Towards a Reference Place and Route Flow for Academic Research

Tiago Augusto Fontana[1], Renan Netto[1], Sheiny Fabre Almeida[1], Erfan Aghaeekiasaraee[2], Laleh Behjat[2], and José Luís Güntzel[1]

[1] Dept. of Informatics and Statistics (INE/PPGCC), Federal University of Santa Catarina, Brazil
[2] Dept. of Electrical and Software Engineering, University of Calgary, Canada
email: tiagoaugustofontana@gmail.com

Abstract— **Due to the complexity of contemporary VLSI circuits, physical synthesis has become a crucial step for achieving design closure. The placement of cells directly impacts the routing solution. For example, a region with a high cell density can lead to pin access issues in detailed routing. Therefore, small inefficiencies in the placement solution can be boosted during routing, which has a negative impact on design quality and convergence. Unfortunately, most academic research works evaluate their results only at the target step without considering the complete place and route flow. In this work, we experimentally explored different flows built up from academic placers and routers to find which one leads to the best overall results so that researchers can use them as reference. In order to evaluate those flows, we used the ISPD 2018 and ISPD 2019 Contest benchmarks, which are the most realistic academic benchmarks available with placement and routing information. Considering the evaluator reports, no combination of tools achieved the best result for all circuits. Nevertheless, the flow Contest placement + CUGR + TritonRoute achieved the best results in seventeen out of twenty benchmark circuits.**

Index Terms— **Electronic Design Automation; Physical Synthesis; Placement; Routing**

I. Introduction

The design of modern digital circuits faces increasing difficulty in meeting timing and power constraints. New devices require high performance, but users expect the battery to last. In this context, physical design became a vital step to enable those power, performance, and area (PPA) constraints [1].

The length of circuit interconnections plays an essential role in the circuit performance and power [2]. Therefore, efficient placement and routing algorithms are necessary to find good solutions for circuits with millions of cells. During placement, locations are determined trying to minimize mainly wirelength while being aware of other metrics such as congestion. The routing step connects all nets, refining the routing solution in terms of wirelength and number of vias while satisfying complex manufacturing constraints.

Due to the complexity of technology constraints and the size of modern circuits, it is infeasible to find an optimal solution for the placement and routing problems in acceptable runtime. Therefore, we need to rely on smart strategies and heuristics to find the best solution in a reasonable runtime.

The most common strategy to solve the placement problem is to start with an analytical solution to find an initial global placement for the whole circuit [3, 4, 5, 6, 7, 8, 9]. Due to the complexity of modern circuits, global placement is simplified to allow overlaps between cells. Then, it is followed by a legalization step that removes all those overlaps while keeping cells within the circuit borders. Finally, several optimization techniques can be used to improve the solution further.

The routing step is also subdivided into smaller steps due to its complexity. First, a global routing step finds the paths to connect all nets without actually assigning wires to the routing tracks. Then, it is followed by a detailed routing step that refines the global routing solution assigning it to specific routing tracks while preserving quality and satisfying technology constraints [1]. Global routing usually starts with pattern routing or maze routing to find a path for each interconnection segment. However, depending on the order the nets are routed, this may lead to over-congested regions. Thus, a subsequent rip-up and reroute step is executed to fix this congestion without degrading too much the wirelength [10, 11]. A similar strategy is adopted for detailed routing, running a path search algorithm to connect the wire segments while respecting all design rules imposed by the technology node. However, sometimes it is impossible to route all the nets following the solution found by global routing. When that happens, the detailed routing algorithm needs to reroute those nets accordingly.

The separation into placement and routing steps is essential to allow handling the huge complexity of contemporary designs. However, these two steps are strongly correlated in such a way that small inefficiencies in the placement solution can be amplified during routing, thus deteriorating design quality and convergence. For example, a region with a high cell density can lead to pin access issues in detailed routing.

Given the aforementioned difficulties, it is possible to deduce that it is a challenge to select which placement and routing tools to use when synthesizing a circuit. This difficulty affects not only the designers that want to rely on academic tools but also the researchers. Particularly, most academic research works evaluate their results only at the target step without considering the complete place and route flows. However, a few recent works propose optimization techniques to promote the collaboration between routing and placement steps (e.g. [12]), and thus, their improvements should be evaluated considering the whole physical synthesis flow.

Since 2016, the IEEE CEDA Design Automation Technical Committee (DATC) is developing a public reference design flow named DATC Robust Design Flow (RDF) [13]. This flow aims to provide a foundation and backplane for academic research in the RTL-to-GDS IC implementation arena. The DATC RDF-2021 version includes the OpenROAD Flow project [14] for the physical synthesis. How-

Digital Object Identifier 10.29292/jics.v17i3.648

ever, the DATC RDF flow just integrates the tools without any detailed analyses about the interaction between them. Searching to fill this gap, in this work, we experimentally explored twelve flows built up from academic placers and routers to find which one can lead to the best results so that researchers can use them as references. In order to evaluate those flows, we used the ISPD 2018 [15] and ISPD 2019 [16] CAD Contest benchmarks, which are the most realistic academic benchmarks available with placement and routing information.

The rest of this paper is structured as follows. Sections II. and III. present, respectively, the state-of-the-art placement and routing academic tools that we used to build the flows evaluated in this work. Then, Section IV. details the experiments we performed to evaluate those flows and the discussions to choose the best flow. Finally, Section V. presents some conclusions and challenges faced by this work.

II. Placement

State-of-the-art placement algorithms are based on the analytical placement approach. This approach uses a mathematical formulation to place all cells in the circuit area [1]. This placement aims to minimize wirelength while keeping density under control. Equation (1) is a typical cost function for placement, where $WL(n)$ represents the wirelength of a net n and $D(x, y)$ is a density penalty applied to the circuit. The parameter λ is used to control the trade-off between wirelength minimization and density penalty. This is typically done in steps, wherein one step wirelength is minimized, and in the next step, cells are spread to reduce density.

$$min(\sum_{n \in N} WL(n)) + \lambda D(x, y) \tag{1}$$

Depending on the way wirelength and density are modeled in Equation (1), placement can be solved in different ways. However, regardless of the model, this formulation still allows overlaps between cells, so the result is not a legal placement. Therefore, a legalization algorithm is used afterward to remove all cell overlaps while keeping cells within the circuit boundaries and properly aligned with the circuit sites and rows. Since the legalization process disturbs the placement, additional local optimization steps are applied to further improve placement without violating the legality constraints.

In this work, we used two placement academic tools as references: Eh?Placer [6] and DREAMPlace [7]. Thus, in this Section, we will further explain these tools.

A. *Eh?Placer*

Figure 1 shows the overall flow of the Eh?Placer engine. Eh?Placer is divided into three main stages: initial computations, region-aware global placement, and technology-driven legalization. The first stage (initial computations) performs some preprocessing that will be useful in the other stages. First, modern circuits have regions with different voltages, and only cells with the same voltage can be placed within a given region. Those are called fence regions, and the initial computations stage starts by identifying which cells belong to each fence region. Also, any cell that is not assigned to

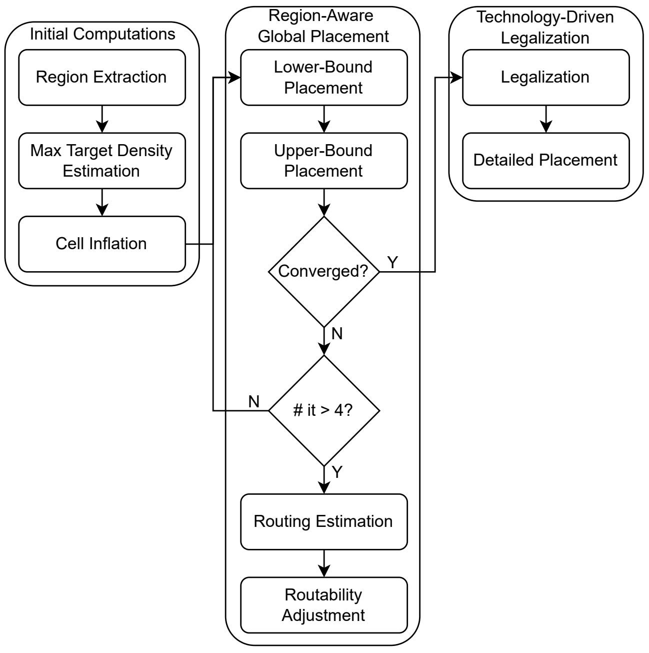

Fig. 1: Flowchart of Eh?Placer
Source: Darav, *et al.* [6]

a specific fence region is assigned to a default region. After that, it can calculate a maximum target density for each region, which will be used later in the placement cost function. Finally, a cell inflation strategy is used to reduce congestion and avoid routability issues in future steps.

The global placement itself is performed in the second stage (region-aware global placement). In order to find a placement solution, Eh?Placer starts with a lower bound wirelength placement solution, which just optimizes wirelength while ignoring density. This will lead to a solution with minimum wirelength. Then, it finds an upper bound placement, which is a fairly non-overlapping solution that respects the fence region constraints while trying to keep the lower bound placement locations as much as possible. This is done using existing techniques from previous works, like lookahead legalization [17, 3] and median improvement [18].

After each iteration of lower and upper bound placements, Eh?Placer checks if the solution has converged. If the gap between both placements is small, that means it has converged and the algorithm can move on to the legalization stage. Otherwise, more iterations are needed. In addition, after the first four iterations, before finding a new lower bound placement Eh?Placer runs a routability adjustment step. This is done by running the NCTUgr global router [19] for routing estimation first, and then this estimation is used to adjust the maximum target density of each region and cell inflation is executed again on the regions, but now with the updated densities.

When global placement converges, the solution still has overlaps. Thus, the third stage legalizes the circuit. Eh?Placer uses an adaptation of Abacus [20] to handle the technology constraints of modern circuits. After the placement is legalized, optimization techniques are applied to some cells to improve the solution. Those techniques can be applied to all cells or only cells that are potentially causing routability issues. Three types of movements inspired by work in [18] can be executed on this step: moving the cell to the median location of its net; creating a local window around the current cell location and finding a better location for it inside this window; switching the location of the cell and one of its closest neighbors. A movement is accepted or rejected based on a cost function that encompasses wirelength, detailed routing violations, and cell density.

B. *DREAMPlace*

The main novelty of DREAMPlace is that it explores the similarity between the analytical placement formulation and neural network training. For example, Equation (2) shows an example of the cost function for neural network training, where $f(x, w)$ is the function the neural network is trying to learn, w are the weights and $R(w)$ is the regularization term. So, $f(x, w)$ can be compared to the wirelength function in placement, while $R(w)$ can be compared to the density penalty. Thereby, the goal of DREAMPlace is to use in placement the same GPU acceleration strategies used to train neural networks.

$$min f(x, w) + \lambda R(w) \tag{2}$$

Figure 2 shows the overall flowchart of DREAMPlace. It starts with an initial random placement, as neural networks start with initial random weights. The initial placement is done by placing all cells in the center of the design and then applying a small random gaussian noise to their locations.

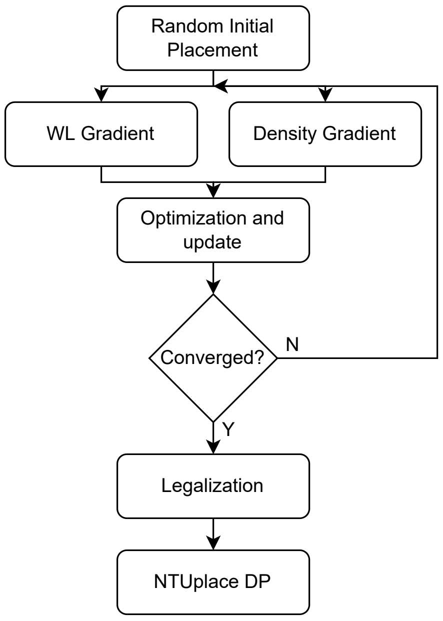

Fig. 2: Flowchart of DREAMPlace
Source: Lin, *et al.* [7]

After assigning the initial locations, DREAMPlace will adjust locations using weighted-average wirelength for the cost function, as proposed in a previous work [9]. The locations are adjusted as follows: (1) a dependency graph is built for the terms that need to be calculated to estimate wirelength; (2) wirelength is estimated in a forward computation of this dependency graph and using the current locations; (3) wirelength gradient is computed; (4) locations are adjusted based on the gradient, in a backward propagation flow. This process is done for both wirelength and density, using the appropriate dependency graph for each. Also, it is done in parallel for each pin.

When the backward propagation strategy converges, cells are legalized. DREAMPlace also uses the Abacus legalization algorithm for that, which is not done using GPU parallelization since the authors observed that legalization was fast enough to execute on CPU. Finally, it uses NTUPlace for further detailed placement [5].

III. Routing

Signal routing, which is the process of interconnecting all logical and sequential elements in a circuit, is an essential step in the physical design flow. This is because as technology scales to deep sub-micron, the delay of the interconnections determines the performance of the circuit [21, 19, 22]. Besides this, the number of tracks in the upper layers is very limited because there are fewer metal tracks available in the upper layers than in the lower ones. This greatly complicates the routing, making it one of the most challenging steps in the physical design flow. Therefore, the quality of a global router deeply influences the timing, power, and density of a chip [23].

Due to this complexity and enormous solution space, the routing process is divided into two steps: Global Routing and Detailed Routing. The Global Routing step allocates routing resources that are used for interconnections, and Detailed Routing is responsible for assigning routes to specific routing tracks within the Global Routing resources [1]. The Global Routing and Detailed Routing steps and their respective academic works considered in this paper are presented in Subsection III.A. and III.B.

A. *Global Routing*

In the Global Routing step, the area of the circuit is partitioned into GCells (Grid Cells), as illustrated in Figure 3a. Each GCell has two attributes: metal layer and capacity. The former associates the GCell with a specific metal layer, whereas the latter defines the number of metal tracks available on that metal layer and routing direction. It is important to note that the routing direction is alternated between metal layers. For instance, if tracks run in the horizontal direction in even metal layers, then, in the odd layers, tracks run in the vertical direction. Thereby, the attribute capacity represents the number of metal tracks available in this direction on this metal layer. In Figure 3a, the green GCell has a capacity of five. The goal of Global Routing is to find, for each net, the set of GCells through which it is possible to establish a connection between pins belonging to the same net, as shown in Figure 3b.

There are two main ways to perform the Global Routing step: using a 3D data structure or using a 2D data structure, followed by a second step called Layer Assignment. The 3D Global Routing determines the metal layers during the path search. For example, Figure 3b shows the 3D Global Routing approach applied to a two-pin interconnection (P1 and P2) presented in Figure 3a. Note that this approach does not require an additional step to determine a layer for each segment. However, due to the high complexity of contemporary

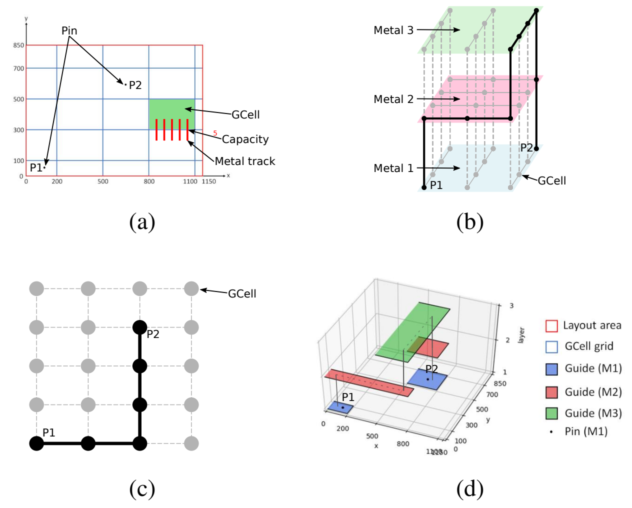

Fig. 3: Example of Global Routing. (a) 2D view of a net with two pins. (b) 3D grid graph where dots represent the vertices and dashed lines represent the edge between vertices. The solid line is a 3D Global Routing. (c) 2D grid graph where dots represent the vertices and dashed lines represent the edges between vertices. The solid line is a 2D Global Routing. (d) View of a valid Global Routing solution.

Source: (a) and (d) adapted from [24]; (b) and (c) the author.

designs, full 3D routing normally results in longer execution times than 2D followed by the Layer Assignment approach [23]. Figure 3c presents the 2D Global Routing approach for the same example shown in Figure 3a. In this scenario, the position of each pin is projected onto the 2D surface. Then, a path-search algorithm is executed to find the set of GCells that completes the interconnection. Finally, a Layer Assignment step is done to determine the layer of each segment. Figure 3d shows the result of Global Routing for the two-pin net represented in Figure 3a. In this work, we will compare two academic open-source Global Routing tools, CUGR [10] and FastRoute [25, 26, 27, 11]. The former employs the 3D approach, whereas the latter relies on the 2D approach.

The CUGR tool comprises three steps: initial routing, multi-level 3D maze routing, and route guide generation, as shown in Figure 4. In the initial routing, each multi-pin net is broken down into a set of two-pin nets in the pattern routing planning step. Then, in the 3D pattern routing step, the FLUTE [28] algorithm is used to generate a rectilinear Steiner minimum tree (RSMT) and a dynamic programming algorithm performs pattern routing and layer assignment simultaneously. After initial routing, the nets with violations are ripped up and go through multiple iterations of rip-up and reroute (RRR) by maze routing. The maze routing algorithm is limited to the bounding box of the net. This approach is to reduce the execution time of performing maze routing on the whole 3D grid graph. The last step in the flow is to generate the guides and insert patches. Patches are extra regions of guides to alleviate some conditions, and they can be of three types: 1) *Pin Region Patching* to improve pin accessibility; 2) *Long Segment Patching* to improve the track assignment; and 3) *Violation Patching* to add more flexibility in regions that may contain violations.

FastRoute is a 2D Global Routing based on a pattern route that first performs a sequence of rip-up and reroutes steps and later maps the 2D solution to 3D by layer assignment.

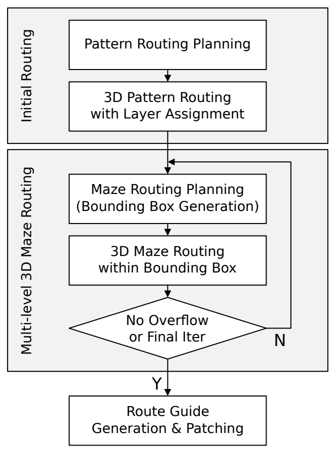
Initial Routing
Pattern Routing Planning
3D Pattern Routing with Layer Assignment
Multi-level 3D Maze Routing
Maze Routing Planning (Bounding Box Generation)
3D Maze Routing within Bounding Box
No Overflow or Final Iter
N
Y
Route Guide Generation & Patching

Fig. 4: CUGR flow.

Source: Liu, Jinwei, *et al.* [10].

The flow of FastRoute is presented in Figure 5. First, a congestion-driven via-aware Steiner topology for each net is constructed, and then a segment-shifting technique is applied. Later, the tree structures are decomposed into 2-pin nets using FLUTE. Then, a pattern routing step using an L-shape and Z-shape initializes the routing solution for each net, and the virtual capacity based on the current routing status is initialized. Next, a loop of rip-up and reroute process is executed until the overflow stops decreasing. Inside the loop, it uses pattern routing and multi-source multi-sink maze routing, and virtual capacity to reduce congestion. The virtual capacity gradually changes the capacity associated with each global edge to divert wire usage from highly congested regions to congestion-free regions. Finally, the 2D solution is extended to a full 3D solution by a spiral layer assignment algorithm.

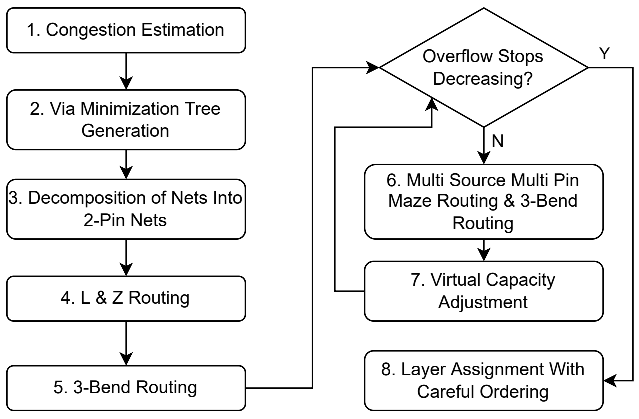
1. Congestion Estimation
2. Via Minimization Tree Generation
3. Decomposition of Nets Into 2-Pin Nets
4. L & Z Routing
5. 3-Bend Routing
Overflow Stops Decreasing?
Y
N
6. Multi Source Multi Pin Maze Routing & 3-Bend Routing
7. Virtual Capacity Adjustment
8. Layer Assignment With Careful Ordering

Fig. 5: FastRoute 4.0 Framework flow.

Source: Xu, Yue, Yanheng Zhang, and Chris Chu [11].

B. *Detailed Routing*

The Detailed Routing step is responsible for determining the exact location (tracks) where each net will be routed. Figure 6 presents this mapping step for a two-pin net that connects cell A and cell B. Generally, Detailed Routing tools receive three inputs: the LEF file containing the technology information, the DEF file with the netlist and cell locations, and the GUIDE file containing the regions, determined by the global routing, in which each net should be routed. Then, the detailed routing will decide the tracks for each net honoring as much as possible the guides from global routing and respecting all design rules imposed by the technology node. Figure 6.b presents a possible detailed routing solution where the dashed lines are the available tracks, and the solid lines are the assigned tracks for the net.

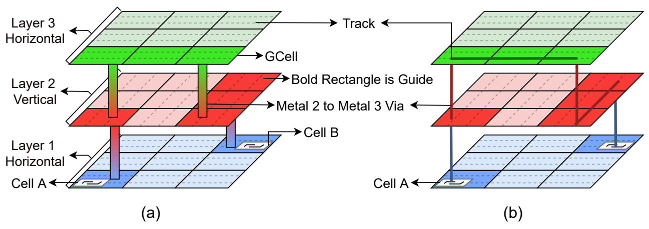

Fig. 6: Global Routing To Detailed Routing

The ISPD 2018 Contest on Initial Detailed Routing [15] revolutionized academic research concerning the detailed routing topic. Before that Contest, only a few works presented end-to-end detailed routing flow. The authors of RegularRoute [29] proposed a correct-by-construction methodology, where the routing tracks are formulated as a Maximum Weight Independent Set (MWIS) problem, and it is solved by a heuristic technique. The authors of MCFRoute [30, 31] proposed a multi-commodity method, where the detailed routing problem with intricate design rules are formulated and solved as an ILP problem. However, no works mentioned above claim a viable solution to the context of real-world IC physical design, i.e., using benchmarks that have technology information and design rules concerning routing.

But after the ISPD 2018 Contest, the academy had access to a set of benchmarks that included metal layer and track information as long as detailed routing challenges. Although after that a few academic works could produce some results [32, 33, 34, 35, 36], most works could not meet design rule violation constraints.

Besides the routing rules, a detailed router should manage the memory usage as the feature size scales down because not only the problem size increases but also the complexity of design becomes increasingly difficult. In order to solve this memory issue, the authors of Dr. CU [33] proposed a routing correct-by-construction design rule satisfaction using a two-level sparse data structure for a 3-D detailed routing grid graph. The key idea of this sparse data structure is to store the information of routed edges utilizing balanced binary search trees (BSTs) and intervals. With this, it is not necessary to store all the graph structure for the whole circuit containing all the nets, tracks, and vias.

When a net is to be routed, a local grid graph is created only for this net, considering the region (guides) provided by the global routing. In this local grid graph, the edge costs are available for conducting the sequential maze routing algorithm. Essentially, nets are routed one after another, whereas previously routed nets are treated as blockages. After routing all nets with possible violations, several rounds of rip-up and reroute (RRR) help to clean them up.

In contemplation of reducing the total runtime of the Dr. CU tool, the authors proposed a bulk synchronous parallel scheme based on [37]. This approach creates batches of nets whose each routing regions do not overlap each other. In comparison to the sequential version, this parallelization technique of Dr. CU reduces the runtime by five to six times using eight threads when compared to the sequential version.

Given the routing challenges introduced by ISPD Contests, an open-source tool called TritonRoute [38, 39] was released aiming to produce a DRC-clean routing solution. This tool routes the circuit following three major steps: Routing Data Preparation, Track Assignment, and Detailed Routing. Figure 7 presents an overview of TritonRoute flow.

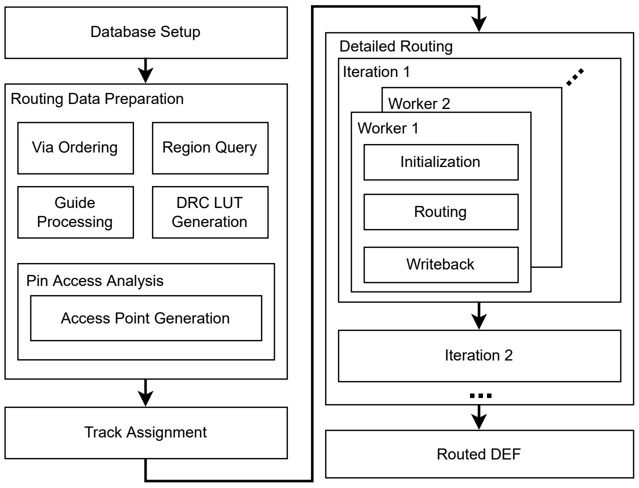

Fig. 7: TritonRoute flow.
Source: Kahng, Andrew B, Wang, Lutong, and Xu, Bangqi [39].

TritonRoute receives as input a Global Routing solution (guides). These guides are pre-processed drawing a center line for each guide along the preferred routing direction. It also creates a spatial data structure for fast shape queries, lookup tables to avoid rule check violations, and performs a pin access analysis to generate at least three access points for each pin. Afterward, a greedy track assignment is performed using panels that follow the preferred routing direction with 50 GCells height. First, it assigns all horizontal layers and then to all vertical layers. The Detailed routing is performed iteratively, the design is partitioned into 7x7 non-overlapping GCell-aligned clips. Each partition is processed in parallel by a worker, a unit that can only modify routing within its partition. Each worker performs a modified version of A* on a non-regular-spaced 3D grid graph.

IV. Experimental Assessment of Place and Route Flows

This section reports the results obtained for different flows of open-source placement and routing tools. It is organized as follows. First, subsection IV.A. explains the methodology adopted to conduct the experiments. Then, subsection B. presents the experimental setup. Subsection C. and D. present the adopted benchmarks and the evaluation process, respectively. Finally, subsection E. brings the results and discussions.

A. *Experimental Methodology*

This section aims to explain the methodology applied to compare and determine the best combination of open-source academic place and route tools. The experimental methodology flow is presented in Figure 8. Its inputs are the benchmarks Library Exchange Format (LEF) and Design Exchange Format (DEF) files. The LEF file contains the physical design characteristics and technology constraints of the cell library. The DEF file hols the location of the physical elements, the netlist, and the routing information.

The first step in the flow is Placement. In this work, we consider three different placements for each benchmark: 1) the original ISPD 2018 and 2019 Placement Contest, and the placements generated by 2) DreamPlace [7], and 3) Eh?Placer [6]. DreamPlace received the best paper award at Design Automation Conference (DAC) 2019. Eh?Placer won second place in ISPD 2015 Blockage-Aware Detailed Routing-Driven Placement Contest [40]. Both DreamPlace and Eh?Placer receive as input the Contest files (LEF and DEF), replace the cells and write the new positions into a new DEF file.

The second step in the flow is Global Routing. CUGR [10] and FastRoute [25, 26, 27, 11] are adopted as academic Global Routing tools in this work. CUGR is the state-of-the-art concerning Global Routing step and outperformed all other contestants in the ICCAD 2019 CAD Contest problem C: LEF/DEF Based Open-Source Global Router [24]. FastRoute is part of the OpenROAD project [41]. Both tools receive LEF and DEF files as input, and output a Guide file. The Guide file is a text file that contains, for each net, a set of rectangles and respective metal layers that interconnect all the pins. These rectangles will be used during the Detailed Routing step to guide the track assignment search space.

The third step in the flow is Detailed Routing. In this work, we consider two academic Detailed Routing tools: Dr. CU [33] and TritonRoute [38]. Initial Detailing Routing has been the subject of two recent Contests, ISPD 2018 [15] and ISPD 2019 [16]. TritonRoute won first place in the ISPD 2018 Contest, and Dr. CU won second and first places in ISPD 2018 and ISPD 2019 Contests, respectively. Therefore, TritonRoute and Dr. CU are state-of-the-art concerning the Detailed Routing step. In the evaluation flow, both receive as input three files: LEF and DEF files from the placement tool and the Guide file from the Global Routing tool. The output of the Detailed Routing step is written into the DEF file.

Finally, the fourth and last step is the evaluation process of all the solutions. The official evaluator binaries from ISPD 2018 and 2019 Placement Contests were used to measure the quality reached by each combination of tools. Note that, considering three possible placements, two global routing tools, and two detailed routing tools, there are twelve different executions for each circuit in the benchmark set. Both evaluators receive as input the LEF, DEF, and Guide files and generate the report file as output. More details about the evaluators are presented in Subsection D.. At the end of this flow, we collected all the results generated by the ISPD evaluators using a python script. This script received as input the report files and generated the Tables III and IV presented in this work.

B. Experimental Setup

The experiments were executed on two different machines:

1. A Linux cluster with 48 cores, 2x Intel® Xeon® Gold 6240R CPU @ 2.40GHz (Cascade Lake, 2021) and 64GB RAM.

2. A Linux workstation with an Intel® Core® i5-4460 CPU running at 3.20 GHz with four cores and 32GB RAM (DDR3 at 1600MHz).

We used these two machines because Eh?Placer and FastRoute cannot be statically compiled, and we do not have root access in the cluster to install the required libraries. In a statically built program, no dynamic linking occurs: all the bindings have been done at compile time. Therefore the executable can run on any machine.

C. Benchmarks

We chose the benchmarks from ISPD 2018 [15] and ISPD 2019 [16] CAD Placement Contests because they are the most recent and complete benchmark sets for placement and routing stages. In ISPD 2018 and ISPD 2019 Contests suites, there are circuits designed with three different technology nodes: 65nm, 45nm, and 32nm. For each of those circuits, there are technology information, macro blockages, IO Cells, and standard cells with complex pin shapes, such as L, Z, U, and others. Nonetheless, there is no power and timing information in these benchmarks. The main characteristics of these circuits are presented in Table I. In this table, Columns 1 to 8 present the circuit names, technology node, number of cells, number of nets, number of I/O pins, number of macro blockages, placement density, and number of routing layers.

Table I.: Main characteristics of ISPD 2018 and ISPD 2019 benchmarks. Columns 1 to 8 bring the name of the circuit, technology node, number of cells, number of nets, number of I/O pins, number of macro blockages, placement density, and number of routing layers.

Circuits	Node (nm)	Cells (K)	Nets (K)	i/o Pin	Macro	Density (%)	Metal Layers
ispd18_test1	45	9	3	0	0	85	9
ispd18_test2	45	36	37	1,211	0	57	9
ispd18_test3	45	36	37	1,211	4	65	9
ispd18_test4	32	72	72	1,211	4	89	9
ispd18_test5	32	72	72	1,211	8	92	9
ispd18_test6	32	108	108	1,211	0	99	9
ispd18_test7	32	180	180	1,211	16	90	9
ispd18_test8	32	192	180	1,211	16	90	9
ispd18_test9	32	193	179	1,211	0	91	9
ispd18_test10	32	290	182	1,211	0	100	9
ispd19_test1	32	9	3	0	0	83	9
ispd19_test2	32	72	72	1,211	4	72	9
ispd19_test3	32	8	9	57	4	84	9
ispd19_test4	65	146	152	4,802	7	21	5
ispd19_test5	65	29	29	360	6	9	5
ispd19_test6	32	180	180	1,211	16	75	9
ispd19_test7	32	360	359	2,216	16	96	9
ispd19_test8	32	540	538	3,221	16	79	9
ispd19_test9	32	899	895	3,221	16	84	9
ispd19_test10	32	899	895	3,221	16	88	9

Notably, the two benchmark sets have circuits with different characteristics. The number of cells and nets vary from 9 to 899K and 3 to 895K, respectively. There are circuits with and without I/O pins and macro blockages. In addition, there are circuits with low (9% - 21%) and high (88% - 100%) placement densities. The 100% placement density in circuit ispd18_test10 is due to the filler cells. There are two

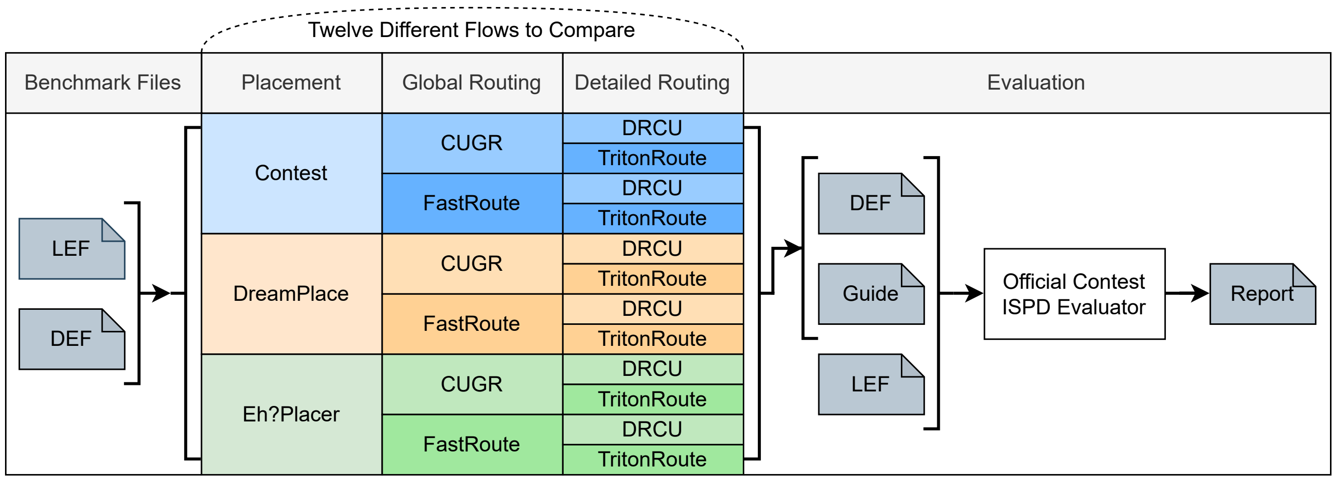

Fig. 8: Place and route flows considered in this work and employed evaluation methodology.

circuits, ispd19_test4 and ispd19_test5, with only 5 metal layers available for the routing. Therefore, we judge that this set of benchmarks can expose different behaviors of the placement and routing engines evaluated in this work.

D. *ISPD evaluator*

The official ISPD 2018 and 2019 Placement Contests evaluators assessed the quality of the routing solutions. The evaluation is based on Routing Metrics, Routing Preference Metrics, and Design Rule Violations. All the metrics and their respective weights are presented in Table II. The final score for a circuit is a weighted sum of all those metrics. A solution is declared invalid if an open net occurs.

Table II.: ISPD Contests evaluation metrics and their respective weights.

Metric	Acronym	Weight
Total length of off-track wires	Wirelength	0.5
Total Number of vias	Vias	2
Total length of off-guide wires	OFGW	0.5
Total number of off-guide vias	OFGV	1
Total length of off-track wires	OFTW	0.5
Total number of off-track vias	OFTV	1
Total length of wrong-way wires	WWW	1
Short metal area	Shorts	500
Number of min-area violations	Min Area	500
Number of spacing violations	Spacing	500

The evaluation flow is: 1) The DEF, Guide, and LEF files are provided to the evaluator. 2) The evaluator performs the design rule and connectivity checking using Cadence® Innovus^{TM} [42]. 3) Innovus generates design rule violation and connectivity reports. 4) The evaluator performs guide and track obedience checking and reads the Innovus reports. 5) The evaluator generates a report table with all the metrics presented in Table II and the raw score as output.

E. *Quality evaluation of place and route flows*

In this section, the experimental results of the twelve different flows are given for the ISPD 2018 and ISPD 2019 benchmarks. Tables III and IV report the breakdown of the results for the detailed routing metrics on each circuit. In both tables, the circuit names (Benchmarks) are displayed in Column 1. Columns 2, 3, and 4 identify the placement, Global Routing, and Detailed Routing tools used in each flow, respectively. Then, Columns 5 and 6 present the routing metrics Wirelength (reported in millimeters) and number of Vias (reported in thousands). For each circuit, the lowest (best) values of wirelength and number of vias are highlighted in bold. Columns 7 to 11 report the routing preference metrics. These metrics are off-guide wirelength (OFGW), off-guide vias (OFGV), off-track wirelength (OFTW), off-track vias (OFTV), and wrong-way wirelength (WWW). Columns 12 to 15 bring the Design Rule Violations (DRV). Then, Columns 16 and 17 present, for each circuit, the final score and the difference to the best score, in percentage. Therefore, in Column 17, 0% identifies the flow that achieved the best score for a given circuit. We ranked all flows based on the final score and the number of short violations. This rank is presented in Column 18 of Tables III and IV. We considered that the solutions with short violations should not outrank the solutions without short violations. In this way, all solutions without short violations, highlighted in bold, come before the solutions that leave short violations.

Before analyzing the results in more detail, it is important to remark that the two flows that use Eh?Placer together with CUGR failed for all benchmarks. For such reason, they do not appear in Tables III and IV. Further investigation is necessary to determine why CUGR reports an error when applied to placements issued by Eh?Placer. In these executions, the binary of CUGR just stops the execution after the “mark fixed metal rtrees..." and “mark fixed metal batch ..." messages, but no errors are reported in the CUGR log file. In addition to these two flows, other flows have produced errors in a few circuits. These errors are reported in the line of the respective flow and can be:

- *Time Out 72h*: we limited the experiment runtime to 72 hours.
- *Out of Memory*: we limited the RAM memory utilization to 64GB.
- *TritonRoute Fail*: this error occurs at the end of the “0th optimization iteration" after printing the

message “post-processing ..." in the log. No error message was reported at that instant. It is possible that open nets occurred, which could not be resolved by TritonRoute.

- *Eh?Placer Fail*: Eh?Placer reported “std::bad_alloc" in the log file.
- *FastRoute Fail*: FastRoute reported the message “Routing congestion too high" in the log file.

Analyzing all results, we can observe that no combination of placement and routing tools led to the best result for all circuits. Considering the top 3 solutions for each circuit, the wirelength varied by 1.7% on average, and the number of vias varied by 4.8% on average. In addition, it is important to observe that at least one of the top 3 best solutions was generated by a flow that begins with the original ISPD Contests placement. This means that original ISPD Contests placements are already well-optimized for routing solutions. We can also conclude that the ISPD 2019 circuits are more difficult to synthetize than the ISPD 2018 ones due to the higher number of failing flows. Particularly, only two flows were able to generate valid solutions for circuit ispd19_test4. The reason why all the other flows failed could be because this circuit has only five metal layers available for routing and was designed with 65 nm technology.

Observing the Design Rule Violations columns of Tables III and IV, for each of the circuits, we notice a huge variance between the flows. Considering only wirelength and number of vias and taking into account only the Global Routing tool, we can observe that the flow that uses CUGR generated the best results for 18 out of 20 circuits We also noticed that FastRoute could lead to the best wirelength only for circuit ispd19_test6 and produced the lowest number of vias for circuit ispd19_test10. For a given combination of Placement and Global Routing tools, it can be observed that TritonRoute leaves much fewer violations than Dr. CU. The best three flows for each circuit are presented in Figure 9. Observe that the flow Contest + CUGR + TritonRoute led to the best score for 17 circuits, the flow Contest + CUGR + Dr. CU led to the best score for 2 circuits, and the flow DreamPlace + CUGR + TritonRoute is the best score only for circuit ispd19_test10. Finally, the experiments we have conducted allowed us to conclude that the best combination of academic open-source tools for placement and routing is Contest + CUGR + TritonRoute.

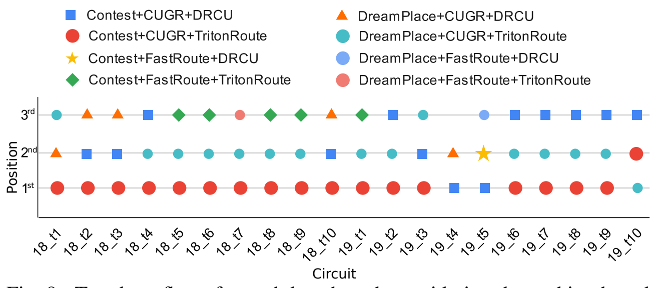

Fig. 9: Top three flows for each benchmark considering the ranking based on score and short violations.

V. Conclusions and Challenges

The process of choosing the best placement and routing tools to synthesize a circuit is a challenging task. Simply replacing one of these tools could generate a drastic impact on the final quality of the results. Therefore, in this work, we explored twelve different combinations of placement and routing academic tools to investigate how they interact with each other. We used the ISPD 2018 Contest [15] and ISPD 2019 Contest [16] benchmarks. Considering the evaluator reports, no combination of tools achieved the best result for all circuits. Nevertheless, the flow Contest placement + CUGR [10] + TritonRoute [39] led to the best results for seventeen out of twenty benchmark circuits.

While preparing and conducting our experiments, we could experience some of the problems that hamper the integration of tools. Some of these tools require specific library versions to work and do not generate static binaries by default. Static binaries do not make references for external libraries, thus, making it possible to execute on a server without additional libraries. This is important when the researcher does not have permission to install and/or modify the server. Many errors are reported during the experimental executions. Further investigation is necessary to remove those errors, especially the integration between Eh?Placer with CUGR.

Acknowledgements

This study was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by the Brazilian Council for Scientific and Technological Development (CNPq) through PQ grant number 316984/2021-3, and by the Natural Sciences and Engineering Research Council of Canada grant number 10015685.

References

[1] 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.

[2] C. Alpert, Z. Li, G. Nam *et al.*, “Placement: hot or not?” in *ICCAD*, 2012, pp. 283–290. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6386624?casa_token=zuJJPa_NDL4AAAAA:UTfDdM9DdZO_E2qHRb31BJDF75akVZW_GMCHsNM6-PnU8Zp_CgBOX2xlW7QPsNk9ntgG_I0Qsy8

[3] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev, “Polar: A high performance mixed-size wirelengh-driven placer with density constraints,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 3, pp. 447–459, 2015. [Online]. Available: https://doi.org/10.1109/TCAD.2015.2394383

[4] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai, and E. F. Y. Young, “Ripple 2.0: High quality routability-driven placement via global router integration,” in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC ’13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2463209.2488922

[5] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, “Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1228–1240, 2008. [Online]. Available: https://doi.org/10.1109/TCAD.2008.923063

[6] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick, and L. Behjat, "Eh?placer: A high-performance modern technology-driven placer," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 3, apr 2016. [Online]. Available: https://doi.org/10.1145/2899381

[7] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317803

[8] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany, and D. Z. Pan, "Abcdplace: Accelerated batch-based concurrent detailed placement on multithreaded cpus and gpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5083–5096, 2020. [Online]. Available: https://doi.org/10.1109/TCAD.2020.2971531

[9] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2018. [Online]. Available: https://doi.org/10.1109/TCAD.2018.2859220

[10] J. Liu, C. Pui, F. Wang, and E. Young, "CUGR: detailed-routability-driven 3d global routing with probabilistic resource model," in *DAC*, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/DAC18072.2020.9218646

[11] Y. Xu, Y. Zhang, and C. Chu, "Fastroute 4.0: Global router with efficient via minimization," in *2009 Asia and South Pacific Design Automation Conference*. IEEE, 2009, pp. 576–581. [Online]. Available: https://doi.org/10.1109/ASPDAC.2009.4796542

[12] E. Aghaeekiasaraee, A. F. Tabrizi, T. A. Fontana, R. Netto, S. F. Almeida, U. Gandhi, J. L. Güntzel, D. Westwick, and L. Behjat, "Cr&p: An efficient co-operation between routing and placement," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 772–777. [Online]. Available: https://doi.org/10.23919/DATE54114.2022.9774530

[13] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, S. Kim, V. N. Kravets, Y.-L. Li, R. Varadarajan, and M. Woo, "Datc rdf-2021: Design flow and beyond iccad special session paper," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICCAD51958.2021.9643553

[14] A. B. Kahng and T. Spyrou, "The openroad project: Unleashing hardware innovation," in *Proc. GOMAC*, 2021. [Online]. Available: https://vlsicad.ucsd.edu/Publications/Conferences/383/c383.pdf

[15] 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. [Online]. Available: https://www.ispd.cc/contests/18/index.htm

[16] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proceedings of the 2019 International Symposium on Physical Design*. ACM, 2019, pp. 147–151. [Online]. Available: https://www.ispd.cc/contests/19/index.htm

[17] M.-C. Kim, D.-J. Lee, and I. L. Markov, "Simpl: An effective placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 50–60, 2011. [Online]. Available: https://doi.org/10.1109/TCAD.2011.2170567

[18] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005*. IEEE, 2005, pp. 48–55. [Online]. Available: https://doi.org/10.1109/ICCAD.2005.1560039

[19] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 32, no. 5, pp. 709–722, 2013. [Online]. Available: https://doi.org/10.1109/TCAD.2012.2235124

[20] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proceedings of the 2008 International Symposium on Physical Design*, ser. ISPD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 47–53. [Online]. Available: https://doi.org/10.1145/1353629.1353640

[21] D. Liu *et al.*, "Layer assignment and routing optimization for advanced technologies," Ph.D. dissertation, 2018. [Online]. Available: http://hdl.handle.net/2152/69109

[22] M. Cho and D. Z. Pan, "Boxrouter: A new global router based on box expansion and progressive ilp," p. 373–378, 2006. [Online]. Available: https://doi.org/10.1145/1146909.1147009

[23] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "Nthu-route 2.0: a robust global router for modern designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1931–1944, 2010. [Online]. Available: https://doi.org/10.1109/TCAD.2010.2061590

[24] A. Volkov and S. Dolgov, "ICCAD-2019 CAD contest in lef/def based open-source global router," in *ICCAD*, 2019. [Online]. Available: http://iccad-contest.org/2019

[25] M. Pan and C. Chu, "Fastroute: A step to integrate global routing into placement," in *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 464–471. [Online]. Available: https://doi.org/10.1145/1233501.1233596

[26] ——, "Fastroute 2.0: A high-quality and efficient global router," in *2007 Asia and south pacific design automation conference*. IEEE, 2007, pp. 250–255. [Online]. Available: https://doi.org/10.1109/ASPDAC.2007.357994

[27] Y. Zhang, Y. Xu, and C. Chu, "Fastroute3. 0: a fast and high quality global router based on virtual capacity," in *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2008, pp. 344–349. [Online]. Available: https://doi.org/10.1109/ICCAD.2008.4681596

[28] 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, 2007. [Online]. Available: https://doi.org/10.1109/TCAD.2007.907068

[29] Y. Zhang and C. Chu, "Construction of all rectilinear steiner minimum trees on the hanan grid and its applications to vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 9, pp. 1655–1668, 2013. [Online]. Available: https://doi.org/10.1109/TCAD.2019.2917896

[30] X. Jia, Y. Cai, Q. Zhou, G. Chen, Z. Li, and Z. Li, "Mcfroute: A detailed router based on multi-commodity flow method," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2014, pp. 397–404. [Online]. Available: https://doi.org/10.1109/ICCAD.2014.7001382

[31] X. Jia, Y. Cai, Q. Zhou, and B. Yu, "Mcfroute 2.0: A redundant via insertion enhanced concurrent detailed router," in *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 87–92. [Online]. Available: https://doi.org/10.1145/2902961.2902966

[32] G. Chen, C.-W. Pui, H. Li, J. Chen, B. Jiang, and E. F. Y. Young, "Detailed routing by sparse grid graph and minimum-area-captured path search," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 754–760. [Online]. Available: https://doi.org/10.1145/3287624.3287678

[33] G. Chen, C. Pui, H. Li, and E. F. Y. Young, "Dr. cu: Detailed routing by sparse grid graph and minimum-area-captured path search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 9, pp. 1902–1915, 2020. [Online]. Available: https://doi.org/10.1109/TCAD.2019.2927542

[34] S. M. M. Gonçalves, L. S. Rosa, and F. S. Marques, "Draps: A design rule aware path search algorithm for detailed routing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 7, pp. 1239–1243, 2020. [Online]. Available: https://doi.org/

[35] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3240765.3240766

[36] F.-K. Sun, H. Chen, C.-Y. Chen, C.-H. Hsu, and Y.-W. Chang, "A multithreaded initial detailed routing algorithm considering global routing guides," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, 2018, pp. 1–7. [Online]. Available: https://doi.org/10.1109/TCSII.2019.2937893

[37] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, p. 103–111, aug 1990. [Online]. Available: https://doi.org/10.1145/79173.79181

[38] A. B. Kahng, L. Wang, and B. Xu, "Tritonroute: The open-source detailed router," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 547–559, 2021. [Online]. Available: https://doi.org/10.1109/TCAD.2020.3003234

[39] ——, "Tritonroute-wxl: The open-source router with integrated drc engine," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1076–1089, 2021. [Online]. Available: https://doi.org/10.1109/TCAD.2021.3079268

[40] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "Ispd 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 157–164. [Online]. Available: https://www.ispd.cc/contests/15/ispd2015_contest.html

[41] OpenROAD, "Openroad," 2022. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD

[42] Cadence, "Innovus," 2022. [Online]. Available: https://www.cadence.com/ko_KR/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html

Table III.: Experimental results for ISPD 2018 Benchmarks.

Circuit	Flow			Routing Metrics		Routing Preference Metrics					Design Rule Violations				Score		Rank
	Placement	Global Routing	Detailed Routing	Wirelength (mm)	Vias (K)	OFGW (mm)	OFGV	OFTW (mm)	OFTV	WWW (mm)	Shorts	Min Area	Spacing	Open Nets	(K)	(%)	
ispd18_test1	Contest	CUGR	DRCU	171.7	**31.7**	0.4	195	0.2	0	2.5	28,800	0	1	0	286.1	0	6
			TritonRoute	**171.5**	35.7	2.9	438	0.3	171	3.4	0	0	0	0	302.5	5.7	**1**
		FastRoute	DRCU	181.3	35.0	71.1	19,413	0.3	33	6.7	19,326,600	1,740	212	0	1,547.4	440.9	7
			TritonRoute	175.5	38.2	89.4	25,612	0.4	220	5.7	0	0	0	0	559.7	95.6	**4**
	DreamPlace	CUGR	DRCU	177.6	36.0	1.0	605	0.1	0	2.0	0	0	7	0	305.6	6.8	**2**
			TritonRoute	176.3	38.0	3.0	2,061	0.6	196	3.5	0	0	0	0	315.6	10.3	**3**
		FastRoute	DRCU	187.6	35.0	72.8	19,213	0.3	36	6.5	13,139,000	1,771	267	0	1,582.3	453.1	8
			TritonRoute	182.6	38.7	93.6	25,831	0.4	215	5.4	0	0	0	0	579.8	102.7	**5**
	EhPlacer	FastRoute	DRCU	231.4	37.7	88.0	21,004	0.3	62	6.6	14,443,600	1,599	287	0	1,610.6	463	9
			TritonRoute	227.7	41.1	114.5	27,303	0.5	229	5.4	0	0	0	2	-	-	-
ispd18_test2	Contest	CUGR	DRCU	**3,120.5**	**316.0**	8.8	3,636	2.4	0	24.8	504,000	0	35	0	4,642.3	0	2
			TritonRoute	3,134.1	359.0	29.4	5,203	4.0	2,164	31.8	0	0	0	0	4,801.1	3.4	**1**
		FastRoute	DRCU	3,284.7	383.1	971.6	218,204	4.5	651	68.0	502,668,340	18,418	3,938	0	20,444.2	340.4	5
			TritonRoute	3,201.0	424.2	1,140.0	294,248	5.2	4,489	48.0	0	0	0	7	-	-	-
	DreamPlace	CUGR	DRCU	3,155.3	381.2	24.1	10,342	2.1	0	19.4	684,400	0	114	0	4,887.6	5.3	3
			TritonRoute	3,150.7	388.1	37.7	25,759	6.6	4,405	32.2	0	0	0	1	-	-	-
		FastRoute	DRCU	3,328.9	388.1	958.8	224,622	4.1	490	68.4	445,963,620	15,596	3,641	0	18,747.6	303.8	4
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	3,284.0	394.0	999.2	227,476	3.7	676	68.6	434,541,160	18,891	4,179	0	20,688.2	345.6	6
			TritonRoute	3,184.3	436.5	1,157.7	302,482	5.8	4,673	49.3	0	0	0	4	-	-	-
ispd18_test3	Contest	CUGR	DRCU	**3,490.6**	**316.2**	23.6	4,025	2.7	0	25.1	8,287,400	0	84	0	5,192.5	0	2
			TritonRoute	3,504.6	357.1	35.2	5,072	4.8	2,197	32.4	0	0	0	0	5,277.0	1.6	**1**
		FastRoute	DRCU	3,636.3	385.8	986.6	217,716	3.9	537	68.0	667,804,940	21,042	5,013	0	23,290.9	348.5	6
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	3,533.2	376.6	27.3	10,578	2.4	0	19.4	64,358,800	0	184	0	5,593.0	7.7	3
			TritonRoute	3,528.8	382.4	39.7	25,502	6.7	4,430	32.2	0	0	0	2	-	-	-
		FastRoute	DRCU	3,672.5	388.6	953.2	222,050	3.9	450	69.1	564,123,420	16,726	4,030	0	20,291.9	290.8	4
			TritonRoute	3,610.8	432.0	1,111.4	299,697	5.7	4,541	49.2	0	0	0	1	-	-	-
	EhPlacer	FastRoute	DRCU	3,986.5	409.7	1,000.2	231,887	3.7	554	70.7	641,798,620	17,562	4,070	0	21,538.5	314.8	5
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd18_test4	Contest	CUGR	DRCU	5,269.8	727.6	30.4	11,610	3.6	0	38.5	1,535,508	16	677	0	15,360.4	3.3	3
			TritonRoute	**5,239.8**	**719.4**	27.2	7,965	10.0	17,336	27.8	0	0	0	0	14,863.5	0	**1**
		FastRoute	DRCU	5,471.8	955.1	1,761.6	603,650	7.4	593	96.2	253,641,208	21,317	8,208	0	43,434.1	192.2	6
			TritonRoute	5,375.2	976.3	1,486.1	687,267	11.7	16,938	61.4	0	0	1	1	-	-	-
	DreamPlace	CUGR	DRCU	5,356.0	719.9	35.6	21,525	3.8	0	41.1	1,644,428	103	612	0	15,622.3	5.1	4
			TritonRoute	5,313.3	721.5	43.9	52,160	10.5	17,363	31.5	0	0	0	0	15,198.7	2.3	**2**
		FastRoute	DRCU	5,555.7	968.2	1,783.0	620,757	7.9	556	96.4	211,694,612	18,680	8,216	0	41,958.1	182.3	5
			TritonRoute	5,458.8	988.0	1,455.0	700,581	11.8	17,279	60.8	0	0	0	4	-	-	-
	EhPlacer	FastRoute	DRCU	9,262.5	1,439.9	3,743.1	1,071,534	8.8	426	295.4	8,132,654,144	18,731	37,502	0	177,097.4	1091.5	7
			TritonRoute	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-
ispd18_test5	Contest	CUGR	DRCU	5,504.0	926.9	21.9	6,401	1.2	3	10.2	3,205,312	72	482	0	16,100.4	3.1	6
			TritonRoute	**5,481.3**	**843.4**	20.1	8,227	3.8	14,946	17.1	0	0	0	0	15,608.9	0	**1**
		FastRoute	DRCU	5,609.4	958.7	2,936.1	735,473	4.7	666	77.5	819,660,424	25,720	9,489	0	59,606.9	281.9	10
			TritonRoute	5,591.8	978.2	2,398.0	781,933	5.6	21,272	45.2	0	0	0	0	28,969.0	85.6	**3**
	DreamPlace	CUGR	DRCU	5,614.5	915.5	27.7	23,559	1.7	10	14.1	172,833,808	701	604	0	18,916.6	21.2	7
			TritonRoute	5,579.6	889.3	62.8	63,515	4.0	19,637	18.2	0	0	0	0	16,225.5	4	**2**
		FastRoute	DRCU	5,713.6	973.5	2,955.7	744,270	4.4	564	76.7	760,443,668	19,502	7,287	0	55,048.5	252.7	8
			TritonRoute	5,695.1	986.6	2,439.4	786,447	5.4	21,538	45.2	0	0	1	0	29,456.0	88.7	**4**
	EhPlacer	FastRoute	DRCU	5,887.3	969.3	3,071.6	741,542	4.3	646	72.6	713,646,584	20,817	7,643	0	56,281.5	260.6	9
			TritonRoute	5,882.7	1,005.9	2,564.7	802,004	5.7	21,412	46.8	0	0	0	0	30,613.2	96.1	**5**
ispd18_test6	Contest	CUGR	DRCU	7,118.8	1,388.1	8.3	6,059	2.4	16	14.5	42,000	106	640	0	21,072.8	2.2	6
			TritonRoute	**7,100.2**	**1,278.5**	31.5	10,082	4.2	22,899	23.8	0	0	0	0	20,627.5	0	**1**
		FastRoute	DRCU	7,267.7	1,449.6	4,421.3	1,119,359	37.6	18,245	115.4	1,210,463,600	28,627	11,483	0	80,169.0	288.7	10
			TritonRoute	7,236.5	1,464.1	3,692.4	1,169,063	5.4	31,482	65.6	0	0	0	0	41,023.6	98.9	**3**
	DreamPlace	CUGR	DRCU	7,787.0	1,398.6	19.2	27,471	2.9	20	17.7	215,210,100	1,168	671	0	26,093.6	26.5	7
			TritonRoute	7,752.2	1,381.8	92.6	76,400	4.9	28,522	24.7	0	0	0	0	22,847.7	10.8	**2**
		FastRoute	DRCU	7,973.7	1,514.9	4,615.3	1,168,527	26.2	11,203	122.2	985,468,100	17,806	8,356	0	73,295.9	255.3	8
			TritonRoute	7,924.9	1,509.4	3,866.8	1,201,688	5.5	31,301	67.5	0	0	0	0	43,749.4	112.1	**5**
	EhPlacer	FastRoute	DRCU	7,500.7	1,464.5	4,641.0	1,121,862	31.9	17,153	108.0	1,046,733,300	26,581	11,053	0	78,545.7	280.8	9
			TritonRoute	7,484.8	1,488.9	3,924.6	1,177,346	5.6	31,088	66.3	0	0	0	0	42,866.5	107.8	**4**
ispd18_test7	Contest	CUGR	DRCU	12,977.9	2,289.1	16.5	9,888	5.3	0	22.8	5,395,132	148	93	0	37,430.5	1.2	5
			TritonRoute	**12,917.7**	**2,096.0**	49.1	15,965	7.1	28,490	37.4	0	0	0	0	36,980.6	0	**1**
		FastRoute	DRCU	13,417.5	2,443.1	4,461.1	1,497,361	39.7	26,634	255.5	1,746,034,824	41,214	17,876	0	115,006.7	211	8
			TritonRoute	13,165.7	2,409.5	2,966.0	1,533,012	9.1	29,076	100.9	0	0	2	0	54,653.6	47.8	**4**
	DreamPlace	CUGR	DRCU	13,141.7	2,228.2	60.1	117,793	6.4	0	30.6	514,719,112	2,399	853	0	45,957.4	24.3	6
			TritonRoute	13,055.3	2,214.0	172.2	196,878	8.0	28,557	37.8	0	0	0	0	38,361.8	3.7	**2**
		FastRoute	DRCU	13,603.5	2,469.0	4,208.5	1,450,091	36.0	23,174	241.2	1,599,312,268	35,892	16,507	0	108,949.3	194.6	7
			TritonRoute	13,368.1	2,414.5	2,825.6	1,463,221	9.2	29,061	102.3	0	0	0	0	54,404.1	47.1	**3**
	EhPlacer	FastRoute	DRCU	24,129.9	3,364.2	7,571.4	2,124,913	184.7	13,664	649.5	38,162,386,680	30,369	77,606	0	641,775.1	1635.4	9
			TritonRoute	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-
ispd18_test8	Contest	CUGR	DRCU	13,099.3	2,345.9	26.2	11,451	5.4	0	23.7	5,652,108	164	144	0	37,939.3	1.4	5
			TritonRoute	**13,032.5**	**2,146.5**	57.6	18,251	8.0	28,508	39.1	0	0	0	0	37,424.3	0	**1**
		FastRoute	DRCU	13,483.2	2,470.8	4,469.6	1,503,164	39.0	25,424	259.6	1,690,035,584	38,876	17,592	0	113,281.4	202.7	8
			TritonRoute	13,221.8	2,426.7	2,959.4	1,531,564	9.2	29,140	102.5	0	0	1	0	54,801.6	46.4	**3**
	DreamPlace	CUGR	DRCU	13,237.1	2,285.5	76.9	122,201	6.4	0	31.3	491,893,448	2,386	807	0	46,088.1	23.2	6
			TritonRoute	13,137.3	2,263.6	186.7	200,180	8.9	28,595	38.9	0	0	0	0	38,749.7	3.5	**2**
		FastRoute	DRCU	13,662.7	2,485.0	4,348.7	1,474,868	35.9	22,029	250.2	1,606,529,252	33,596	16,048	0	108,611.4	190.2	7
			TritonRoute	13,404.1	2,428.7	2,888.2	1,487,491	9.1	29,080	103.1	0	0	0	0	54,863.6	46.6	**4**
	EhPlacer	FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd18_test9	Contest	CUGR	DRCU	10,872.6	2,341.2	22.0	10,455	4.4	0	22.1	154,000	204	117	0	32,268.5	1.3	6
			TritonRoute	**10,808.3**	**2,149.7**	57.8	17,983	7.5	28,478	37.7	0	0	0	0	31,863.2	0	**1**
		FastRoute	DRCU	11,229.9	2,394.8	7,242.1	1,854,366	59.3	34,432	211.9	1,928,646,624	53,556	20,101	0	133,108.1	317.7	10
			TritonRoute	11,092.0	2,404.9	5,805.8	1,913,859	9.1	28,970	102.3	0	0	0	0	64,046.1	101	**3**
	DreamPlace	CUGR	DRCU	10,980.6	2,289.5	37.5	46,220	5.1	0	28.2	447,390,300	1,882	543	0	39,223.0	23.1	7
			TritonRoute	10,884.2	2,264.8	161.4	126,621	8.0	28,862	37.6	0	0	0	0	32,910.9	3.3	**2**
		FastRoute	DRCU	11,329.3	2,391.3	7,238.4	1,847,634	61.2	33,480	199.0	1,908,258,040	50,640	19,312	0	131,155.9	311.6	9
			TritonRoute	11,217.9	2,404.1	5,864.3	1,909,683	9.0	28,991	102.6	0	0	0	0	64,648.7	102.9	**4**
	EhPlacer	FastRoute	DRCU	12,282.4	2,426.9	7,718.8	1,862,499	52.9	29,320	200.5	1,873,411,300	42,861	17,660	0	130,858.6	310.7	8
			TritonRoute	12,188.4	2,482.8	6,330.6	1,960,986	9.8	29,034	106.9	0	0	0	0	69,638.3	118.6	**5**
ispd18_test10	Contest	CUGR	DRCU	13,623.6	2,496.3	137.5	27,585	6.2	0	29.0	13,705,600	230	669	0	40,548.3	2.3	2
			TritonRoute	**13,559.7**	**2,309.9**	145.7	52,557	12.9	32,519	52.4	0	0	0	0	39,626.5	0	**1**
		FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	16,187.5	2,495.6	331.9	102,678	8.2	427	42.4	532,716,036	1,936	3,097	0	56,630.4	42.9	3
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
		FastRoute	DRCU	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	Time Out 72h		-	-	-	-	-	-	-	-	-	-	-	-

Table IV.: Experimental results for ISPD 2019 Benchmarks.

Circuit	Flow			Routing Metrics		Routing Preference Metrics					Design Rule Violations				Score		Rank
	Placement	Global Routing	Detailed Routing	Wirelength (mm)	Vias (K)	OFGW (mm)	OFGV	OFTW (mm)	OFTV	WWW (mm)	Shorts	Min Area	Spacing	Open Nets	(K)	(%)	
ispd19_test1	Contest	CUGR	DRCU	128.5	38.4	0.5	220	0.1	731	1.7	100,000	30	13	0	433.4	4.3	5
			TritonRoute	**126.0**	38.0	1.9	524	0.4	3,965	1.8	0	0	0	0	415.4	0	**1**
		FastRoute	DRCU	130.8	38.8	77.2	29,056	0.5	1,274	3.2	25,746,516	1,377	779	0	2,238.0	438.8	7
			TritonRoute	128.4	38.8	75.8	28,953	0.4	4,064	2.4	0	0	0	0	823.9	98.4	**3**
	DreamPlace	CUGR	DRCU	134.9	**37.9**	2.1	1,882	0.2	718	2.2	6,710,756	31	85	0	579.2	39.4	6
			TritonRoute	132.1	38.7	5.0	3,396	0.4	3,990	2.0	0	0	0	0	450.9	8.5	**2**
		FastRoute	DRCU	137.1	39.7	80.5	30,076	0.6	1,359	3.2	29,962,292	1,543	907	0	2,473.4	495.5	9
			TritonRoute	135.0	39.7	79.6	29,937	0.4	4,080	2.5	0	0	0	0	862.6	107.7	**4**
	EhPlacer	FastRoute	DRCU	163.2	41.1	96.0	30,938	0.4	1,182	3.3	33,117,896	1,229	810	0	2,453.1	490.6	8
			TritonRoute	161.1	41.6	93.5	31,272	0.5	4,109	2.6	0	0	0	1	-	-	-
ispd19_test2	Contest	CUGR	DRCU	4,979.7	842.3	17.7	6,372	4.6	23,390	28.2	5,145,400	1,190	2,336	0	16,232.1	11	3
			TritonRoute	**4,939.9**	798.2	65.2	19,160	14.3	132,593	32.0	0	0	0	0	14,619.6	0	**1**
		FastRoute	DRCU	5,132.3	936.4	1,078.0	481,483	7.0	26,866	72.1	804,037,992	12,443	15,185	0	44,844.4	206.7	6
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	5,059.5	801.0	77.3	52,073	4.6	23,458	52.4	238,509,864	1,651	3,417	0	20,501.6	40.2	4
			TritonRoute	4,992.0	**790.9**	137.2	90,630	14.9	133,119	37.3	0	0	3	0	15,196.9	3.9	**2**
		FastRoute	DRCU	5,194.1	936.1	1,059.1	487,862	6.9	27,084	72.6	799,785,424	13,632	17,460	0	46,591.6	218.7	7
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	5,057.3	913.8	1,027.6	465,330	6.1	26,043	73.2	709,309,396	10,919	14,970	0	42,292.2	189.3	5
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd19_test3	Contest	CUGR	DRCU	166.7	66.4	1.0	478	0.3	664	2.5	560,000	67	270	0	744.6	26.4	2
			TritonRoute	**164.3**	63.4	5.3	2,056	0.5	6,083	3.2	0	0	0	0	589.0	0	**1**
		FastRoute	DRCU	173.8	65.2	41.8	29,558	0.9	1,436	8.8	75,885,988	2,755	2,014	0	4,184.1	610.4	6
			TritonRoute	166.6	63.2	42.1	30,890	0.6	6,467	5.0	0	0	0	5	-	-	-
	DreamPlace	CUGR	DRCU	173.8	**60.9**	5.2	4,167	0.4	703	5.2	4,803,524	85	319	0	876.5	48.8	4
			TritonRoute	170.6	62.8	13.3	9,833	0.6	6,338	4.4	8,016,512	0	13	0	765.2	29.9	3
		FastRoute	DRCU	181.4	65.3	44.6	29,828	0.8	1,481	8.8	75,685,460	2,736	1,967	0	4,182.1	610	5
			TritonRoute	173.6	64.0	44.4	31,253	0.6	6,417	5.0	7,962,596	0	16	7	-	-	-
	EhPlacer	FastRoute	DRCU	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd19_test4	Contest	CUGR	DRCU	5,928.7	**907.4**	77.8	24,813	2.4	2	27.1	35,990,788	142	466	0	17,945.6	0	1
			TritonRoute	5,428.1	1,050.3	899.5	286,445	31.1	3,874	259.2	35,409,532,306	0	90,561	1	-	-	**-**
		FastRoute	DRCU	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	**5,334.3**	926.1	725.4	175,288	3.5	7	131.2	26,140,503,316	1,764	14,953	0	354,769.4	1876.9	2
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
		FastRoute	DRCU	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	FastRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd19_test5	Contest	CUGR	DRCU	967.7	137.9	1.8	1,040	0.4	21	3.7	2,949,569	6	163	0	2,845.7	0	1
			TritonRoute	**926.5**	150.9	33.3	10,184	1.3	155	15.7	2,804,046,064	0	31	0	37,942.7	1233.3	5
		FastRoute	DRCU	976.0	150.9	222.7	84,325	1.1	0	15.9	192,367,300	4,071	8,101	0	12,512.3	339.7	2
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	937.4	**136.9**	16.3	5,537	0.5	25	6.5	2,393,896,156	27	1,161	0	33,256.4	1068.7	4
			TritonRoute	930.0	158.0	76.1	25,044	1.6	169	19.1	2,890,065,340	0	1,287	0	39,915.3	1302.7	6
		FastRoute	DRCU	984.1	152.5	222.6	86,056	1.1	4	16.0	213,965,102	3,586	9,820	0	13,425.1	371.8	3
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	EhPlacer Fail		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	EhPlacer Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd19_test6	Contest	CUGR	DRCU	13,241.1	2,086.8	38.8	14,132	5.3	12,206	53.9	7,100,800	1,097	1,541	0	39,187.0	4.3	3
			TritonRoute	13,116.8	1,972.3	92.6	24,966	12.2	46,333	57.3	0	0	0	0	37,587.8	0	**1**
		FastRoute	DRCU	13,572.1	2,344.3	2,916.7	1,236,163	21.8	27,598	180.6	1,883,483,896	25,974	28,825	0	106,366.7	183	5
			TritonRoute	13,347.6	2,220.2	2,064.6	1,180,111	12.8	57,808	101.3	0	3	15	3	-	-	-
	DreamPlace	CUGR	DRCU	13,305.4	1,967.4	159.8	123,064	5.6	12,250	112.1	558,908,764	3,442	3,066	0	48,947.9	30.2	4
			TritonRoute	13,136.4	**1,947.6**	210.1	176,021	12.2	54,757	68.9	0	6	16	0	38,403.2	2.2	**2**
		FastRoute	DRCU	13,617.0	2,332.1	2,770.6	1,239,857	27.9	33,548	176.0	2,056,446,124	35,526	40,227	0	118,364.8	214.9	7
			TritonRoute	13,404.3	2,197.0	2,081.6	1,165,393	12.8	58,685	103.2	0	3	6	7	-	-	-
	EhPlacer	FastRoute	DRCU	12,985.0	2,301.3	2,823.6	1,224,941	21.6	29,265	178.0	1,832,001,176	29,731	37,135	0	109,714.1	191.9	6
			TritonRoute	**12,786.0**	2,197.4	2,150.7	1,178,931	12.9	59,735	106.0	0	0	6	3	-	-	-
ispd19_test7	Contest	CUGR	DRCU	24,403.2	4,069.6	120.4	40,845	8.5	24,497	145.5	64,847,568	4,392	10,615	0	78,876.9	11.6	3
			TritonRoute	**24,198.2**	3,789.3	264.6	88,661	34.3	255,196	169.9	0	0	0	0	70,676.3	0	**1**
		FastRoute	DRCU	26,656.8	3,882.4	5,059.1	1,672,731	29.1	43,217	1,484.9	38,070,911,988	77,462	375,867	0	811,466.6	1048.1	6
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	24,929.0	3,737.7	720.8	331,608	9.5	24,466	437.3	1,325,446,516	9,983	19,816	0	107,435.9	52	4
			TritonRoute	24,380.9	**3,726.0**	707.9	554,977	40.3	264,367	231.0	0	0	10	0	74,023.8	4.7	**2**
		FastRoute	DRCU	26,861.0	3,877.7	5,107.7	1,670,887	29.1	43,859	1,471.5	37,245,083,408	79,271	386,301	0	807,940.9	1043.2	5
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	49,293.4	4,773.7	8,797.0	2,124,200	33.7	39,101	2,978.5	203,966,053,804	69,234	996,638	0	3,276,417.3	4535.8	7
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
ispd19_test8	Contest	CUGR	DRCU	37,364.4	6,450.7	112.8	52,544	12.8	36,469	135.0	31,813,600	6,182	6,914	0	114,617.6	6.5	3
			TritonRoute	**37,073.1**	**6,176.4**	225.6	84,142	44.8	382,323	174.3	0	0	0	0	107,613.7	0	**1**
		FastRoute	DRCU	42,984.6	6,738.8	9,992.3	3,378,498	46.7	64,644	2,459.6	188,749,368,432	137,735	981,980	0	3,105,982.6	2786.2	6
			TritonRoute	37,965.3	6,907.2	7,768.3	3,976,150	56.3	424,486	347.5	0	0	9	7	-	-	-
	DreamPlace	CUGR	DRCU	37,794.7	6,225.4	307.4	280,245	13.7	36,473	289.4	2,273,847,480	17,616	17,092	0	156,049.5	45	4
			TritonRoute	37,377.2	6,241.9	590.5	547,359	47.0	403,575	203.4	0	0	14	0	110,971.9	3.1	**2**
		FastRoute	DRCU	43,186.1	6,501.1	10,113.9	3,256,179	46.0	65,140	2,353.6	181,849,673,384	144,279	1,039,681	0	3,051,842.4	2735.9	5
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	44,891.4	6,451.2	10,246.9	3,149,428	41.0	58,378	2,581.5	225,566,244,400	126,959	1,136,688	0	3,643,984.9	3286.2	7
			TritonRoute	39,899.5	7,100.6	8,590.8	4,157,514	61.7	440,137	376.4	0	4	3	9	-	-	-
ispd19_test9	Contest	CUGR	DRCU	56,504.4	10,745.0	169.1	86,994	21.3	60,771	228.3	70,983,984	10,527	12,946	0	177,563.1	7.7	3
			TritonRoute	**56,021.2**	**10,265.1**	376.2	141,606	77.4	637,914	295.0	0	3	2	0	164,914.6	0	**1**
		FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	57,251.9	10,392.7	477.8	452,374	22.6	60,722	492.3	4,037,655,500	30,912	32,847	0	251,685.5	52.6	4
			TritonRoute	56,560.7	10,412.6	991.6	914,673	80.7	671,099	337.5	0	0	24	0	170,672.4	3.5	**2**
		FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	59,937.4	11,569.4	13,728.6	6,719,820	101.0	730,078	603.6	0	0	25	16	-	-	-
ispd19_test10	Contest	CUGR	DRCU	55,827.8	10,337.6	167.5	62,283	21.1	61,013	325.7	110,579,000	10,726	13,665	0	176,464.0	8.1	3
			TritonRoute	**55,324.7**	9,574.1	517.7	156,548	85.1	637,558	423.5	1,000	0	15	0	163,180.2	0	2
		FastRoute	DRCU	61,709.3	9,380.7	9,704.5	3,471,939	81.8	102,649	3,366.4	118,752,103,024	191,805	1,551,339	0	2,598,141.5	1492.2	5
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	DreamPlace	CUGR	DRCU	57,261.1	9,436.4	1,236.7	601,340	23.6	61,034	956.5	2,476,987,464	28,511	45,958	0	241,910.1	48.2	4
			TritonRoute	56,003.6	9,427.7	1,446.3	1,197,385	96.9	663,260	572.3	0	0	66	0	171,092.9	4.8	**1**
		FastRoute	DRCU	62,410.5	**9,283.5**	9,855.2	3,463,843	82.3	104,980	3,344.5	121,739,103,340	205,335	1,596,340	0	2,666,942.6	1534.4	6
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-
	EhPlacer	FastRoute	DRCU	Out of Memory		-	-	-	-	-	-	-	-	-	-	-	-
			TritonRoute	TritonRoute Fail		-	-	-	-	-	-	-	-	-	-	-	-