Evaluation of Trunk Routing Heuristics Applied to Detailed Routing

Eder Monteiro, Mateus Fogaça, Henrique Placido, Jucemar Monteiro, Isadora Oliveira
André Oliveira, Marcelo Johann and Ricardo Reis PGMicro/PPGC - Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
{emrmonteiro, mpfogaca, hplacido, jucemar.monteiro, isoliveira, asoliveira, johann, reis}@inf.ufrgs.br

Abstract—**This work presents an evaluation of two trunk routing heuristics: Single Trunk Steiner Tree (STST) and Refined Single Trunk Tree (RST-T), applied to detailed routing. We compare the efficiency of both algorithms to generate the final topology of the nets, considering wirelength, number of segments and vias and the number of nets that each algorithm can route. Experimental results show that RST-T can route about 5% more nets than STST for the test cases without obstacles in superior metal layers, and 6% fewer nets than STST when the benchmarks have blockages in these layers. In addition, we did another experiment that consists of ignoring all the blockages present in the superior metal layers when routing the nets of the benchmarks. RST-T can route about 5% more nets than STST in this experiment. Also, RST-T generates topologies with 1-4% better wirelength and produces on average 2% fewer vias in both cases. However, these differences are too small, and they are not significant on this evaluation.**

Index Terms—**Steiner trees, Detailed routing, Physical Design, EDA, Microelectronics**

I. Introduction

The steady decrease of feature sizes in the integrate circuits allows the integration of more functional blocks in a single die which nowadays can hold hundreds of millions of transistors. However, the complexity of modern systems also raises the cost of the design. Companies have to afford a high number of engineers, tools and enablements. The time to market is also critical to the success of the final product. Effective techniques, able to reduce the time to market, and therefore its costs, are desirable.

The design of an integrated circuit can be divided into two main phases: synthesis and implementation. Synthesis takes a high-level description of the design, traditionally written using languages such as *Verilog* or *HDL*, and produces an optimized gate-level netlist. The implementation is responsible for taking the gate-level netlist and produced the layout. *Placement* and *routing* are the two main steps of the implementation – Placement finds the location of each gate in the netlist and routing is responsible for tracing the interconnections among the gates using wires and vias. Due to the its high complexity, routing is divided into *global* and *detailed routing*. The global routing divides the circuit area into a regular grid and finds a global path which is composed of cells of the grid for each net. The detailed routing traces the routing of each net inside the path determined during the global routing. Since the grid is usually coarse, the complexity of the detailed routing is significantly reduced. Even tough, routing is the most timing consuming task in the integrated circuits design flow and usually a bottleneck for meeting the frequency constraints.

Early stages of the implementation flow, such as placement, usually need to estimate the actual wire length of the interconnections and therefore apply *routing prediction heuristics*. Two well-known routing prediction heuristics are the single-trunk steiner tree (STST) [1] and the Refined Single Trunk Tree (RST-T) [2]. These heuristics have an linear runtime and produce a routing topology very similar to the actual routing and therefore are *fast* and *efficient*. In this work we explore the application of these heuristics to detailed routing. We describe how these heuristics are adapted and measure their efficiency in a comprehensive set of experiments.

The contributions of this paper may be summarized as follows:

- We adapt the fast and effective routing prediction heuristics STST and RST-T to detailed routing;
- We perform comprehensive experiments in the ISPD18 contests set of benchmarks, which provides testcases for modern challenges, with circuit utilization rates from 47% to 100% and obstacles
- We show that RST-T can route circuits almost 26% of the nets of a circuit with 290386 gates

The remaining of this paper is organized as follows: Section II presents a review of concepts related to this work and a formal definition for the problem of detailed routing while Section III presents the related works. Section IV explains the methodology to compare the performance of both trunk routing algorithms, the definition of these algorithms and the adaptations that were made for the use of them in detailed routing. The test cases and results are discussed in Section V and Section VI draws the final remarks.

II. Preliminaries

This section presents a review of the concepts related to the content of this paper. First, we discuss the definition of routing prediction and some methods applied to this problem. We present the routing flow in VLSI applications and the problem definition of detailed routing and its objectives.

A. Routing Prediction

When predicting the routing at initial stages of the design flow is important to have estimations of the physical parameters of interconnect nets (wirelength, source to sink distance,

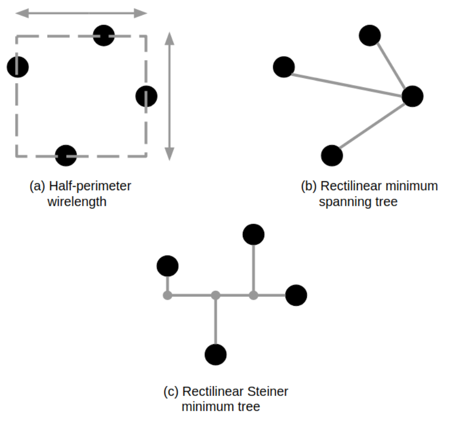

(a) Half-perimeter wirelength
(b) Rectilinear minimum spanning tree
(c) Rectilinear Steiner minimum tree

Fig. 1. Three methodologies to estimate routing

etc.) and its parasitics (the interconnections capacitance and resistance) to guide design tools to optimize the layout, particularly at floorplanning and placement. Since routing is one of the last stages of the design flow, to assure a good result, it is desired all previous steps to be aware of the physical design [3]. However, the development of accurate and fast estimation tools is a complex task. For example, an actual routing algorithm could be used during placement iterations to evaluate the quality of the solution, but the runtime of this practice would be prohibitive [4].

There are many approaches to estimate routing in literature. The half-perimeter wire length (HPWL), depicted in Figure 1(a), is often used during placement as a metric to estimate the total net length, computed as the half-perimeter of the minimal bounding box enclosing all the pins of the net [5]. This methodology presents optimal results for nets with up to 3 pins, and have a fast runtime, but produces imprecise wirelength estimates for nets with more than three pins.

A more precise estimate is given by finding the shortest tree topology that connects all the pins of a net. This method is named rectilinear minimum spanning tree (RMST) [6], and is shown in Figure 1(b). In graph theory, the RMST of a set of points in the plane is a minimum spanning tree of that set, where the weight of the edge between two points is the rectilinear distance between those two points.

A technique that usually produces the result closest to the final topology of a routed net is the rectilinear Steiner minimum tree (RSMT) [4], illustrated in Figure 1(c). It extends the idea of the RMST by adding extra nodes in the net, usually called Steiner nodes. With these extra nodes, it is guaranteed that the generated topology will consist only of horizontal and vertical wires, close to the final routing of a net. The RSMT is an NP-Complete problem [7], that is, there is no known algorithm capable of resolving it in polynomial time. Therefore, heuristics and approximate algorithms are required, since algorithms with high runtime are computationally too expensive to be used in VLSI-design applications.

B. Routing Flow

Routing is the most consuming step in the design flow, and therefore, is divided in literature into two phases [8], [9]: global and detailed routing. Also, some works recommend an intermediate step between these two phases, called track assignment [10], as shown in Figure 2. Next, we detail each phase of the routing flow.

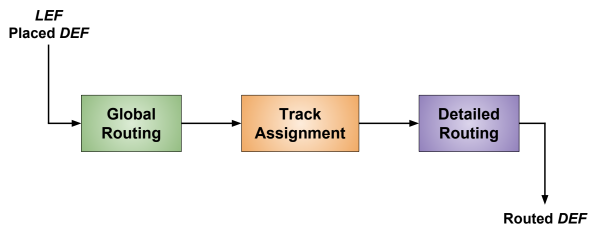


Fig. 2. Routing Flow

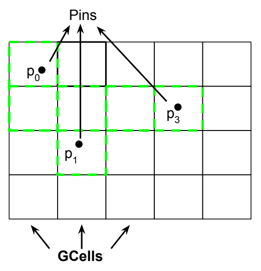
Global Routing. At this step, the circuit is divided into rows and columns equally spaced, creating a regular grid where each cell is called global routing cell (GCell). A net can have pins in different GCells, and the global routing defines a routing topology composed of GCells, enclosing all pins of the net. The set of GCells of a net is also called global guide. Figure 3(a) shows an example of global routing for a 3-pin net.

Track Assignment. The circuit has vertical and horizontal tracks, spaced according to the design rules. These tracks compose the routing grid. It is desirable that the segments of the routing of a net be assigned to these tracks. By doing this, we avoid design rule violations like minimum spacing violations. The track assignment step is responsible for this task. As shown in Figure 3(b), it divides the global routing of each net into horizontal and vertical segments, which are called tracks. Specific tracks are assigned for the connection of each pin of a net by the use of heuristics of optimization, aiming congestion minimization for example. Tracks are only assigned for connections that have at least the height or width of a GCell in length. In Figure 3(b), the pins *p0* and *p1* will have connections that occupy less than one GCell, so there is no track assignment for them.

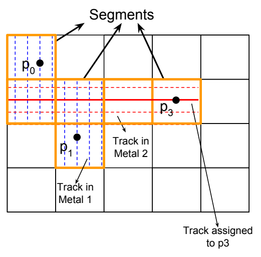
Detailed Routing. The final step of the routing flow, detailed routing is responsible for the generation of the final topologies of the nets of a circuit, using metal wires and vias. Ideally, detailed routing must respect the global guides, that is, the wires and vias of the routing must be inside the guides. Furthermore, design rules like minimum spacing must be respected. The use of track assignment makes compliance with these requirements easier. Detailed routing will only handle with the metal layers and vias and the local connections (that is, connections that occupy less than one GCell). Figure 3(c) shows the final topology for a 3-pin net.

C. Problem Definition

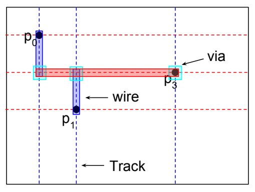
The input for the detailed routing may be defined as a hypergraph G(V, E), where V is the set of vertices representing pins, and E is the set of hyperedges representing the nets. The placement gives information about pin locations, and the global routing assigns nets to specific GCells and metal layers. All technology rules have to be considered, such as the metal



(a) Global Routing



(b) Track Assignment



(c) Detailed Routing

Fig. 3. Example of the global routing, track assignment and detailed routing for a 3-pin net

pitches, minimum metal area, end of the line and cut spacing to ensure manufacturability. Besides a set of obstacles may be specified for the routing.

III. Related Works

In this section, we present a review of some works on the prediction of RSMTs and detailed routing.

A. Prediction of RSMTs

The fastest implementation for the prediction of RSMTs, which generates optimal RSMTs is the GeoSteiner package [11], [12]. However, its high computational cost makes its use impractical in VLSI applications.

Due to their complexity, several heuristics were proposed to find RSMTs with a lower runtime. An example of such heuristics is the single-trunk Steiner tree (STST) [1], which construct a tree by connecting each pin to a trunk that goes horizontally or vertically through the median position of all pins. Although producing near-optimal trees for nets up to 5 pins, the wirelength of STST is far from optimal for nets with more than ten pins. Therefore, its application is limited to low-degree nets, that is, nets with a small number of pins (up to 5 pins).

In [2], Chen proposes an improvement of STST technique, called Refined Single Trunk Tree (RST-T). Instead of connecting each pin directly to the trunk, it is verified if it is shorter to connect the pin to the trunk or the nearest segment that already connects a pin to the tree. By adding this extra option to connect the pins of a net, RST-T guarantees to generate RSMT for nets up to 5 pins, and near-optimal solution for nets up to 10 pins. Besides that, RST-T guarantees more stable topologies. That is, by changing the position of a pin in a net, the new RSMT will have a similar topology of the previous RSMT.

Chu *et al.* [4] presents an O($n\log n$) algorithm with high accuracy for nets up to 9 pins, called fast lookup table estimation (FLUTE). It classifies the nets of the circuit by their pins relative positions and provides, for each of these groups, a pre-computed set of possible optimal routing topologies. The authors showed that FLUTE produces optimum results for nets up to 9 pins. More large nets are decomposed into small nets in a net-breaking technique so that smaller nets can use the table, at the cost of precision. However, since most of the nets in the circuits have a degree equal to 2 or 3 pins, the total wirelength error is less than 2%.

Huang *et al.* [13] studies the obstacle-avoiding rectilinear Steiner minimum trees (OARSMT) problem, which consists of finding an RSMT for a given set of pins in the presence of obstacles. In modern VLSI designs, obstacles block the device layer and a fraction of metal layers. Is possible to route wires on top of obstacles, but a long wire routed over an obstacle may cause signal integrity problems because buffers cannot be placed on top of any obstacle. To avoid this problem, the authors of this study imposes slew constraints on the interconnects that are routed over an obstacle. Also, they propose an algorithm to find an optimal solution to this problem, showing that the tree structures over obstacles with slew constraints will follow simple forms. This algorithm achieves 800 times speedup and reduces nearly 5% routing resources on average when compared with the state-of-the-art optimal OARSMT algorithm.

B. Detailed Routing

Detailed routing has extensive literature that can be traced back to the use of maze search algorithms such as Lee's [14] and A* [15]. More recently, Zhang et al. [16] propose a flow encouraging the use of regular routing patterns to achieve correct-by-construction solutions with higher routability. The authors show the efficiency of techniques such as trunk routing and global segment assignment to route most of the netlist. An implementation of the A* is used to route the remaining nets. In [17], Jia et al. propose an ILP formulation based on a technique called commodity flow. The work is extended in [18] by Han et al. to cope with advanced design rules present in sub-20nm technologies. ILP-based techniques reach the optimal solution but do not scale well, so they cannot be applied to regions bigger than a GCell [18].

IV. Methodology

In this work, we present an evaluation of STST and RST-T when applied to detailed routing. Both algorithms are formerly for routing prediction. In this section, we show the techniques we apply to complete the detailed routing of the test cases used for the experiments, focusing on the extension of the implemented trunk routing techniques for detailed routing.

We present the routing flow implemented by Team UFRGS-Brazil for the ISPD 2018 Contest on Initial Detailed Routing [19]. First, we have selected a set of local nets that will be routed by the trunk routing algorithms. These local nets

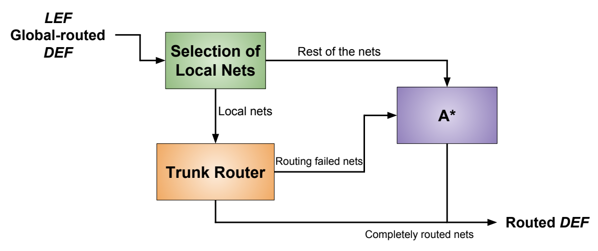
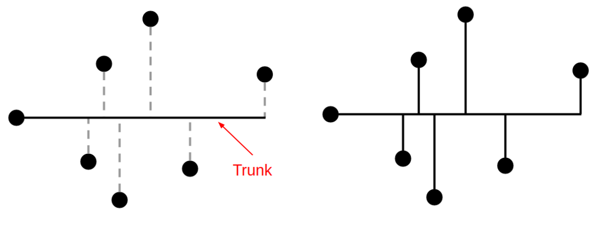


Fig. 4. Team UFRGS-Brazil routing flow for the ISPD 2018 Contest on Initial Detailed Routing



(a) Trunk construction (b) Final topology

Fig. 5. Example of a tree generated by STST

are defined as the nets whose summation of bounding-box height and width are smaller than 25 row-heights and with layer assignment in Metal 3 or below. For the rest of the nets, we apply the A* algorithm, completing the routing of the circuit. Figure 4 shows this flow.

A. Implemented Heuristics

Now, we present the definitions of both trunk routing heuristics, STST and RST-T, followed by the adaptations made to the algorithms for their applications in detailed routing.

Single Trunk Steiner Tree (STST).

STST is a Steiner heuristic used for many years in wire estimation, due to its linear computing time. This approach has several advantages as a routing tree generator for routing estimation: (i) it is easy to construct; (ii) The source to sink distance is smaller or equal to the length of the net's bounding box; (iii) the topology is stable, and (iv) the wirelength is close to RSMT for nets up to 5 pins.

On the other hand, it presents a considerable drawback: the total wirelength of STST grows at a higher rate than RSMT. The growth rate of STST is O(n), while for RSMT this growing is in the rate of $O(\sqrt{n})$, where n is the number of pins in a net. Therefore, for nets with a large number of pins, STST gives a routing tree with longer wirelength than RSMT.

STST presents a simple way to construct a tree: create a segment called trunk that goes horizontally or vertically through the median position of all pins and connect each pin to this segment. Figure 5(a) shows a horizontal trunk created between the pins, and Figure 5(b) shows the final topology generated by STST.

Refined Single Trunk Tree (RST-T). RST-T is a routing algorithm based on the STST and proposes a procedure

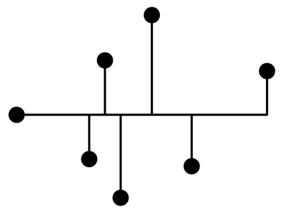
Algorithm 1: RST-T

```
   Data: A set of points $P = (x_i, y_i)$
   Result: A rectilinear Steiner tree with all points in P
           connected
1  begin
2      $y_{mid} \leftarrow$ median of all Yi
3      $x_{min} \leftarrow$ Min{ $x_i$—($x_i$, y) $\in P$ }
4      $x_{max} \leftarrow$ Max{$x_i$—($x_i$, y) $\in P$ }
5      Construct a horizontal trunk from $(X_{min}, Y_{mid})$ to
         $(X_{max}, Y_{mid})$
6      U $\leftarrow$ {(x, y)|(x, y) $\in P$ and y > $y_{mid}$ }
7      L $\leftarrow$ {(x, y)|(x, y) $\in P$ and y < $y_{mid}$ }
8      $P_{Uini} \leftarrow$ (x, y), where (x, y) $\in U$ and (x, y) is closer
         to the middle of the trunk
9      $P_{Lini} \leftarrow$ (x, y), where (x, y) $\in L$ and (x, y) is closer
         to the middle of the trunk
10     Connect $P_{Uini}$ and $P_{Lini}$ to the trunk
11     Remove $P_{Uini}$ and $P_{Lini}$ from U and L, respectively
12     for All pins in U do
13         Connect pin to the neighboring stem or to the
             trunk, depending on which one is shorter
14     end
15     for All pins in L do
16         Connect pin to the neighboring stem or to the
             trunk, depending on which one is shorter
17     end
18     Build an RST-T with vertical trunk in a similar way
         to 1 - 16
19     return Tree with shorter wirelength from horizontal
         and vertical tree
20 end
```

to fix the problem of total wirelength grow rate of STST. Figures 6(a) and 6(b) show the difference of the topologies generated by STST and RSTT. Algorithm 1 describes this procedure. The algorithm receives as input a set of points *P*, representing the pins of a net, and returns a tree that connects all the points in *P*. In lines 1-4 the median, maximum and minimum of the points are computed. In line 5, the horizontal trunk is constructed through all pins, at the mean *y* position computed before. Lines 6 and 7 separate the pins into two groups: one group contain the pins that are above the trunk, and the other contains the pins that are below the trunk. In lines 8 and 9 the first two pins to be connected are defined, and line 10 completes the connections. From line 11 to line 16, the rest of the pins are connected to the tree. Line 18 repeats the process but building a vertical trunk.

The expected wirelength for RST-T is $O(\sqrt{n})$, where n is the number of pins. When n approaches infinity, the expected distance between a pin to this neighboring is of order $n^{-0.5}$. Therefore, the average cost to connect one pin to the tree is $O(n^{-0.5})$.

Also, for nets with no more than four pins, RST-T is an RSMT [2]. For the nets with three pins, RSMT must have, at most, one Steiner point, and for the nets with four pins,


(a) Topology with STST

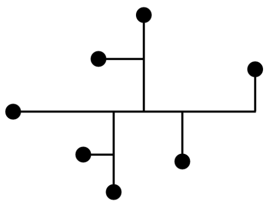

(b) Topology with RSTT

Fig. 6. Difference between a topology generated by STST and RSTT

RSMT must have up to 2 Steiner points. RST-T can find the best Steiner points for these cases. According to experimental results, RST-T in practice is also an optimal RSMT for 5-pins nets. For nets up to 10 pins, RST-T gives trees with wirelength 6% larger than RSMT.

B. Implementation Adaptations

Both STST and RST-T are heuristics for routing prediction. In this work, we extend these heuristics to perform the actual routing of VLSI circuits. Each segment of the topology of a net is assigned to a specific metal layer, respecting the preferred direction of the layers. Moreover, the final topologies must respect as much as possible the global guides of the nets.

An essential procedure of detailed routing is the insertion of vias. Vias are connections between two different metal layers. This connection can be between segments in different layers, or between a segment and a pin. Also, the number of vias required to connect two different layers is equivalent to the difference between these two layers. For example, if we want to connect a segment in Metal 3 to a pin in Metal 1, we will need two vias; one via to connect Metal 3 to Metal 2 and one via to connect Metal 2 to Metal 1.

We also modified certain aspects of both algorithms in order to simplify the generation of the final topologies of the nets. Instead of creating two topologies with different trunk directions, we define that the trunk will be routed in the topmost layer available for a net. Hence, the direction of the trunk will be the preferred direction of that layer. This strategy aims to reduce the number of vias used to route a net since the segments that will connect the pins to the trunk will be closer to the first metal layer.

In our implementation of RST-T, we added a cost function to compute the best layer to construct the other segments of a net topology. First, we defined weights for the violation of the routing guide, the violation of layer preferred direction and the insertion of vias, according to the evaluation metrics of the ISPD 2018 Contest [19]. For each one of the available layers, we compute the cost of the insertion of a segment based on the length of that segment and the defined weights. Eq. (1) shows our cost function.

$$C(s,l) = (s.length \times 0.5) \times (3^{l.gV} \times 3^{l.pdV}) + 2 \times numVias, \quad (1)$$

where s is the segment, l is the current layer, gV is a boolean indicating routing guide violation, pdV is a boolean indicating

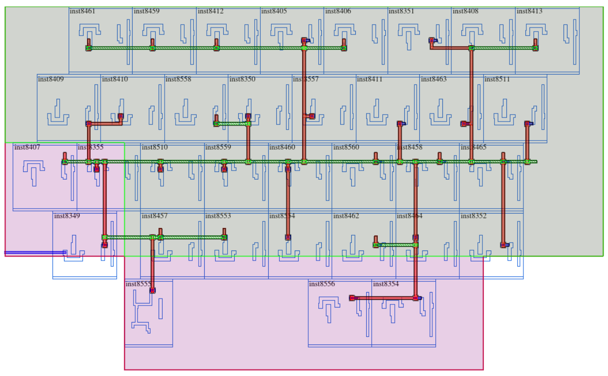


Fig. 7. Detailed Routing with RST-T

preferred direction violation, and $numVias$ is the number of vias necessary to connect this segment to the topology.

After computing the cost of insert a segment for each layer, we choose the layer with the lowest cost to construct the segment. If the attempt of constructing this segment fails, the next layer with the lowest cost is chosen, and so on. Figure 7 shows a net routed by RST-T, where we can see some segments out of their guides or violating the preferred direction of a metal layer.

V. Results

A. Experimental Setup

The algorithms were implemented using C++ programming language, and the experiments were performed on an Intel Core i7-470K CPU @ 4.00GHz×8 machine with 32 GB of memory.

We conduct our experiments in the benchmarks specified in Table I from ISPD 18 [19]. The smaller benchmark is *test1*, with 3153 nets and 8879 cells, while *test10* is the largest benchmark and has 182000 nets and 290386 cells. All test cases have 1211 I/O pins, except for *test1*, that does not have any I/O pin.

Macro blocks and cell density vary on each benchmark, which helps to evaluate the robustness of our routing techniques. As shown in Figure 8(a), *test2* has no blockages and lower density of cells. Figure 8(b) shows *test5*, which has blockages and a higher density of cells while Figure 8(c) shows *test10*, which has no blockages, but has the highest cell density between the other test cases.

The benchmarks *test1*, *test2*, and *test3* have the technology node of 45nm, while the other benchmarks have the technology node of 32nm. Also, the die size of the test cases varies between 3.8x10^{-2}mm^2 and 1.81mm^2. Moreover, we can divide the test cases into three groups. One group consists of the benchmarks that have Power/Ground pins and obstacles in standard cells only in Metal 1, as shown in Figure 9(a), where the filled blue dots represent the obstacles and the empty blue dots represent the available positions for segments and vias. These benchmarks are *test1*, *test2* and *test3*. The benchmarks of the second group are *test4*, *test5*, *test6*, and *test7*, and have obstacles and Power/Ground in Metal 1 and Metal 2. In Figure 9(b) the filled red dots represent the

TABLE I
DETAILS OF THE TEST CASES

Benchmark	#inst	#nets	#I/Os	#Blks	#Layers	Floorplan			Tech. (nm)
						A. Ratio (height/width)	Die Area (10^{-1}mm^2)	Util	
test1	8879	3153	N/A	N/A	9	0.95	0.4	0.85	45
test2	35913	36834	1211	N/A	9	0.87	3.7	0.47	45
test3	35973	36700	1211	4	9	0.71	6.9	0.65	45
test4	72094	72401	1211	4	9	0.68	5.4	0.81	32
test5	71954	72394	1211	8	9	0.99	8,5	0.92	32
test6	107919	107701	1211	N/A	9	0,62	4.5	0.98	32
test7	179865	179863	1211	16	9	1.02	18	0.89	32
test8	191987	179863	1211	16	9	1.02	18	0.89	32
test9	192911	178857	1211	N/A	9	0.86	7.1	0.90	32
test10	290386	182000	1211	N/A	9	0.86	7.9	1.00	32

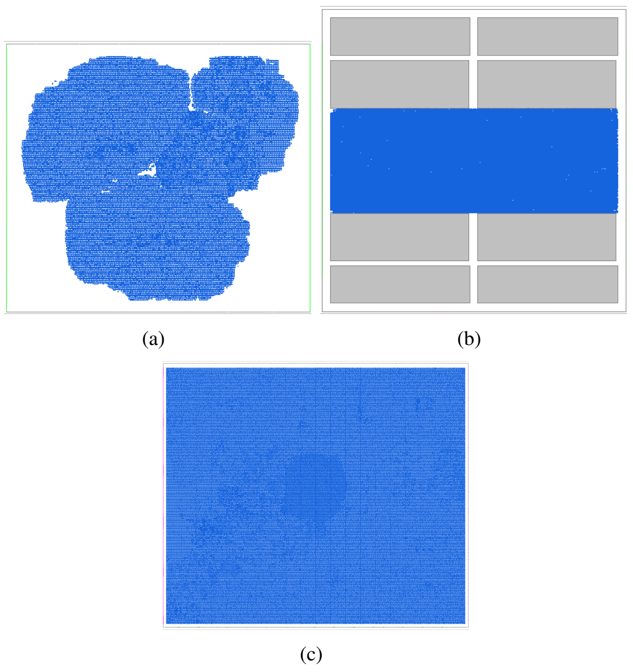


Fig. 8. Floorplanning of benchmarks test2 (a), test5 (b) and test10 (c)

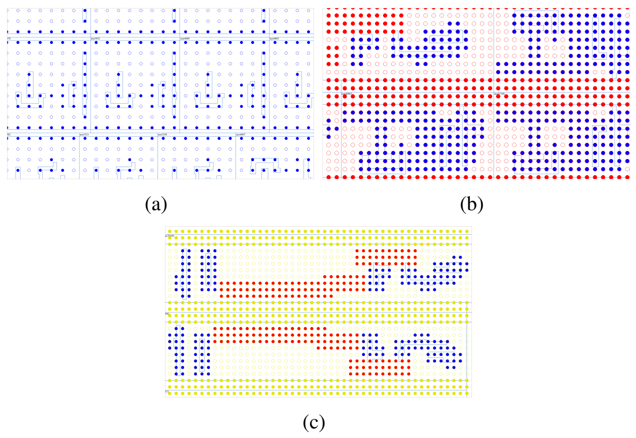


Fig. 9. Obstacles and Power/Ground pins in different metal layers

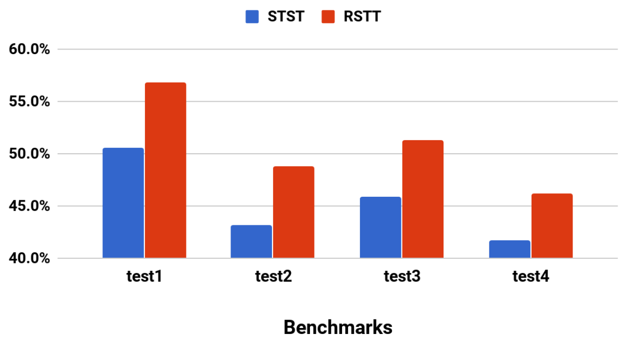


Fig. 10. Percentage of routed nets with each algorithm

obstacles in Metal 2. The third group is composed of the other benchmarks, which have Metal 2 to Metal 3 obstacles and Metal 2 to Metal 4 Power/Ground pins as blockages. Figure 9(c) shows an example of that obstacles and blockages, with the filled yellow dots representing them.

B. *Comparison between STST and RST-T*

The results are divided into two analysis: (1) the percentage of routed nets; (2) quality of the results, where we analyze wirelength, the number of vias and the number of segments. The first experiment consists of running STST and RST-T for the same set of local nets, following the recipe described in Section IV, and comparing the number of nets that each algorithm can successfully route. In the second experiment, we ran STST and RST-T for the nets that both algorithms managed to route then we compare the final wirelength, the number of vias and the number of wires for each algorithm.

Routed nets: The number of nets routed by RST-T is higher than the nets routed by STST for the benchmarks that do not have Power/Ground pins in Metal 2 or superior metal layers. RST-T routed between 4% and 6% more nets than STST in the benchmarks *test1*, *test2*, *test3* and *test4*, as shown in Figure 10. For the other benchmarks, STST routed up to 6% more nets than RST-T, which was an unexpected result. Figure 11 shows these results.

Quality of results: RST-T provides a better result for wirelength than STST, with a difference between 1% and 3%. RST-T also produces between 1% and 4% fewer vias. For the first four benchmarks, STST produces fewer wires than RST-T, but there is no significant difference between the number of wires generated by each algorithm for the other test cases.

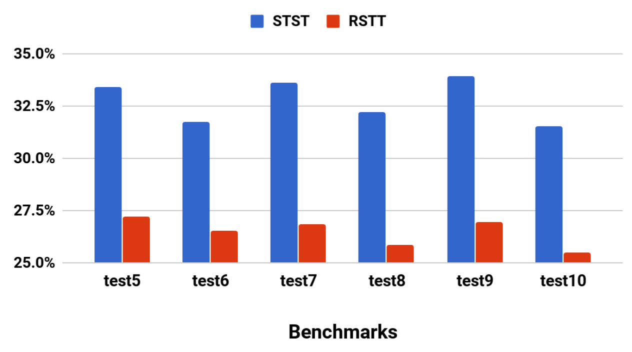


Fig. 11. Percentage of routed nets with each algorithm: unexpected behavior

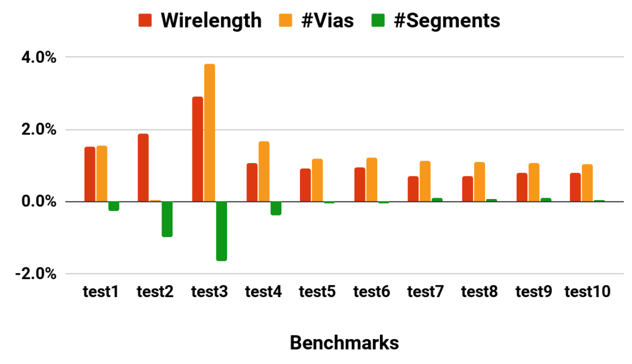


Fig. 12. Results regarding wirelength, number of wires, number of vias in STST and RST-T

These differences are shown in Figure 12, and are computed as $1 - (RST\text{-}T/STST)$.

Removal of macro blocks and obstacles in superior layers: In addition to the results presented, we did another experiment to understand the unexpected behavior of RST-T when the benchmarks have Power/Ground pins in superior metal layers. This experiment consists in ignoring all the blockages present in the superior layers when routing the nets of the benchmarks. In this experiment, we show that it was the presence of these blockages that affected the routing efficiency of the RST-T. Figure 13 shows that RST-T routed about 5% more nets than STST without the blockages. Besides that, STST also achieved an efficiency improvement when we ignored the blockages, routing between 1% and 3% more nets. RST-T continues to produce topologies with better wire length than STST, with 1-3% less wire length but has produced 3% more vias and 4% more segments, as shown in Figure 14.

VI. Conclusions and Future Works

In this work, we present an analysis of two routing prediction algorithms, STST and RST-T, being used in detailed routing. A set of experiments was conducted to compare the efficiency of both algorithms to generate the final topology of the nets using metal wires and vias. We performed these experiments upon a proposed implementation of STST and RST-T. The results show that RST-T can route between 4-6% more nets than STST in most of our test cases, with 1% to 4% less wirelength and use 2% to 5% fewer vias. However, for the test cases with blockages in Metal 2, STST can route 6% more nets than RST-T. We also show that the presence of macro blocks and obstacles in superior metal layers affected the routing efficiency of both algorithms, specially RST-T. When ignoring these obstacles, RST-T routed about 5% more nets than STST, and 13% more nets than before the removal of the obstacles.

As future work, we intend to extend our experiments to get a better understanding of the behavior of STST with the test cases with blockages in Metal2 and propose a heuristic for RST-T to avoid blockages in Metal2 and superior metal layers.

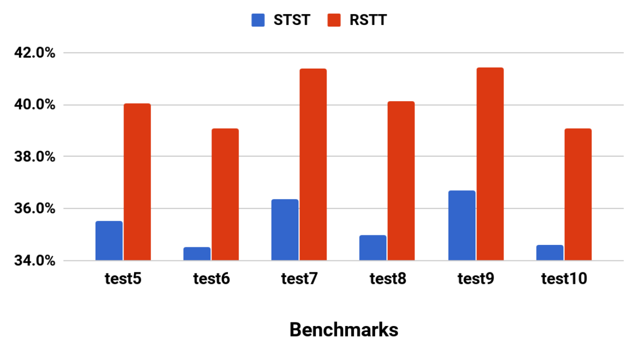


Fig. 13. Percentage of routed nets after removing blockages in Metal 2 or superior metal layers

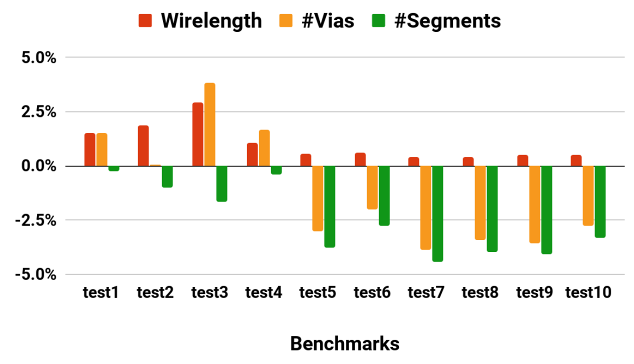


Fig. 14. Results regarding wirelength, number of wires, number of vias in STST and RST-T without blockages in Metal 2 or superior metal layers

References

[1] J. Soukup, "Circuit layout," *Proceedings of the IEEE*, vol. 69, no. 10, pp. 1281–1304, Oct 1981.

[2] H. Chen, C. Qiao, F. Zhou, and C.-K. Cheng, "Refined single trunk tree: A rectilinear steiner tree generator for interconnect prediction," in *Proceedings of the 2002 International Workshop on System-level Interconnect Prediction*, ser. SLIP '02. New York, NY, USA: ACM, 2002, pp. 85–89. [Online]. Available: http://doi.acm.org/10.1145/505348.505366

[3] M. Tatsuoka, R. Watanabe, T. Otsuka, T. Hasegawa, Q. Zhu, R. Okamura, X. Li, and T. Takabatake, "Physically aware high level synthesis design flow," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 162:1–162:6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744893

[4] C. Chu and Y. C. Wong, “Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 70–83, Jan 2008.

[5] A. Kennings and I. Markov, “Analytical minimization of half-perimeter wirelength,” in *Proceedings 2000. Design Automation Conference. (IEEE Cat. No.00CH37106)*, June 2000, pp. 179–184.

[6] H. Zhou, N. Shenoy, and W. Nicholls, “Efficient minimum spanning tree construction without delaunay triangulation [vlsi cad],” in *Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455)*, 2001, pp. 192–197.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[8] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[9] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, Eds., *Electronic Design Automation: Synthesis, Verification, and Test*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[10] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, “Track assignment: a desirable intermediate step between global routing and detailed routing,” in *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002.*, Nov 2002, pp. 59–66.

[11] D. M. Warme, P. Winter, and M. Zachariasen, *Exact Algorithms for Plane Steiner Tree Problems: A Computational Study*. Boston, MA: Springer US, 2000, pp. 81–116. [Online]. Available: https://doi.org/10.1007/978-1-4757-3171-2_6

[12] “*GeoSteiner—Software for Computing Steiner Trees*.” [Online]. Available: http://www.geosteiner.com/

[13] T. Huang and E. F. Y. Young, “Construction of rectilinear steiner minimum trees with slew constraints over obstacles,” in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2012, pp. 144–151.

[14] C. Y. Lee, “An algorithm for path connections and its applications,” *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sept 1961.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, “Correction to ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths”,” *SIGART Bull.*, no. 37, pp. 28–29, Dec. 1972. [Online]. Available: http://doi.acm.org/10.1145/1056777.1056779

[16] Y. Zhang and C. Chu, “Regularroute: An efficient detailed router with regular routing patterns,” in *Proceedings of the 2011 International Symposium on Physical Design*, ser. ISPD ’11. New York, NY, USA: ACM, 2011, pp. 45–52. [Online]. Available: http://doi.acm.org/10.1145/1960397.1960410

[17] X. Jia, Y. Cai, Q. Zhou, G. Chen, Z. Li, and Z. Li, “Mcfroute: A detailed router based on multi-commodity flow method,” in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD ’14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 397–404. [Online]. Available: http://dl.acm.org/citation.cfm?id=2691365.2691446

[18] K. Han, A. B. Kahng, and H. Lee, “Evaluation of beol design rule impacts using an optimal ilp-based detailed router,” in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[19] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, “Ispd 2018 initial detailed routing contest and benchmarks,” in *Proceedings of the 2018 International Symposium on Physical Design*. ACM, 2018, pp. 140–143.