exactly at the same 3-D coordinates. Here, the partial solutions for net1, net2, net3 end at $p1$, $p2$, $p3$ in both solutions. Also, due to Lemma 4.2, we need to consider only the partial solution with the min cost among the two alternatives in parts (a) and (b).

Based on the independent subproblem property, we propose a DP algorithm given in Figure 6. Here, we use a priority queue (pQ) to keep track of the minimum cost partial solution states. After the current min-cost state (curr_state) is extracted, it is expanded to new states by adding other segments from $(L_H \cup L_V)$ to curr_state. Note that not all segments in $(L_H \cup L_V)$ can be added to curr_state. For example, if the last_layer in curr_state is a horizontal layer, only vertical segments can be added to curr_state, and vice versa. Also, if a segment already exists in curr_state, it cannot be added again.

As stated in Lemma 4.1, curr_state ends at a specific coordinate for each net in $\mathcal{N}$. For example, in Figure 5, state A ends at $p1$, $p2$, and $p3$ for nets 1, 2, and 3, respectively. The cost of adding segment s to curr_state (denoted as $cost(curr\_state, s)$ in Figure 6) is computed based on placing s at the coordinates corresponding to each net. In the example of Figure 5, cost of adding segment s to state A can be computed as the sum of the cost of adding s: 1) at $p1$ for net 1, 2) at $p2$ for net 2, and 3) at $p3$ for net 3. Note that segment s may have positive or negative directions for different nets based on C_{mH} and C_{mV} matrices. The cost metric we use here is based on Definition 2.2, and it is as follows in lexicographic order: 1) overflow impact of s; 2) via cost due to layer switch from $last\_layer$ of $curr\_state$ to the layer of s. It is possible to perform some preprocessing to be able to compute overflow impact of each segment in $O(1)$ time. Due to lack of space, the details of this preprocessing will not be given.

After the new cost value is computed in Figure 6, we check whether there is an existing partial solution in the priority queue pQ corresponding to the new_state. If so, we update pQ based on the new cost if it has less cost. Otherwise, we insert the new_state into pQ. These iterations continue until a complete path is found from driver to receiver of each net. This happens when the $curr\_state$ extracted from pQ contains all segments in L_H and L_V.

Theorem 4.3. *The DP algorithm given in Figure 6 finds the optimal segment permutation and layer assignment for given pairs of configuration matrices (C_{mV}, C_{mH}), and distance vectors (D_H, D_V) if* $|ud_h - ud_v| \leq 1$.

```
MATCH-ROUTES ($\mathcal{N}$)
  Compute $L_H$ in $C_{mH} \times L_H = D_H$ using Lemma 3.5
  Compute $L_V$ in $C_{mV} \times L_V = D_V$ using Lemma 3.5
  Define a state as $\{\mathcal{S}_H, \mathcal{S}_V, last\_layer\}$
  curr_state ← empty_state ← $\{\emptyset, \emptyset, metal1\}$
  pQ ← empty_state // initialize priority queue
  while (all segments in $(L_H \cup L_V)$ not contained in curr_state)
    curr_state ← extract_min(pQ)
    for each segment $s$ in $(L_H \cup L_V)$
      for each layer $l$ in the layer set
        if $s$ can be added to curr_state
          new_state ← curr_state $\cup\{s\}$
          new_cost ← cost(curr_state, s) + cost(curr_state)
          if new_state exists in $pQ$ AND new_cost is smaller
            update new_state in $pQ$
          else
            insert new_state to $pQ$
  Perform backtracking to compute the best solution.
```

Figure 6: DP-based algorithm to compute matching routes for a set of nets $\mathcal{N}$

Theorem 4.4. *The DP algorithm in Figure 6 has space complexity of $O(grid\_size + 2^{ud_h} * 2^{ud_v})$, and time complexity of $O(grid\_size + 2^{ud_h} * 2^{ud_v} * ud_h * ud_v)$. Here, ud_h and ud_v are both less than or equal to the number of nets to be matched, which is expected to be a small constant. Hence, the asymptotic complexity of the DP algorithm is linear in the input grid size.*

Note that in most common cases, exact route matching requirement is imposed on less than or equal to 5 nets. Also, each iteration in DP algorithm is very simple and can be computed very fast. Furthermore, we propose A*-based heuristics in Section 4.3 to explore the search space a lot faster.

4.2 Continuous Space Exploration Through Segment Splitting

In the previous subsection, we assumed that $|ud_h - ud_v| \leq 1$ for a given set of N nets. In this subsection, we discuss how to handle the general cases where this may not be the case. Furthermore, the techniques described here can be used to add extra bends to the routing solution even when $|ud_h - ud_v| \leq 1$, for the purpose of further cost reductions.

Without loss of generality, assume that $ud_v < ud_h - 1$. Since the routing solution must alternate between horizontal and vertical segments, we need to have at least $ud_h - 1$ vertical segments, as described in Section 2. We know that solving the linear system $C_{mV} \times L_V = D_V$ will result in a vector L_V of size ud_v. So, we need to add at least $(ud_h - ud_v - 1)$ extra vertical segments in addition to the ones in L_V.

This concept is illustrated through an example in Figure 7. Here, the given D_H vector has 3 elements, but D_V vector has only 1 element corresponding to 3 nets. This is due to the fact that the terminals of all 3 nets are separated by the same vertical distance of 8, but by different horizontal distances: -4, 12, and 16. In this case, we have 3 horizontal segments (as given in L_H vector), but only 1 vertical segment (as given in L_V vector). To be able to construct a route that alternates between horizontal and vertical segments, we need to add 1 extra vertical segment. For this purpose, we can split the vertical segment into 2: with lengths y and $8 - y$. Here, the exact value of y should be determined while considering the cost minimization objectives.

The general problem can be formulated as follows. For given horizontal and vertical segment size vectors (L_H and L_V) and extra segment counts (e_h, e_v) that need to be introduced in the horizontal and vertical orientations, find the routing solution with minimum total cost. In this paper, since we focus on min-bend routing solutions, we add extra segments only when it is necessary, i.e. when $|ud_h - ud_v| > 1$. In such cases, only one of e_h or e_v will be nonzero, and it will be equal to

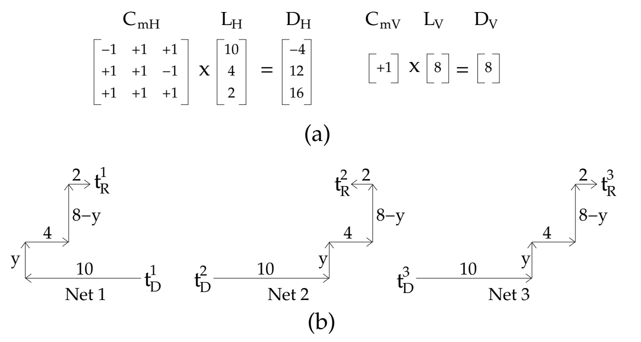


Figure 7: An example illustrating a case where $|ud_h - ud_v| > 1$. In this case, the vertical segment needs to be split into two with lengths: y and $8 - y$.

$|ud_h - ud_v - 1|$. Without loss of generality, assume that $ud_h < ud_v - 1$ in the rest of this subsection.

The dynamic programming formulation given in Section 4.1 can be extended to solve this problem. For this, we need to redefine the partial solution state as follows.

Definition 4.2. *A partial solution state can be defined as the quartet $\{\mathcal{S}_H^c, \mathcal{S}_V, last\_layer, extra\_bend\_count\}$, where $\mathcal{S}_V$ is the unordered set of vertical segment IDs, and $\mathcal{S}_H^c$ is the unordered set of pairs of horizontal segment IDs and partial wirelengths. In other words, while the lengths of the vertical segments are discrete, the lengths of the horizontal segments are continuous variables.*

As an example, consider a problem with $L_H = [8, 10, 14]$, and assume that we need to add 1 extra horizontal segment. For brevity, the vertical segments are ignored in this example. A partial solution state can be the quartet: $(\mathcal{S}_H^c = \{(id1, 8), (id2, 10)\}, \mathcal{S}_V, layer, 0\})$. In this partial solution, there exist two horizontal segments without any extra bends. This partial solution can be later expanded to a new state: $(\mathcal{S}_H^c = \{(id1, 8), (id2, 10), (id3, 6)\}, \mathcal{S}_V', layer', 1\})$. In this partial solution, the third horizontal segment is split, and it has partial length of 6. Hence, the $extra\_bend\_count$ variable in the state is set to 1. Later, it can be expanded to a new state: $(\mathcal{S}_H^c = \{(id1, 8), (id2, 10), (id3, 14)\}, \mathcal{S}_V'', layer'', 1\})$. In this partial solution state, all horizontal segments are utilized, and 1 extra bend has been added.

The independent subproblem properties stated in Lemma 4.1 and 4.2 still hold for this extended formulation. Furthermore, the DP algorithm given in Figure 6 can be extended in a straightforward way as follows. Every time a new segment s is added to the $curr\_state$, we need to consider all partial segment lengths of s from 1 to the full length of s. Furthermore, if a partial segment s is added to $curr\_state$, the $extra\_bend\_count$ field in new_state needs to be incremented by one. If the $extra\_bend\_count$ field of $curr\_state$ is already equal to the maximum value, then only full segments are allowed to be added to $curr\_state$.

Theorem 4.5. *If the number of nets to be matched is asymptotically constant, the extended DP algorithm that can handle continuous segment sizes has time and space complexity of $O(grid\_size + l_{max}^{|ud_h - ud_v - 1|})$, where l_{max} is the maximum element in $(L_H \cup L_V)$.*

Note that the number of extra segments that need to be added for matching up to 5 nets is typically small. Hence, in our practical experience, we were able to afford to run the optimal DP formulation. However, if more bends are needed, different heuristic algorithms can be proposed that are derived from this formulation. Furthermore, we propose A*-based techniques to explore the search space faster.

4.3 Efficient A* Search Techniques

It is well known that graph path search problems can be solved efficiently using A*-based heuristics. The basic A* idea is to replace the original cost function $g(x)$ with $g(x)+h(x)$, where $h(x)$ is an estimate of the cost from node x to the target. The $h(x)$ function is defined to be *admissible* if $h(x)$ does not overestimate the actual cost from node x to the target. It is known that if the heuristic estimator is admissible, the path obtained is guaranteed to be the optimal path. A* heuristics have been widely used by general purpose routing algorithms such as [14] [2].

The dynamic programming (DP) formulation proposed in the previous subsections can be considered as a conceptual graph, where each partial solution state corresponds to a node, and each transition between two states corresponds to an edge. Furthermore, we define an admissible heuristic estimation function from a DP state to the target state based on the cost metric in Definition 2.2.

Since the total segment lengths are fixed in our formulation, the wirelength estimation can be done exactly. The lower bound for the total segment cost from each physical location (x, y) to each receiver terminal is precomputed before the DP algorithm begins using Dijkstra's shortest path algorithm. Similarly, the number of vias from the current state to the target can be computed based on $last\_layer$ field in the current state. In summary, we use an admissible heuristic estimator for our A* search so that the optimal solution is guaranteed for DP without having to generate every partial solution state. This improves our runtimes without sacrificing solution quality.

It is also known that a trade-off between solution quality and runtime can be achieved by using heuristic estimators that potentially overestimate the cost to the target. If runtime of the DP algorithm becomes an issue, such heuristics can be utilized.

5. EXPERIMENTAL RESULTS

Although we were able to test some of our algorithms on real industrial analog designs, we report our results on public routing benchmarks from *ISPD'07 Global Routing (GR) Contest* [6] due to IP restrictions. In a typical mixed-signal design, the analog nets need to be routed in the same design with digital nets. So, it is realistic to use the ISPD'07 benchmarks to evaluate the effectiveness of our algorithms.

For our experiments, we first obtained the GR solutions of Archer [11] on these benchmarks. Based on these solutions, we have extracted a congestion map for each routed design, and increased the granularity of the GR grid by 2 in both dimensions, as discussed in Section 2. After that, we have randomly chosen a set of nets from each benchmark circuit to impose matching constraints on, and we have removed their global routes. In particular, we have chosen the nets with driver-to-receiver distances between 100 and 200 GR grid cells. For our evaluation purposes, we can safely assume that these nets are the *analog* nets with matching constraints, while the other already-routed nets are the digital nets. We have enforced the exact matching constraints on groups of nets, where each group contains between 2 and 5 nets, which is the most typical size in practice. For each benchmark in Table 1, we have randomly chosen 50 such groups. During random selection for some benchmarks, we have encountered 1 or 2 (out of 50) net groups for which exact matching was not possible due to terminals being too close to chip boundaries (and hence making it impossible to match the routes exactly without going out of the chip boundary). For such nets, either the placement needs to be changed or the exact matching constraints need to be relaxed by the designers. Hence, we have excluded such corner cases from our benchmarks. Based on this setup, we have used *overflow* as our segment cost metric in Definition 2.2.

For our Spice simulations, we used the public parameters given for *ISPD'09 CTS Contest* [17] for 45nm technology from PTM [13]. We computed the size of each GR grid cell by multiplying the number of tracks specified in each benchmark by the track pitch specified in [17]. Since via resistance was not specified in [17], we used the guideline given in *ISPD'08 GR Contest* [16], which stated that one via corresponds to 1-tile length of wire. Furthermore, we have used the same narrow-wire R/C (from [17]) for wires on layers metal1-metal4, and the same wide-wire R/C for wires on layers metal5-metal6.

Table 1: Experimental results

Benchmark	EXACT-MATCH (DP)							EXACT-MATCH (Randomized)							LR-BASED MATCH						
	cong cost	via cnt	bend cnt	len. mism.	via mism.	len+via mism.	delay mism.	cong cost	via cnt	bend cnt	len. mism.	via mism.	len+via mism.	delay mism.	cong cost	via cnt	bend cnt	len. mism.	via mism.	len+via mism.	delay mism.
adaptec1	411	72	27	0%	0%	0%	0%	568	52	27	0%	0%	0%	0%	364	59	47	13%	71%	8%	16%
adaptec2	34	67	27	0%	0%	0%	0%	112	52	27	0%	0%	0%	0%	69	50	43	11%	80%	7%	44%
adaptec3	95	69	27	0%	0%	0%	0%	201	54	27	0%	0%	0%	0%	156	54	46	14%	83%	9%	36%
adaptec4	15	64	27	0%	0%	0%	0%	120	55	27	0%	0%	0%	0%	71	55	47	14%	81%	9%	31%
adaptec5	175	75	26	0%	0%	0%	0%	323	57	26	0%	0%	0%	0%	192	52	41	11%	77%	7%	24%
newblue1	40	65	26	0%	0%	0%	0%	147	52	26	0%	0%	0%	0%	79	51	45	13%	83%	8%	31%
newblue2	74	61	25	0%	0%	0%	0%	200	52	25	0%	0%	0%	0%	117	61	50	13%	77%	7%	16%
newblue3	20	60	25	0%	0%	0%	0%	70	48	25	0%	0%	0%	0%	75	47	40	12%	83%	7%	32%
Avg	108	67	26	0%	0%	0%	0%	218	53	26	0%	0%	0%	0%	140	54	45	13%	79%	8%	29%

For comparison, we have used the code for Lagrangian relaxation (LR)-based length matching algorithm [9]. Since this algorithm was originally developed for board level routing, we had to modify the code to make it applicable to IC-level routing. In the modified code, we also considered the impact of via resistances while doing length matching by assuming that 1 via corresponds to 1 GR grid cell as in [16]. To the best of our knowledge, this is the most relevant work in the literature that is applicable to this problem. We also implemented a randomized version of our algorithm, which uses the same route matching models, but is not based on dynamic programming (DP). Instead, it randomly enumerates a fixed number (about 50,000) of different matching solutions, and picks the best one.

Table 1 lists the results of our algorithms (DP and randomized versions) and LR-based algorithm. Each entry in the table is an average of 50 groups of nets, where all nets in one group have the exact matching constraints. In this table, the congestion costs, via counts, and bend counts are reported per net group. Furthermore, the wirelength mismatch, via count mismatch, and (wirelength+via count) mismatch percentages are reported per net group, where each mismatch value is computed as: $(max\_value - min\_value)/max\_value$. For all instances, our algorithms took less than 0.1 seconds, while the LR-based algorithm took more than minutes. We believe that runtime comparison with the LR-based algorithm is not fair, because it was originally implemented for board-level designs, where the problem sizes are much smaller. Hence, runtimes are not reported in this table.

As shown in Table 1, matching wirelengths is not enough for matching delays. Even though the LR-based algorithm can match the (wirelength + via count) values for different nets reasonably well, the routes obtained have significantly different interconnect characteristics, leading to delay mismatches of 29% on the average. On the other hand, the algorithms proposed in this paper find the routing solutions that match each other exactly. Hence, the interconnect characteristics for these nets end up to be identical.

Another point to note about Table 1 is that the DP-based exact matching algorithm outperforms the others in terms of congestion cost minimization objective. This is the case even though the LR-based algorithm uses maze routing, and our algorithms are based on pattern routing. Furthermore, we can see that the DP-based algorithm explores the solution space more thoroughly, and finds solutions with smaller costs, compared to the randomized algorithm.

6. CONCLUSIONS

In this paper, we studied the exact route matching problem, which is an important requirement for analog designs. We have proposed mathematical models for exact route matching, and we provided a theoretical study to derive important theoretical conclusions. Then, we proposed a dynamic programming algorithm to solve this problem efficiently. This algorithm can be used to automatically match the routes of a given set of nets so that interconnects have identical electrical properties, which is usually a tedious manual task in practice.

7. ACKNOWLEDGEMENTS

The authors would like to thank Gustavo Wilke (UFRGS, Brazil) for the Spice experimental setup, and Mircea Draghicescu and Steven M. Burns (Intel Corporation) for valuable discussions.

8. REFERENCES

[1] M. Bai, C. Auth, and et al. A 65nm logic technology featuring 35nm gate length, enhanced channel strain, 8 cu interconnect layers, low-k ILD and $0.57um^2$ SRAM cell. In *Proceedings of International Electron Devices Meeting (IEDM)*, 2004.

[2] R. Hentschke, J. Narasimhan, M. Johann, and R. Reis. Maze routing steiner trees with delay versus wire length tradeoff. *IEEE Trans. on VLSI Systems*, 17:1073–1086, 2009.

[3] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh. Predictable routing. In *Proc. of ICCAD*, pages 110–114, 2000.

[4] P.-H. Lin, H.-C. Yu, T.-H. Tsai, and S.-C. Lin. A matching-based placement and routing system for analog design. In *Proc. of VLSI-DAT*, pages 1–4, 2007.

[5] Mai, Alon, and Labonte. Analog layout lecture notes. http://www.stanford.edu/class/ee272/lectures/lect.10.pdf.

[6] G.-J. Nam. ISPD 2007 Global Routing Contest, 2007.

[7] K. Oda, P. L., and A. J. Gadient. A new methodology for analog/mixed-signal (AMS) SoC design that enables AMS design reuse and achieves full-custom performance. In *Proc. of the 9th IEEE/DATC Electronic Design Process Workshop*, 2002.

[8] M. M. Ozdal and M. D. F. Wong. Algorithmic study of single-layer bus routing for high-speed boards. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25:490–503, 2006.

[9] M. M. Ozdal and M. D. F. Wong. A length-matching routing algorithm for high-performance printed circuit boards. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25:2784–2794, 2006.

[10] M. M. Ozdal and M. D. F. Wong. Two-layer bus routing for high-speed printed circuit boards. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 11:213–227, 2006.

[11] M. M. Ozdal and M. D. F. Wong. Archer: A history-based global routing algorithm. *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)*, 28(4):528–540, 2009.

[12] M. Pan and C. Chu. Fastroute: A step to integrate global routing into placement. In *Proc. of Int'l Conf. on Computer Aided Design*, pages 464–471, 2006.

[13] PTM: Predictive Technology Model. http://www.eas.asu.edu/ ptm.

[14] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *Proc. of ICCAD*, pages 496–502, 2007.

[15] R. A. Rutenbar. Design automation for analog: The next generation of tool challenges. In *Proc. of ICCAD*, 2006.

[16] C. Sze. ISPD 2008 Global Routing Contest, 2008.

[17] C. Sze. ISPD 2009 Clock Network Synthesis Contest, 2009.

[18] T. Yan and M. Wong. BSG-Route: A length-matching router for general topology. In *Proc. of ICCAD*, pages 499–505, 2008.