A PRE-SEARCH ASSISTED ILP APPROACH TO ANALOG INTEGRATED CIRCUIT ROUTING

A Thesis

by

CHIA-YU WU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2015

Major Subject: Computer Engineering

ABSTRACT

The routing of analog integrated circuits (IC) has long been a challenge due to numerous constraints (such as symmetry and topology-matching) that matter for overall circuit performance. Existing automatic analog IC routing algorithms can be broadly categorized into two approaches: sequential approach that heuristically routes one net after another and constructive ILP (Integer Linear Programming). The former approach is usually fast but may miss opportunities of finding good solutions. The constructive ILP provides optimal solutions but can be very time consuming. We propose a simple yet efficient method that combines the advantages of both existing approaches. First, sequential routing is performed to obtain a set of candidate routing paths for each net. Then, an ILP is applied to commit each net to only one of its candidate routes. Experiments on two op-amp designs show that the post-layout performance (such as gain and phase margin) from our method is close to that of manual design. Our method also outperforms a previous work of automated analog IC routing.

DEDICATION

To my family

ACKNOWLEDGEMENTS

I would like to thank all those who encouraged me and helped me during my study and research at Texas A&M University. Firstly, I would like to give my heartfelt gratitude to my advisor, Dr. Jiang Hu, who gave me constant guidance as well as warm encouragement throughout this research project. He was always patient, kind and helpful whenever I had questions on my academic life. I could not have completed this thesis and know this wondrous physical design without his guidance and generous support. I also would like to thank Dr. Peng Li and Dr. Anxiao Jiang for being my committee members, and for their suggestions on this research. Especially, I would like to thank my family for their ceaseless support, encouragement and endless love. Without which I would never able to complete my master degree so smoothly. Last but not least, thanks to all my friends and colleagues who accompany with me these years, and also the department faculty and staff for giving me a warm and kind environment during my life at Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Physical Layout Design in Analog Circuit

Nowadays, analog integrated circuits (IC) design is still mostly a manual process, which is typically very time consuming. A key reason is that it involves a large amount of constraints, which are hard to capture, and makes automated design tools very difficult to be competitive. Routing of analog IC is no exception. Unlike digital IC routing, where the constraints are mostly restricted to spacing and capacity, analog routing additionally entails constraints of symmetry, topology-matching, wirelength-matching, etc. These constraints are vital for obtaining a desired analog circuit performance, such as gain, phase margin and linearity. Simultaneously satisfying these constraints has been a challenge for automated software tools.

1.2 Previous Works

People have been trying hard to develop analog IC routing algorithms. Most of the previous works can be viewed from two perspectives:

(i) What constraints do they follow?

(ii) What routing methodology do they use?

To answer the first question, satisfying matching constraints is a very important to achieve the best analog IC performance. Most of the previous works focus on symmetry constraints [1, 2], exact-matching [3], wirelength-matching constraints [1, 4], and topology-matching constraints [1, 3]. Symmetry constraints and topology-matching constrains are two most major constraints to reduce the parasitic effects. However, it is not easy to satisfy these constraints due to the resource-consuming process. For the less restricted nets, wirelength-matching will achieve a reasonably good performance. More detail about matching constraints will be elaborated in Section 2.3

From the viewpoint of routing methodology, most of previous work [5, 6] is based on sequential routing. That is, the signal nets are routed one after another [5]. Each two-pin net is usually routed with maze routing algorithm [5]. By properly defining edge cost in the routing graph, the sequential maze routing can satisfy parasitic and performance sensitivity constraints. For two nets with symmetry constraints, one is routed and the other is obtained by mirroring the first net routing. In order to avoid spatial contention with other nets, nets with symmetry constraints are routed with higher priority. Multi-pin nets can be routed with rectilinear Steiner trees. Sometimes, the Steiner trees are further decomposed into two-pin nets. A survey of analog routing works is provided in [5].

In sequential routing, the decision for routing a net is based on the space occupation of previously routed nets and pays no attention to nets to be routed later. Among multiple equally good routes for a net itself, the router may inadvertently choose one that hinders the subsequent routing. Figure 1.1 show an example that sequential routing cannot find a feasible solution. In [7], this weakness is mitigated by rip-up and reroute. However, the rerouting of a net is based on the routing of the other nets, which might be poor in the first place. Another work in [8] improved the weakness of sequential routing by generating candidate routes firstly and then select from candidate routes to construct a feasible solution. However, the mechanism for selecting the candidate routes is a greedy heuristic, which may result in inferior solutions. A more radical solution is ILP (Integer Linear Programming) [1], which is able to eliminate the weakness of sequential routing. In the ILP approach [1], which we term as constructive ILP, the 0-1 decision variables tell if to assign an edge in the routing graph to a net. Besides typical layout constraints, the constructive ILP entails additional constraints to ensure that the edges assigned to a net form a legitimate route. The constructive ILP is generally very time consuming and has poor scalability.

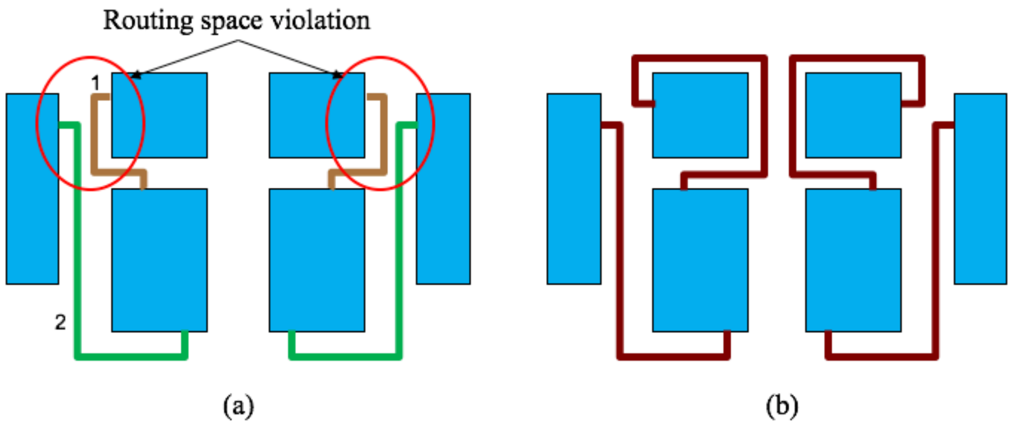

Figure 1.1: (a) Weakness of sequentail routing. (b) Simultaneous routing can eliminate the weakness.

1.3 Our Contributions

In this paper, we introduce a simple yet efficient approach that attempts to achieve the high routing quality of the constructive ILP and the runtime of sequential routing. The main idea is to first generate multiple candidate routes for each net independently considering various constraints. In the second phase, we use ILP, which we call pre-search assisted ILP, to choose only one route among the candidate routes for each net such that spatial contentions and crosstalk issues are solved. Compared to the constructive ILP, the pre-search assisted ILP avoids the constraints for legitimate paths and has much less decision variables as the solution space has already been narrowed down. Overall, our approach is faster than the constructive ILP and often finds better solutions than sequential routing. A similar idea was explored in [8]. However, its selection procedure is a greedy heuristic, which may result in inferior solutions. The contributions of our work include:

- We propose a new ILP-based analog routing algorithm which simultaneously considers constraints of symmetry, topology-matching, bend-matching, orientation-matching, wirelength-matching and wire parasitic. To the best of our knowledge, this is the first work that handles all of these constraints at the same time.

- The pre-search has a large flexibility to incorporate designer's intentions as it can be performed on different routing grids and can even be obtained manually. This is another advantage of our approach over the constructive ILP.

Our method is tested on two op-amp designs. The post-layout performance (such as gain and phase margin) is near to manual layout designs, while our approach takes only a couple of minutes compared to hours that a manual design may take. We also compared with a previous work [6]. In one case, both [6] and our approach result in similar solution. In the other case, our method finds a solution close to manual design while [6] fails to find a feasible solution.

1.4 Overview

The remainder of this thesis is organized as follows. Chapter 2 is to introduce the background knowledge, assumptions, and problem formulation. Chapter 3 introduces our Pre-search Assisted ILP Approach to Analog Integrated Circuit Routing and the detailed implementation. Chapter 4 shows our experimental results. Chapter 5 concludes this thesis.

2. PRELIMINARIES

2.1 Problem Formulation

The analog IC routing problem can be defined as:

Given an analog circuit design composed by a set of placed device modules $\mathcal{M} = \{M_1, M_2, ...\}$*, a set of nets* $\mathcal{N} = \{N_1, N_2, ...\}$*, a routing grid* $G = (V, E)$ *where* V *is a set of nodes and* E *is a set of routing edges, connect all pins of each net through a wiring tree on* G *such that a linear combination of total wirelength and total number of wiring bends is minimized subject to a set of constraints, which are elaborated in the next section.*

2.2 Assumptions

In this work, we follow the usual assumptions of analog IC layouts.

- **Assumption 1**: In order to reduce crosstalk between different signal wires and transistors, we follow the convention [6] that wires are not allowed to go above the active area of transistors. In other words, the active area of devices should be considered as routing blockages. This is illustrated in Figure 2.1(a).
- **Assumption 2**: This is the exception to Assumption 1. If a connection in a net is between the source and drain of the same active area, the connection can be implemented by wires routed above the active area. Since the connection is usually the shortest possible, we assume it is done as such without further being considered in the routing algorithm. This is illustrated in Figure 2.1(b).
- **Assumption 3**: Designers usually put guard rings around devices to shield noises. Guard rings provide well voltage by connecting with VSS and VDD signals, which are assumed to be routed on metal 1. (See Figure 2.2)

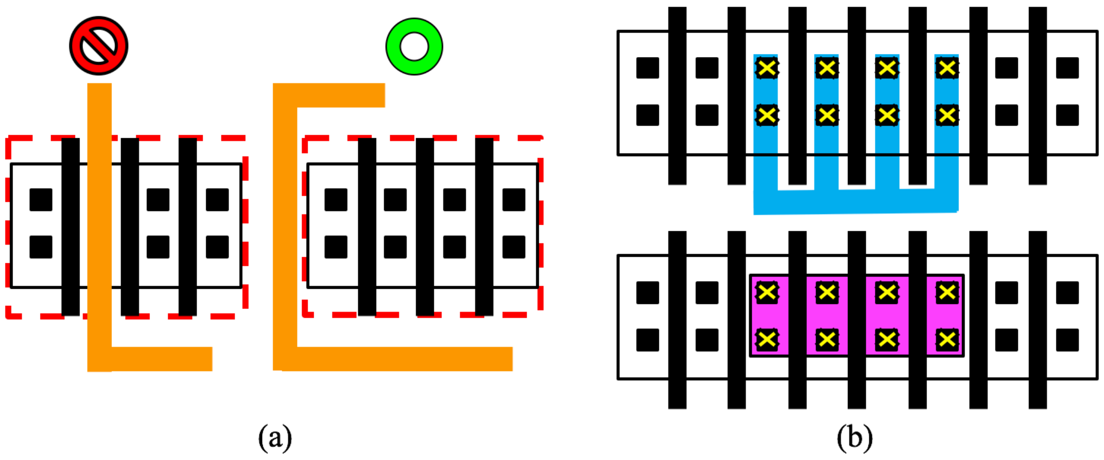

Figure 2.1: (a) Assumption 1: wire cannot overlap with active area indicated by dashed red rectangles. (b) An exception to Assumption 1.

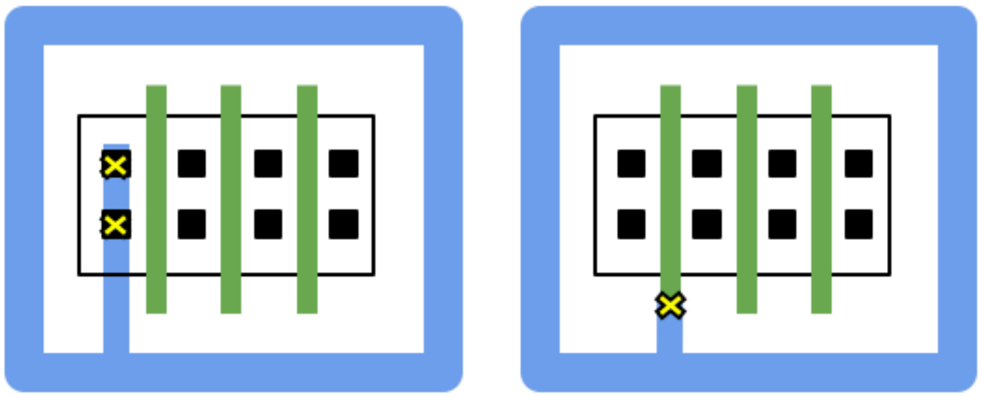

Figure 2.2: Examples of guard rings around the devices.

- **Assumption 4**: If a device has multiple fingers in layout, there must be two access pins for its poly. In other words, the multi-finger poly has two pins for its signal net. This is to improve signal conductivity for transistors.

2.3 Routing Constraints

Layout is the blueprint of planar geometric shapes that are used to create photolithography masks for an IC in a specific fabrication technology. Therefore, analog IC routing constraints can be described by geometric forms, for example, length, width, distance, spacing between wires, etc.

The most fundamental constraints are due to *layout design rules*, which are to ensure manufacturability. These rules are required for digital circuits as well and are incorporated in the routing grid $G = (V, E)$. In the routing algorithm, the constraints are then simplified to routing capacity constraints. That is, at most one wire segment can be routed on each edge $e \in E$.

Another set of constraints arises from *parasitic-dependent performance* requirement in analog circuits. Although the performance of digital circuits also depends on layout parasitic, the performance-parasitic dependence in analog circuit is usually more complicated. To facilitate analog circuit performance, one often needs to restrict wire resistance/capacitance, the number of vias and coupling capacitance between adjacent wires.

Some constraints are *specific to analog circuits*, for instance, symmetry constraints. Some analog circuit components, such as differential pairs, are composed of two structurally symmetric parts. The layout for the two parts needs to be symmetric as well. When variations (like process and thermal variations) are significant, they tend to manifest in the same way in two symmetric parts and cancel out each other in the overall effect. Given one route of a signal, its symmetrical route can be obtained by flipping the given route around the layout symmetry axis. Topology-matching constraints are very similar to symmetry constraints. To obtain a topology-matching route for a given route, one needs to perform shifting operations in addition to flipping. Other constraints include bend-matching, orientation-matching, wirelength-matching. All of these *analog-specific constraints* are illustrated in Figure 2.3.

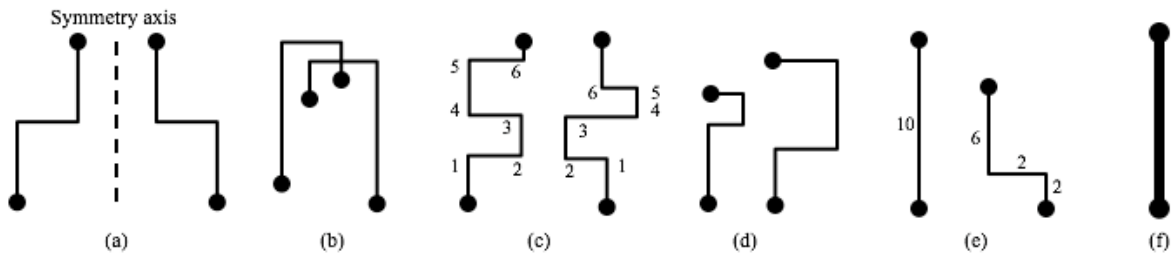

Figure 2.3: Examples of analog circuit routing constraints: (a) symmetry constraint; (b) topology-matching constraint; (c) bend-matching constraint; (d) orientation-matching constraint; (e) wirelength-matching constraint; (f) width constraint.

In addition, there are *reliability constraints*. For example, wire width has a minimum bound to reduce the risk of electromigration [9].

Altogether, analog IC routing constraints are much more complicated than those in digital circuits, especially the symmetry and topology-matching constraints. To help the description of our algorithm techniques, we categorize the constraints into two types:

- **Single-net constraints**: These are the constraints that can be specified for each net individually, such as wire length, wire width and the number of bends.
- **Multi-net constraints**: These are the constraints that involve interactions among multiple nets, such as symmetry, topology-matching and coupling capacitance constraints.

These two types of constraints will be enforced in different stages of our algorithm.

2.4 Fast Lookup Table Based Technique (FLUTE)

In our work, we adopt FLUTE [10] algorithm to generate the Rectilinear Steiner Minimal Tree for the high-degree-nets. The FLUTE is based on pre-computed lookup table to generate minimum spanning tree very fast and very accurate. For low-degree nets less than 9 nets, the set of all degree-n nets can be partitioned into n! groups according to the relative positions of their pins. For the nets with higher than 9 degree, FLUTE broke the nets into several sub-nets with degree ranging from 2 to 9 to avoid huge CPU time and memory requirement. FLUTE helps separate the multi-pin nets into sets of two-pin nets for every nets with the Steiner points.

2.5 Bend-aware A* Search Algorithm

In analog circuits, designers try to reduce number of bend and number of bends (or vias) as much as possible for better performance. We propose a Bend-aware A* search algorithm and it properly perform this convention as well. It is modified based on a well-known variant of Dijkstra's shortest path algorithm called A* search. The pseudo code for the Bend-aware A* search is provided in Algorithm 1.

The notations in Algorithms 1:

- S_{open} : the set of nodes to be determined
- S_{closed} : the set of nodes to be determined
- v_c : current determined node
- $cost_{current}(v)$: current cost of the node v
- $Neighbors(v)$: neighbors of the node v
- $came\_from(v)$: came from of the node v

Algorithm 1 Algorithm of A* search

```
Input: Routing grid $G = (V, E)$
    Source node $v_s$
    Target node $v_t$
Output: A minimum $\alpha w(p) + \beta b(p)$ routes $p$
1: for each $v_i \in G$ do
2:    $came\_from(v_i) \leftarrow 0$
3:    $cost_{current}(v_i) \leftarrow 0$
4: end for
5: $S_{closed} \leftarrow \emptyset$
6: $S_{open} \leftarrow \{v_s\}$
7: while $S_{open} \neq \emptyset$ do
8:    $v_c \leftarrow$ the highest priority node in $S_{open}$
9:    if $v_c = v_t$ then
10:       construct path $p$
11:       return $p$
12:   end if
13:   $S_{open} \leftarrow S_{open} - \{v_c\}$
14:   $S_{closed} \leftarrow S_{closed} \cup \{v_c\}$
15:   for each $v_i \in Neighbors(v_c)$ do
16:       $cost_{new}(v_i) \leftarrow costcurrent(v_c) + f_{cost}(v_c, v_i)$
17:       if $v_i \notin S_{closed}$ then
18:          continue
19:       end if
20:       if $v_i \notin S_{open}$ or $cost_{new} < cost_{current}(v_i)$ then
21:          $cost_{current}(v_i) \leftarrow cost_{new}(v_i)$
22:          $priority(v_i) \leftarrow cost_{current} + f_{remain\_cost}(v_c, v_t)$
23:          $S_{open} \leftarrow S_{open} \cup \{v_i\}$
24:          $came\_from(v_i) \leftarrow v_c$
25:       end if
26:   end for
27: end while
```

- $priority(v)$: priority of the node v

The difference between A* search algorithm and the Bend-aware A* search algorithm is the method to determine the cost value of the nodes. The basic A* search algorithm just uses the actual edge cost from the start and it is not enough to generate path with minimum number of bends. We addressed the direction of node into the cost function with two customized constants α and β. While updating the cost of the neighbors for current determined node(step 16), the direction that current determined node v_c came from has been stored in $came\_from(v_c)$ already at the previous iteration. The cost function is trade-off the turning to edge cost. Thus, the priority in S_{open} is based on the actual edge cost from the source, the estimated edge cost to the target, and the weighted number of bends. The path bend-aware A* found will be the minimum $\alpha w(p) + \beta b(p)$ route.

3. THE NEW ANALOG IC ROUTING APPROACH

Our main idea is to generate a set of candidate routes for each net and then perform ILP (Integer Linear Programming) to commit each net to only one of its candidate routes. Sequential routing is fast but poor at handling interactions among multiple nets. In contrast, ILP is good at handling interactions among nets but is slow. Our approach attempts to combine the advantages of both techniques. The candidate routes generation is focused on constructing high quality routes for individual nets and considering single-net constraints. Unlike the ILP in [1], which takes care of the complete routing procedure, our ILP emphasizes only on the interactions among nets and multi-net constraints such that its computing load is remarkably reduced. A similar approach has been applied in networks-on-chip routing [11]. However, our situation is more complicated than that in [11]. Between the candidate routes generation and ILP route selection, we need to have another stage of candidate refinement and annotation. This is to process the candidate routes for generating appropriate constraints for the ILP. For example, it can pair up routes that satisfy bend-matching constraint. An overview of our approach is provided in Figure 3.1. The details of each stage are elaborated in subsequent sections.

3.1 Candidate Routes Generation

For each net $N_i \in \mathcal{N}$, we wish to find a set of candidate routes, which can be selected by the ILP. This stage boils down to three sub-problems:

(i) How many candidate routes do we need?

(ii) What kind of candidate route do we prefer?

(iii) How to generate desired candidate routes?

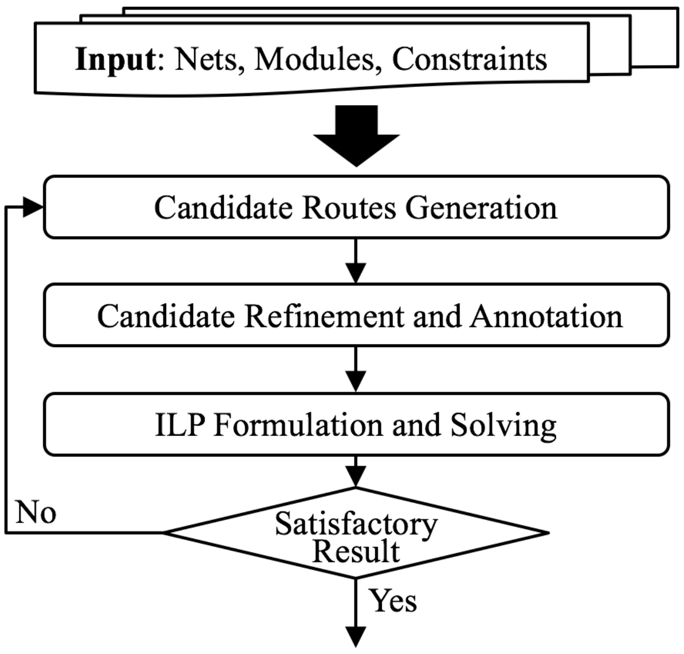
Input: Nets, Modules, Constraints
Candidate Routes Generation
Candidate Refinement and Annotation
ILP Formulation and Solving
Satisfactory Result
No
Yes

Figure 3.1: Overview of the new routing approach.

The first sub-problem is a matter of trade-off between solution quality and runtime. If we find all possible routes as candidates, our approach would lead to the optimal solution. If the number of candidate routes is too small, e.g., only one candidate route per net in the extreme case, the ILP would mostly fail to find a feasible solution. Evidently, runtime cost increases proportionally with the number of candidate routes to be generated. There are two ways to address the trade-off. The first is to empirically find a number that is large enough to obtain good solutions yet its resulting runtime is practical. The second is to start with a small number and then incrementally add new candidates based on the feedback from ILP results (see Figure 3.1).

The second sub-problem involves two parts: (a) a candidate route should fulfill the objective function and single-net constraints; (b) the candidate routes for a net should have a good chance for avoiding contention with other nets in the ILP. Part (a) can be achieved by generating candidate routes with short wirlength, small number of bends, limited parasitic, etc. In order to address (b), we take care that the candidate routes are diversified.

Our approach has three key elements to solve the third sub-problem with consideration of (a) and (b) in the second sub-problem.

1. We decompose each multi-pin net N_i into a set of two-pin nets $\mathcal{N}_i = \{N_{i,1}, N_{i,2}, ...\}$ using FLUTE [10], which is a rectilinear Steiner minimum tree software, as directly generating diversified multi-pin routes is difficult. Actually, such decomposition is fairly common in many routing works.

2. For each two-pin net, we propose bend-aware A* search (mentioned in Section 2.4) to generate routes with small wirelength and small number of bends.

3. In the routing grid $G = (V, E)$, if an edge $e \in E$ has already been used by a candidate route for net $N_{i,j}$, we increment its edge cost with a small amount. This edge cost increase discourages this edge to be used again in later candidate routes generation for $N_{i,j}$. Consequently, later candidate routes tend to be new routes and thereby candidate routes are diversified.

Our analog IC routing has additional complexities compared to usual routing works. To follow the Assumption 1 in Section 2.2, we need to move Steiner nodes out of active area after the net decomposition. There are multiple options for such moving and therefore we keep multiple candidate Steiner nodes. This is illustrated by an example in Figure 3.2.

Conventional two-pin net routing is often done by A* search for short wirelength and avoiding congestion provided that routing edge cost is proportional to congestion.

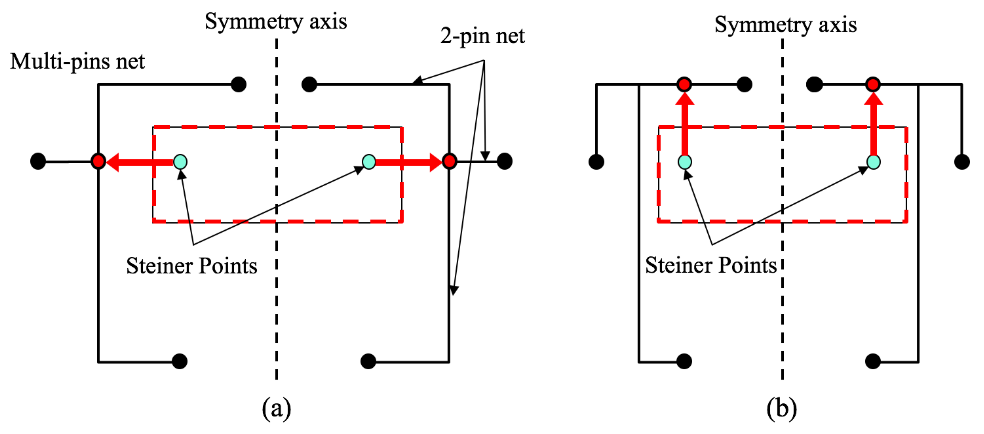

Figure 3.2: Steiner nodes need to be moved out of active area to follow Assumption 1: (a) against the symmetry axis; (b) along the symmetry axis.

In analog IC routing, we also need to restrict the number of bends (or vias). Thus, we modify the A* search by minimizing $\alpha w(p) + \beta b(p)$ mentioned in Section 2.4, where $w(p)$ and $b(p)$ are the total edge cost and number of bends along path p on the routing grid. Two constants α and β are determined for the trade-off of edge cost and number of bends. In practice, we choose $\beta > \alpha$ to emphasize more on bend minimization considering that the routing grid G is a fine-grained grid.

The pseudo code for the candidate generation is provided in Algorithm 2.

3.2 Candidate Refinement and Annotation

The candidate routes obtained as described in Section 3.1 are not immediately ready for the ILP to use. The candidate routes may not include routes that satisfy symmetry or topology-matching constraints for a pair of nets. Routes from different nets need to be annotated if they satisfy wirelength-matching, orientation-matching, or bend-matching constraints. Annotation is also needed to inform ILP if two routes conflict with each other.

In our work, we not only implement five matching constraints, symmetry, topology-

Algorithm 2 Algorithm of candidate generation

```
Input: A set of two-pin nets $\hat{\mathcal{N}} = \{\hat{N}_1, \hat{N}_2, ...\}$
    Routing grid $G = (V, E)$, parameter $K$
Output: A set of candidate routes $P_i$ for each $\hat{N}_i \in \hat{\mathcal{N}}$
 1: for each $\hat{N}_i \in \hat{\mathcal{N}}$ do
 2:     $P_i \leftarrow \emptyset$
 3:     $weight(e) \leftarrow$ length of $e$, $\forall e \in E$
 4:     $j \leftarrow 1$
 5:     while $j \leq K$ do
 6:         $p_{i,j} \leftarrow$ bend-aware A* search for $\hat{N}_i$ on $G$
 7:         $P_i \leftarrow P_i \cup \{p_{i,j}\}$
 8:         for each edge $e \in p_{i,j}$ do
 9:             increase $weight(e)$ by $\delta$
10:         end for
11:         $j \leftarrow j + 1$
12:     end while
13: end for
```

matching, wirelength-matching, orientation-matching, and bend-matching, but also implement wire width constraints and a special case to neglect conflict in two nets called omission. For the constraints input file, we can specified the matched constraints or the width constraints by using command shown in Table 3.1. The Constraints 1-5 are the multi-net constraints we mentioned in Section 2.3 and the commands are very intuitive to designate two nets should be symmetry or matching by the constraints type. The Constraint 6 is a single-net constraint to assign the width for any particular net. For every net without specifying the width, the default value is 1. It is the minimum wire width for the circuit and it is also the unit of grid. Therefore, the value argument specifies how many times wider than minimum width. The last one, Constraint 7, is omit constraint and it is designed for neglecting routing space conflict in two nets. Sometimes a net might be symmetry matched to the other net in local but it is not symmetry in globle. It is helpful to have the omission constraints to handle the situation like that.

Table 3.1: Commands for the constraints.

	Type of Constraints	Command
1	Symmetry	sym net1 net2
2	Toplogy-matching	topology net1 net2
3	Bend-matching	bend net1 net2
4	Orientation-matching	ori net1 net2
5	Wirelength-matching51	length net1 net2
6	Width	width net value
7	Omission	omit net1 net2

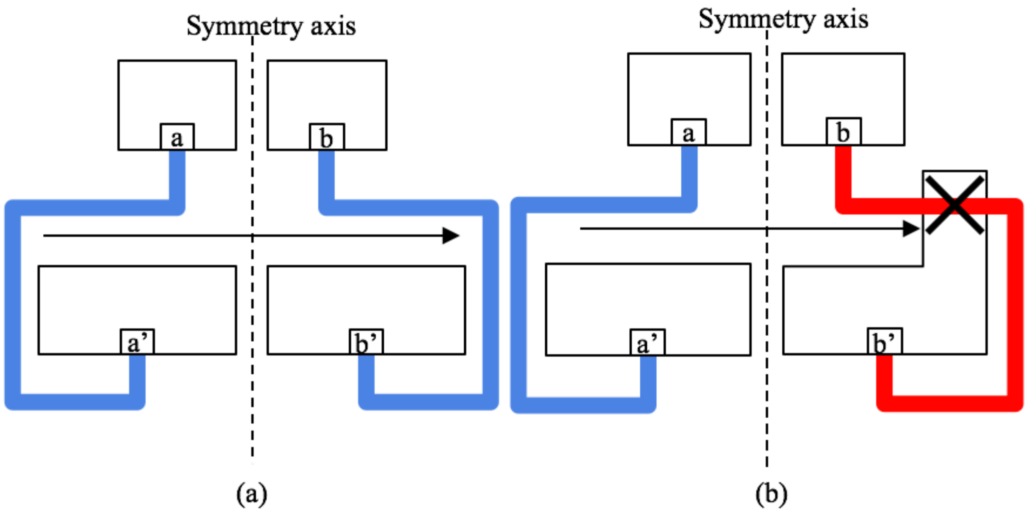

Figure 3.3: Mirroring to obtain symmetric route: (a) successful mirroring; (b) failed mirroring.

Consider a pair of nets N_i and N_j with a symmetry constraint. For each candidate route $p_{i,a}$ of net N_i, we mirror it to obtain a symmetric route $p_{j,a}$. If the new route $p_{j,a}$ is legitimate, e.g., no overlap with active area, it is added into the set of candidate routes for net N_j. We annotate $p_{i,a}$ and $p_{j,a}$ as a pair of potential feasible routes for net N_i and N_j. Examples of this procedure are shown in Figure 3.3. Of course, we can also get such pairs by mirroring routes originally obtained from net N_j. If a candidate route $p_{i,a}$ cannot find its symmetric counterpart for net N_j, then $p_{i,a}$ is removed from the set of candidate routes for N_i. Refining candidate routes for satisfying topology-matching constraints is very similar except that the new route is obtained by shifting besides mirroring. One example of obtaining route for topology-matching is provided in Figure 3.4.

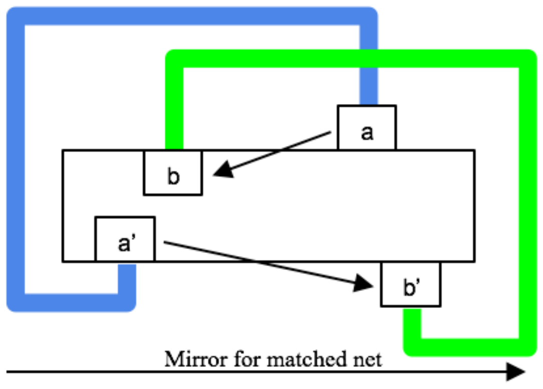

Figure 3.4: Mirroring and shifting to obtain topology-matched route.

To satisfy the other analog specific constraints, we group candidate routes satisfying one such constraint together. For example, consider nets N_i and N_j with a bend-matching constraint. We group candidate routes $P_i^k = \{p_{i,k1}, p_{i,k2}, ...\}$ and $P_j^k = \{p_{j,k1}, p_{j,k2}, ...\}$ with same number of bends k together and add an annotation to them. Then in the ILP step, we would know that one candidate route from P_i^k and another candidate route from P_j^k can form a pair of routes for N_i and N_j satisfying the bend-matching constraint. Figure 3.5 shows one example of candidate routes satisfying the orientation-matching constraint.

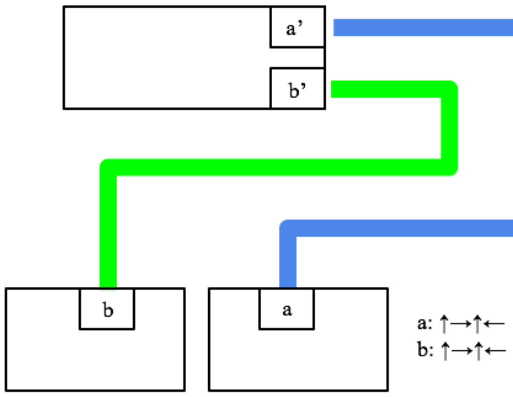

Figure 3.5: Grouping candidate routes with orientation-matching constraint.

The last step before ILP is annotating conflicting pairs of candidate routes. A pair of candidate routes conflict with each other if simultaneous selection of them results in either routing capacity violation or crosstalk (coupling capacitance) violation. The annotations would inform the ILP not to simultaneously select conflicting candidate routes.

3.3 Integer Linear Programming (ILP) Formulation

We use a decision variable $x_{i,j} \in \{0, 1\}$ to tell if to select the candidate route $p_{i,j}$ for two-pin net N_i. Each candidate route $p_{i,j}$ is characterized by its wirelength $l_{i,j}$ and number of bends $b_{i,j}$. Then, the objective of the ILP is described by

$$\min \sum_{\forall i, \forall j} (\alpha \cdot l_{i,j} \cdot x_{i,j} + \beta \cdot b_{i,j} \cdot x_{i,j}) \tag{3.1}$$

where α and β are constant weighting factors.

One fundamental constraint is that only one candidate route is selected for each net. This constraint is represented as

$$\sum_{\forall j} x_{i,j} = 1, \qquad \forall N_i \in \hat{\mathcal{N}} \tag{3.2}$$

For each pair of nets N_i and N_j with a symmetry or a topology-matching constraint, if candidate routes $p_{i,a}$ and $p_{j,b}$ satisfy the constraint, then we require

$$x_{i,a} = x_{j,b} \tag{3.3}$$

which means either both candidate routes are selected or none of them is selected.

The other analog specific constraints, including bend, orientation and wirelength-matching, are formulated according to the grouping described in Section 3.2. We show the ILP formulation for them using a bend-matching constraint as an example. If two nets N_i and N_j has a bend-matching constraint, the annotations described in Section 3.2 can identify two groups of candidate routes $P_i^k = \{p_{i,k_1}, p_{i,k_2}, ...\}$ and $P_j^k = \{p_{j,k_1}, p_{j,k_2}, ...\}$ with the same number of bends $k = 0, 1, 2....$ Then, we require

$$x_{i,k_1} + x_{i,k_2} + ... = x_{j,k_1} + x_{j,k_2} + ..., \qquad k = 0, 1, 2, ... \tag{3.4}$$

According to Equation (3.2), at most one variable at the left-hand side (or right-hand side) above can take value of 1. The above equation ensures that there are only two possible outcomes. One is that one candidate route is selected from P_i^k while another candidate route is simultaneously selected from P_j^k. The other is that none candidate route is selected from either P_i^k or P_j^k.

If two candidate routes $p_{i,a}$ and $p_{j,b}$ conflict with each other, they can never be simultaneously selected. This constraint is

$$x_{i,a} + x_{j,b} \leq 1, \quad \forall \text{ conflicting } n_{i,a} \text{ and } n_{j,b} \tag{3.5}$$

4. EXPERIMENTAL RESULTS

We implemented our analog IC routing method and the algorithm of [6] in C++ programming language. For each net, 20 candidate routes are generated. Empirically, we chooe $\alpha = 1$ and $\beta = 100$ for the objective function defined in (3.1). All the experiments were performed on 4x AMD Opteron 6176 12-core 2.3HGz 6MB L2/L3 Cache Linux workstation with 128GB memory. GUROBI6.0.0 [12] is the library used to solve the ILP problem.

Two op-amps OP1 and OP2 are designed with IBM 0.18μm technology library environment, and up to three metal layers were used to route the circuits. OP1 is a single output amplifier and OP2 is a differential output amplifier. Statistics of the two testcases are shown in Table 4.1. After automatically generating layout routing, we used Calibre nmDRC to check the design rules and nmLVS to verify the layout versus schematic.

Table 4.1: Statistics of testcase OP1 and OP2

Circuit	Number of components				Specification		
	Transistor	Capacitor	Resistor	Total	DC Gain (dB)	Unity Gain Bandwidth (MHz)	Phase Margin (°)
OP1	9	8	0	17	60.53	319.51	59.54
OP2	51	9	4	64	83.26	856.73	52.08

In the first part of the experiment, we compared our method with manual design and the previous work [6]. The manual layouts were performed by an experienced analog designer. We ran post-layout simulation to measure various performance parameters. The performance comparisons in terms of gain, unity gain bandwidth and phase margin are summarized in Table 4.2 and Table 4.3. For OP1, both our approach and [6] produce the result that is nearly the same as manual design. For OP2, our results are the same or slightly better than the manual design while the method of [6] failed to generate a feasible routing solution.

Table 4.2: Comparison of DC gain, unity gain bandwidth, and phase margin between [6] and our approach for OP1.

Parameters	Schematic	Manual	[6]	Oours
DC Gain (dB)	60.53	61.48	61.47	61.47
Unity Gain Bandwidth (MHz)	319.51	312.86	312.83	312.84
Phase Margin (°)	59.54	55.45	53.65	53.65

Table 4.3: Comparison of DC gain, unity gain bandwidth, and phase margin between [6] and our approach for OP2.

Parameters	Schematic	Manual	[6]	Oours
DC Gain (dB)	83.26	84.23	N/A	84.23
Unity Gain Bandwidth (MHz)	856.73	782.74	N/A	789.23
Phase Margin (°)	52.08	52.76	N/A	53.52

Table 4.4 shows the time spent on generating results by these different methods.

Our approach is hundreds of times faster than the manual design. Overall, the advantage of our approach over [2] and manual design is very clear.

To have a complete view of circuit OP2 performance, the frequency domain gain and phase margin curves are shown in Figure 4.1 and Figure 4.2, respectively.

Table 4.4: Comparison of runtime between [6] and our approach.

Circuit	Runtime		
	Manual	[6]	Ours
OP1	1 hour	<1 sec.	<1 sec.
OP2	10 hours	N/A	70.18 sec.

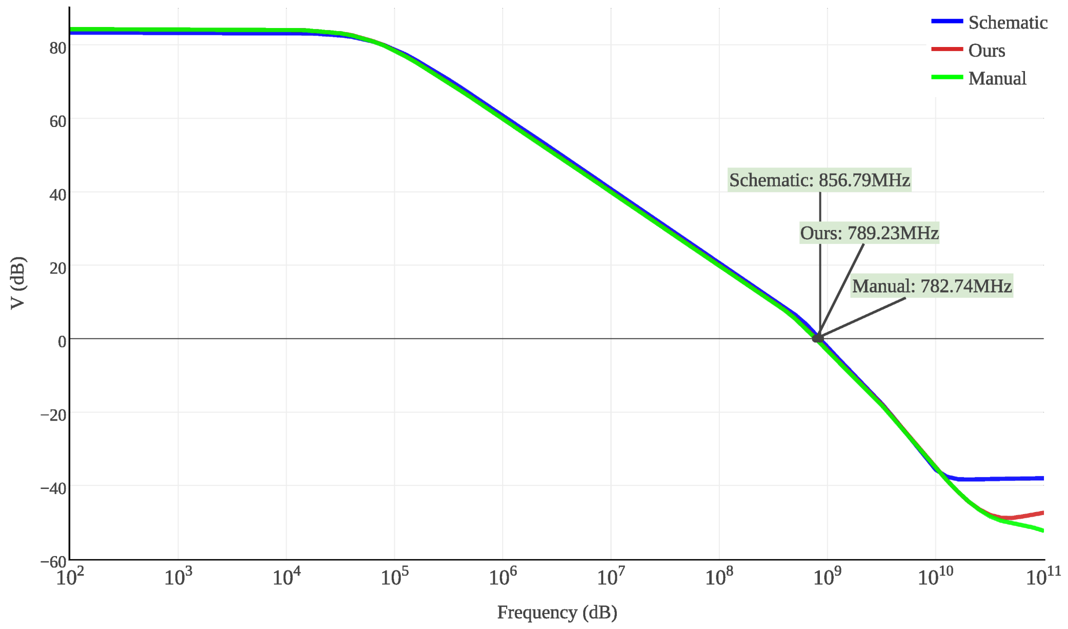

Figure 4.1: Comparison of DC gain and unity gain bandwidth for OP2.

Ideally, the parasitic from layout should not degrade performance of schematic design where parasitic is neglected. In this regard, the performance from our method is almost identical to the schematic design in most part (the part mattering in practice) of the spectrum. The picture of our OP2 layout is displayed in Figure 4.3.

In the second part of the experiment, we analyzed the runtime of our method and the impact from the number of candidate routes for OP2. The results of this part are shown in Table 4.5 and Table 4.6. One can see that the runtime is dominated

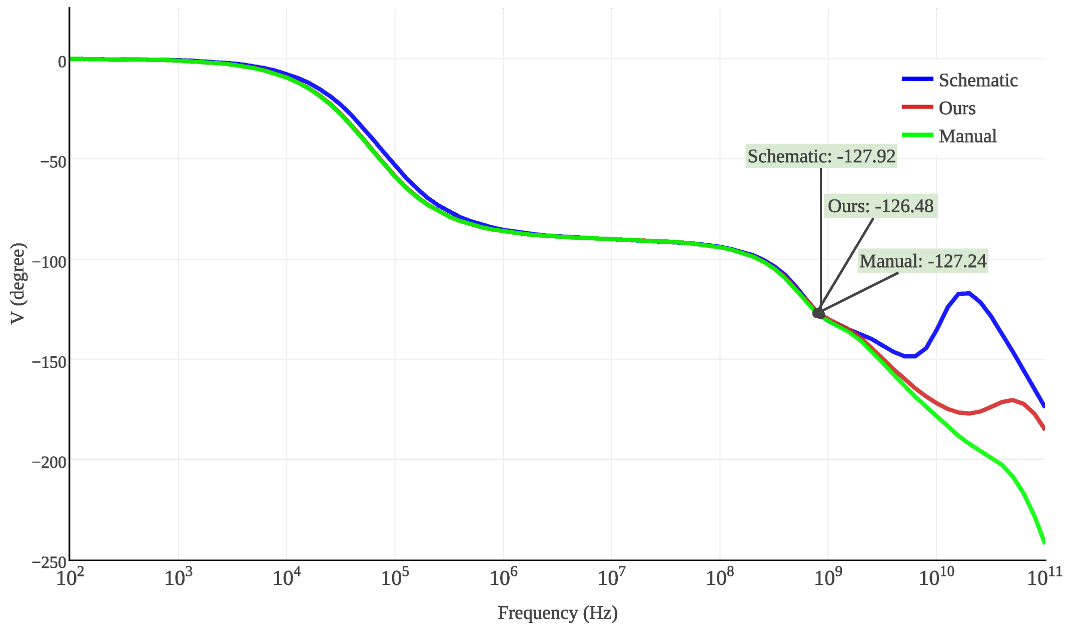

Figure 4.2: Comparison of phase margin for OP2.

Table 4.5: Runtime analysis of the number of candidate routes for OP2.

Maximum Generated Number	Candidate Routes Generation		Candidate Refinement and Annotation		ILP Solving		Total	Objective Function
	Time(s)	%	Time(s)	%	Time(s)	%	Time(s)	
5	14.17	98.99	0.13	0.93	0.01	0.08	14.32	N/A
10	29.99	98.79	0.33	1.07	0.04	0.14	30.36	4884
20	69.03	98.36	0.99	1.41	0.16	0.23	70.18	4871
50	233.86	95.32	5.69	2.32	5.78	2.35	245.33	4866
100	683.63	95.05	22.61	3.14	13.00	1.81	719.23	4856

by the candidate generation. This is because the routing graph $G = (V, E)$ here is a 395x639 grid, which reaches the details of routing tracks and has about 0.25 millions of nodes. The numbers of ILP variables and constraints are in the order of $1K$ and

Table 4.6: Number of variables and constraints impact of the number of candidate routes for OP2.

Maximum Generated Number	Number of ILP Variables	Number of ILP Constraints	Objective Function
5	226	698	N/A
10	398	2290	4884
20	780	9276	4871
50	2023	68417	4866
100	4019	296353	4856

$10K$, respectively. The GUROBI solver can solve ILP of such sizes very quickly. The constructive ILP [1] has one variable for each pair of routing edge $e \in E$ and signal net. Thus, it would entail several millions of variables, which constitute a big challenge to ILP solvers.

When the number of candidate routes per net is 5, the ILP cannot find a feasible routing solution. As the number increases from 10 to 100, the value of the objective function defined by (3.1) monotonically decreases, as indicated by the rightmost column of Table 4.5. However, the pace of the decrease is slow. When the number of candidate routes increases from 20 to 50, the objective function value decreases by only 0.1%. Therefore, 20 candidate routes per net is a reasonable trade-off between runtime and solution quality.

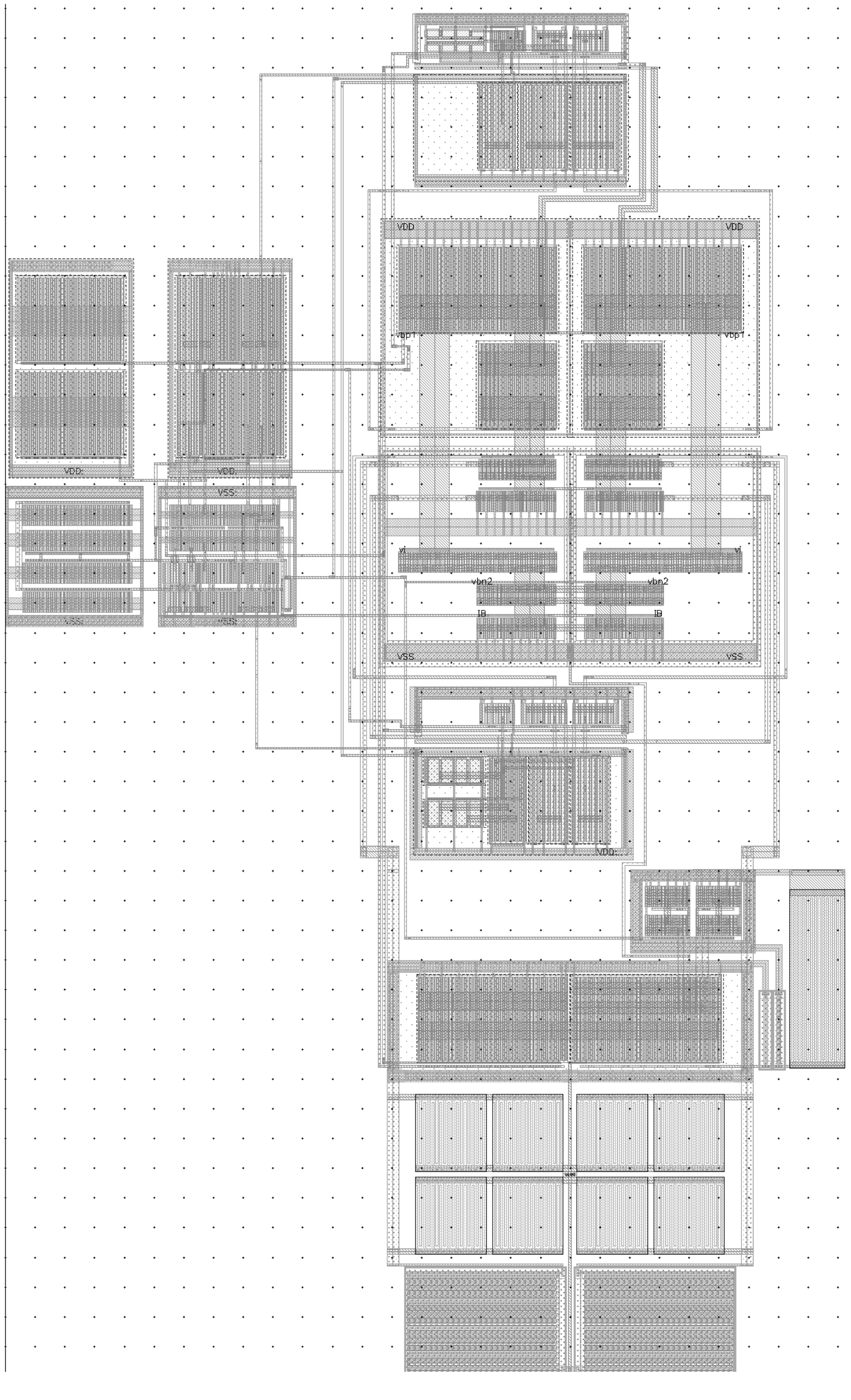

Figure 4.3: Layout of OP2.

5. CONCLUSION

In this paper, we present an efficient two-stage approach to analog IC routing. The first stage is to generate a set of candidate routes for each net satisfying single-net constraints. In the second stage, an ILP is focused on selecting candidate route for each net such that the spatial contention and multi-net constraints are solved. Such approach tends to produce high quality solutions with reasonable runtime. Experimental results show that our routing results lead to performance close to manual design but is orders of magnitude faster. It also outperforms a previous work of automatic analog IC routing.

REFERENCES

[1] H.-C. Ou, H.-C. Chang-Chien, and Y.-W. Chang. Non-uniform multilevel analog routing with matching constraints. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 549–554, June 2012.

[2] U. Choudhury and A. Sangiovanni-Vincentelli. Constraint-based channel routing for analog and mixed analog/digital circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 12(4):497–510, Apr 1993.

[3] M.M. Ozdal and R.F. Hentschke. Exact route matching algorithms for analog and mixed signal integrated circuits. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 231–238, Nov 2009.

[4] H. Yao, Y. Cai, and Q. Gao. Lemar: A novel length matching routing algorithm for analog and mixed signal circuits. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 157–162, Jan 2012.

[5] H. E. Graeb, editor. *Analog layout synthesis: a survey of topological approaches.* Springer Publishing Company, 2010.

[6] L. Xiao, E. F. Y. Young, X. He, and K. P. Pun. Practical placement and routing techniques for analog circuit designs. In *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, pages 675–679, Nov 2010.

[7] D. J. Garrod, R. A. Rutenbar, and L. R. Carley. Automatic layout of custom analog cells in anagram. In *Computer-Aided Design, 1988. ICCAD-88. Digest*

of Technical Papers., IEEE International Conference on, pages 544–547, Nov 1988.

[8] K. Sajid, J. D. Carothers, J. J. Rodriguez, and W. T. Holman. Global routing methodology for analog and mixed-signal layout. In *ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International*, pages 442–446, 2001.

[9] J. Lienig, G. Jerke, and T. Adler. Electromigration avoidance in analog circuits: two methodologies for current-driven routing. In *Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings.*, pages 372–378, 2002.

[10] C. Chu and Y.-C. Wong. Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(1):70–83, Jan 2008.

[11] H. He, G. Yang, and J. Hu. Algorithms for power-efficient qos in application specific nocs. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, pages 165–170, 2014.

[12] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.