ICCAD: G: VLSI Routing: Seeing Nano Tree in Giga Forest

Gengjie Chen (Advisor: Evangeline F. Y. Young)
The Chinese University of Hong Kong
gjchen@cse.cuhk.edu.hk

1 PROBLEM AND MOTIVATION

We are using nanometer-size transistors and Giga Hertz clock frequency in very large scale integration (VLSI). Under such extreme conditions, timing, power, manufacturability and reliability are all crucial issues in VLSI design. For example, 50% – 80% of gates in the high-performance integrated circuit (IC) today are repeaters, which do not perform useful computation but work for timing closure [40]; over 50% of the chip at around 7nm will be powered off and cannot be utilized due to the power constraint [26].

The tree structure is the major topology used in VLSI routing. Optimizing the tree and the forest is essential for a successful design flow. However, the problems are in general challenging. First, even for many *single-net routing* problems, finding an optimal tree from a huge candidate forest is already NP-hard. Second, for *multiple-net routing*, a large number of trees need to be built on the chip by sharing resources and need to be well coordinated for avoiding conflicts. Third, in order to achieve a full-flow success, it is also necessary to foresee routing trees and to consider *routability* in early stages (e.g., placement). We devise efficient and effective algorithms to tackle the three levels of challenges with not only practical considerations in VLSI design but also mathematical rigorousness and guarantee. The three levels of algorithms can be integrated together to help to resolve the crucial timing, power, manufacturability and reliability problems in VLSI routing.

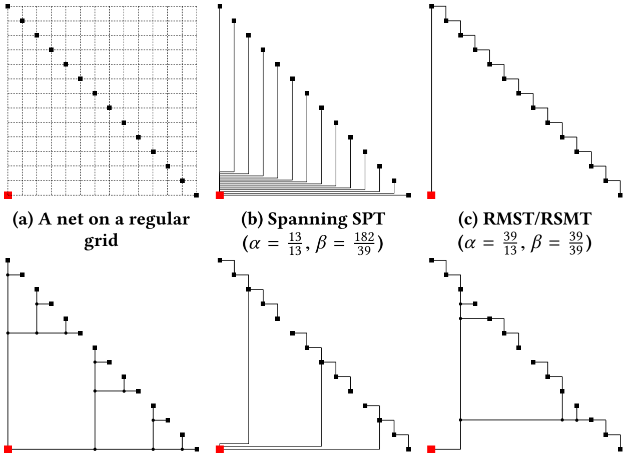
2 BACKGROUND AND RELATED WORK

For **single-net routing**, an important problem is the *shallow-light tree* (SLT) problem. For a signal net, shallowness implies wire delay, while lightness implies routing resource usage, power, cell delay and wire delay [19]. Formally, in a spanning/Steiner tree with *shallowness* α and *lightness* β, each path length is at most α times the shortest-path distance, while the tree weight is β times the minimum tree weight. For an $(\bar{\alpha}, \bar{\beta})$-SLT, $\alpha \leq \bar{\alpha}$ and $\beta \leq \bar{\beta}$.

It is a well-studied problem if only one of the objectives between light tree weight and shallow path length is pursued, whether the domain is the spanning tree or the rectilinear Steiner one. For spanning trees, both *minimum spanning tree* (MST) and *shortest-path tree* (SPT) can be obtained efficiently. For rectilinear Steiner trees, the one with minimum tree weight is called a *rectilinear Steiner minimum tree* (RSMT), while the lightest one with all paths from root being shortest is a *rectilinear Steiner minimum arborescence* (RSMA). Both problems are NP hard [27, 47] but many fast heuristics exist [18, 21]. SLT combines the two objectives together, as TABLE 1 and Figure 1 show [11]. A spanning SLT approximates SPT and MST simultaneously, where the trade-off is in the order of $(1 + \epsilon, O(\frac{1}{\epsilon}))$ [32]. Recently, Steiner SLTs are proved to be exponentially lighter than spanning ones by Elkin and Solomon [24, 25]. The ES algorithm can efficiently build a Steiner $(1 + \epsilon, O(\log \frac{1}{\epsilon}))$-SLT with a time complexity of $O(n^2)$. The constants in the bound $(1 + 2\epsilon, 4 + 2\lceil \log \frac{2}{\epsilon} \rceil)$ are, however, quite large.

Table 1: Spanning and Steiner Shallow-Light Tree (SLT)

	Shallowest	Lightest	Shallow light
Spanning	Spanning SPT $(O(m + n \log n))$	MST $(O(m + n \log n))$	Spanning SLT
Steiner	Steiner SPT (NP hard)	SMT (NP hard)	Steiner SLT
Rectilinear Steiner	RSMA (NP hard)	RSMT (NP hard)	Rectilinear Steiner SLT


(a) A net on a regular grid

(b) Spanning SPT ($\alpha = \frac{13}{13}, \beta = \frac{182}{39}$)

(c) RMST/RSMT ($\alpha = \frac{39}{13}, \beta = \frac{39}{39}$)

(d) RSMA ($\alpha = \frac{13}{13}, \beta = \frac{54}{39}$)

(e) Spanning SLT ($\alpha = \frac{17}{13}, \beta = \frac{61}{39}$)

(f) Steiner SLT ($\alpha = \frac{17}{13}, \beta = \frac{44}{39}$)

Figure 1: Routing trees with varied shallowness α and lightness β (root is marked by red).

Besides SLT, we also consider the wirelength and skew minimization in clock trees. Here, *skew*, the maximum difference in signal arrival time among all sinks, should be small to ensure timing correctness. There are in general two base formulations for clock tree construction, *zero-skew tree* (ZST) and *bounded-skew tree* (BST).

Many methods have been proposed for building ZSTs. Among them, deferred-merge embedding (ZST/DME) is a dynamic programming approach [6]. For a given topology, it outputs the locations of Steiner points achieving zero skew and optimal wirelength. For determining the topology, Greedy-DME [23] is regarded as the best algorithm in practice. Nowadays, ZST is not a good choice in practice due to two reasons. First, ZST is too expensive in wirelength, which implies excessive power usage [41] and usually comes with larger path divergence and suffers from more on-chip variations. Second, ZST is not necessary, considering the large tolerance (due to buffer insertion and sizing) and the widely-used useful-skew optimization techniques. Nonetheless, ZST is still useful as it can serve as the backbone of a BST [7, 53]. Regarding BST, ZST/DME is extended to BST/DME by generalizing the merging segments to regions [20]. Due to the more complicated shapes of the merging regions, BST/DME prunes many possibilities and is not optimal for a given topology.

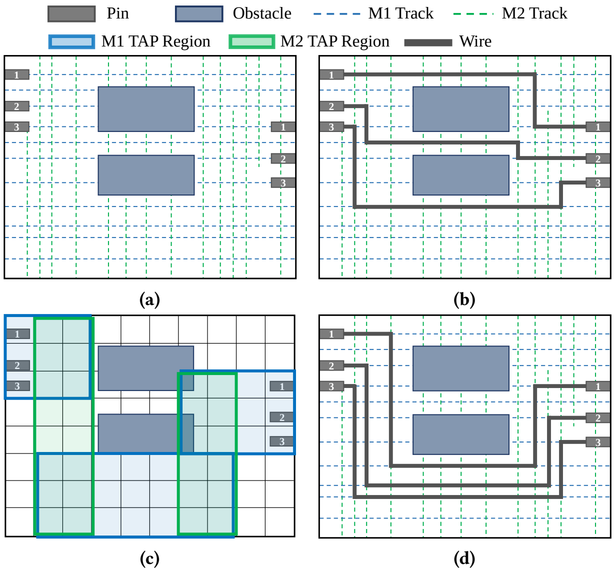


Figure 2: (a) A toy bus with two pins and three bits to. (b) Routing bits one after one. (c) Result of our topology-aware path planning. (d) Reulst of our track assignment for bits.

With theoretical interest, approximation algorithms are also proposed [7, 53]. The algorithm is not as good as the best heuristics in practice, but provides several inspirations for this work.

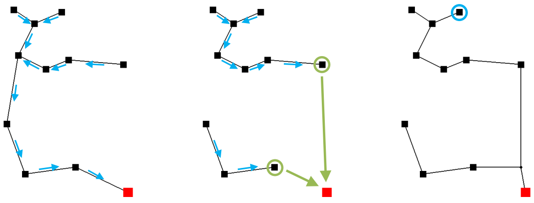
For **multiple-net routing**, it is challenging to coordinate different nets for a full-chip success. Because of its enormous computational complexity, VLSI routing is usually performed in two stages, global and detailed. In the *global routing* stage, the routing space is split into an array of regular cells (*g-cells*), where a coarse-grained routing solution is generated. It optimizes wire length, via count, routability, timing and other metrics with a global view. *Detailed routing*, on the other hand, realizes the global routing solution by considering exact metal shapes and positions. It takes care of many complicated detailed design rules due to manufacturability and reliability [54]. Meanwhile, its solution space, a 3D grid graph, is significantly larger than that of global routing. In advanced technology nodes, detailed routing becomes the most complicated and time-consuming stage [39]. During the past decade, many approaches were proposed to complete fast and high-quality global routing with sustaining progress [28, 37]. However, there is insufficient effort for exploring efficient and effective detailed routers in academia.

Meanwhile, the continuous development of modern VLSI technology has brought new challenges for on-chip interconnections. Different from classic net-by-net routing, bus routing requires all the nets (bits) in the same bus to share similar or even the same topology. Otherwise, it is very difficult to match the delay of different bits and close the timing. The techniques of net-by-net routing can hardly be straightforwardly applied in bus routing due to the difficulty of maintaining topology consistency. If processing bit by bit (e.g. route bit 1, 2 and 3 sequentially as in Figure 2 (b)), the latter bits may lack available track segments to be routed on. There are some previous works handling issues related to escape routing on printed circuit board (PCB) designs [33, 38]. However, for typical escape routing on PCB designs, having the same topology among different bits of the same bus is not a hard constraint. The work [36] tries to address the bus routing problem. It routes a representative bit first and all the other bits in the bus try to follow the topology of the representative one. However, the topology may not be achievable due to the lack of routing resources. To handle this issue, there is a post-refinement stage where the original bus will be divided into several sub-buses with different topologies.

To enable a full-flow success, it is also essential to consider **routability** in early stages. The continuous shrinking of the semiconductor feature size to 7 nm make the local pin access difficult [51]. The global routability also becomes more challenging due to ever increasing circuit size. It thus becomes more necessary to improve the routability of a design before the routing stage. As a general approach, inflating/depopulating cells according to a congestion estimation at g-cell level, has been successfully applied in both application-specific IC (ASIC) and field-programmable gate array (FPGA) [30, 35]. Besides, being programmable, the routing architecture in FPGA is quite different from that in ASIC [2]. There is work [29] considering the segmented routing of FPGA by graph embedding, but the objective is timing and congestion actually becomes worse in some cases. In [16] propose a smooth function to approximate the discrete routing cost under a nonlinear placement framework. As the major weakness, both works are not scalable for large designs due to the complicated problem formulation.

In short, the major limitations of the previous work on VLSI routing are in three folds. First, many works heavily relies on heuristics without any theoretical guarantee, which means inferior average-case quality and unpredictable worst-case performance. Second, step-after-step post processing may be effective in some situations, but it is insufficient for a comprehensive routing problem where many factors (e.g., power, timing, manufacturability) need to be simultaneously considered. Third, some methods cannot be scaled to today's chip with up to billions of transistors. To overcome the limitations, we made the following contributions.

- We study two fundamental problems for single-net routing with mathematical rigorousness. Regarding the trade-off between wirelength and path length in signal net, our proposed Steiner SLT algorithm achieves a best bound of $(1+\epsilon, 2+\lceil \log \frac{2}{\epsilon} \rceil)$ [11, 13]. For clock tree, we prove the equivalence between zero-skew tree problem and hierarchical clustering problem. The new insight lead us to simple yet more effective algorithms for pursuing wirelength and skew [12].
- We propose two fast and high-quality routers for two multiple-net routing problems beyound the relatively well-studied global routing problem. Special design rules are satisfied together with other design rules in a correct-by-construction manner. For detailed routing, a scalable framework with two-level sparse data structures and path search capturing the nontrivial minimum area constraint is designed [10]. In order to meet the topology consistency constraint in bus routing, a efficient maze router under a concurrent and hierarchical scheme is proposed [14].
- We also optimize the routability in early design stages to ensure a full-flow success. In FPGA packing and placement, a simultaneous packing and placement flow with some FPGA-routing-architecture-aware optimization techniques are proposed to achieve both the global and local routability [9, 43,



(a) MST T_M (b) Forest F (c) SALT T

Figure 3: Sample run of SALT algoritm ($\epsilon = 1$). (a) Construct MST T_M, where each blue arrow points from a vertex v to its parent $p[v]$. (b) Update $p[v]$ and identify breakpoints B (circled by green) during the DFS on T_M, which results to a forest F with tree roots being B. (c) Obtain the Steiner SPT T_B on $G[B \cup \{r\}]$, and $T = F \cup T_B$ is the final Steiner SLT.

44]. For the placement legalization of multi-row cells, a few routability issues (e.g., edge spacing, pin access, pin short) are considered besides minimizing the disturbance to the global placement with routability optimized [34].

3 APPROACH AND UNIQUENESS

3.1 Single-Net Routing

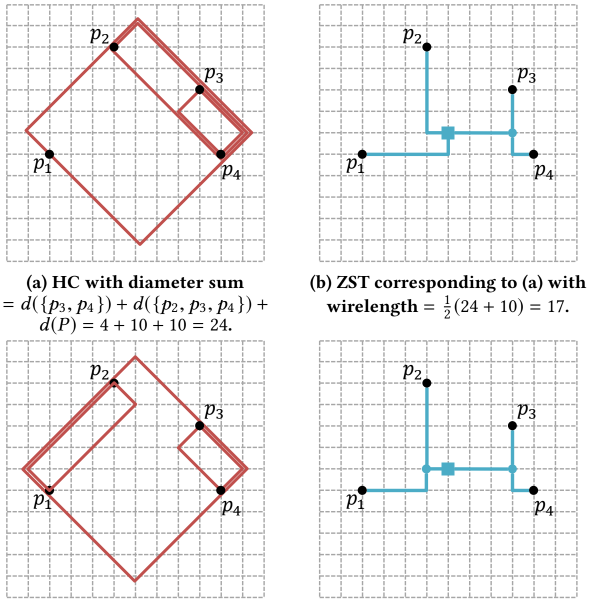
3.1.1 Trade-Off Between Wirelength and Path length. We propose SALT algorithm for the Steiner SLT on general graphs, where the trade-off between wirelength and path length is controlled by a parameter ϵ. The algorithm is briefed by Figure 3. SALT first identifies some breakpoints on an initial topology (MST) and then connect them to the root by a Steiner SPT, which is similar to ES [25]. However, we propose to use (i) a tighter criterion for identifying breakpoints and (ii) a better initial topology (i.e., an MST instead of a Hamiltonian path). The bound of SALT algorithm is stated by Theorem 1. To the best of our knowledge, the bound is tighter than all the previous methods for constructing general-graph spanning/Steiner SLTs.

Theorem 1. *SALT generates a Steiner* $(1 + \epsilon, 2 + \lceil \log \frac{2}{\epsilon} \rceil)$*-SLT.*

Proof. Refer to Theorem 3 of [13]. □

When applied to the Manhattan space for VLSI routing, SALT is simplified with runtime reduced from $O(n^2)$ to $O(n \log n)$. Moreover, we further decrease path lengths and tree weight by integrating SALT with the classical RSMA [45] and RSMT [18] algorithms. It provides a smooth trade-off between RSMA and RSMT controlled by ϵ. This is similar to what KRY and PD algorithms do for MST and SPT, but with an exponentially tighter bound. Moreover, three safe refinement techniques and a shallowness-constrained edge substitution method are also adopted to further improve the result.

3.1.2 Trade-Off Between Wirelength and Skew. A problem similar to ZST is the *hierarchical clustering* (HC) [50]. In HC, each point starts as a cluster by itself, and pairs of clusters are merged when moving up the hierarchy. By retrospecting the classical DME algorithm, we found the equivalence between ZST and HC. First of all, there is solution correspondence between ZST and HC. To be more specific, a ZST T implies a HC, where $leaves(v)$ of every $v \in T$ is treated as a cluster, and vice versa (see Figure 4 for two ZST/HC instances on the same set of points). Moreover, the wirelength of a ZST is a linear function of the sum of diameters of its corresponding HC, as Theorem 2 states.



(a) HC with diameter sum $= d(\{p_3, p_4\}) + d(\{p_2, p_3, p_4\}) + d(P) = 4 + 10 + 10 = 24$.

(b) ZST corresponding to (a) with wirelength $= \frac{1}{2}(24 + 10) = 17$.

(c) HC with diameter sum $= d(\{p_3, p_4\}) + d(\{p_1, p_2\}) + d(P) = 4 + 8 + 10 = 22$.

(d) ZST corresponding to (c) with wirelength $= \frac{1}{2}(22 + 10) = 16$.

Figure 4: Equivalence between zero-skew tree (ZST) and hierarchical clustering (HC) on points $P = \{p_1, p_2, p_3, p_4\}$.

Theorem 2. *For points/sinks P in the Manhatten space, a ZST T has a wirelength cost length(T) equivalent to the sum of diameters* $\sum_{v \in T} d(leaves(v))$ *of its corresponding HC:*

$$length(T) = \frac{1}{2}\left(\sum_{v \in T} d(leaves(v)) + d(P)\right) \quad (1)$$

Proof. Refer to Theorem 4 of [12]. □

With this new insight, we design an effective $O(n \log n)$-time iterative merging algorithm for ZST construction, which is also an $O(1)$-approximation algorithm. Besides, an optimal trie-based dynamic programming is proposed, which can also be used to make efficient local refinement to a sub-optimal ZST.

Based on the improved ZST method, we construct BST by combining RSMT and ZST with a help of a tree decomposition algorithm. A initial RSMT is decomposed into several subtrees under a constraint on the maximum distance of each subtree. Our proposed tree decomposition method can optimally minimize the number subtrees in linear time. The centers of the subtrees are then connected to a source by ZST and lead to a BST.

3.2 Multiple-Net Routing

3.2.1 Detailed Routing. Our detailed routing framework is superiorly scalable in runtime as well as memory usage and provides more correct-by-construction design rule satisfaction.

To handle a 3D detailed routing grid graph of enormous size, a set of two-level sparse data structures is designed, as shown in Figure 5. The global grid graph stores routed edges sparsely by binary search tree and intervals to enable economic memory usage and fast update.

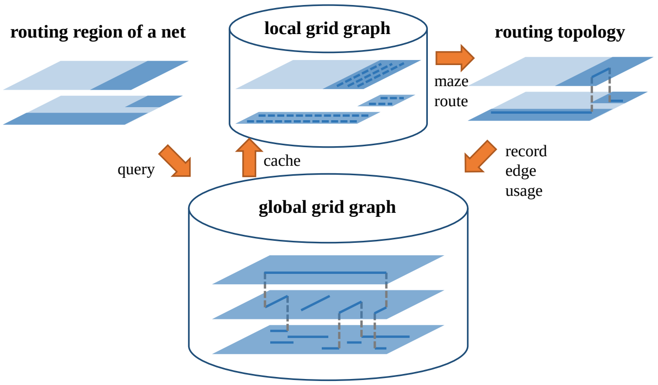


Figure 5: An overview of the two-level grid graph data structures for detailed routing.

When routing a net, a local grid graph is created according to the routing region of the net. It stores vertex/edge information (e.g., edge cost) explicitly by direct-address tables, which can be regarded as a cache of the global grid graph. After a net is routed, its solution will be record in the global grid graph so that later nets will avoid conflicting with it.

Since the minimum-area constraint is dynamic in nature and cannot be captured by the edge cost in the local grid graph, an optimal correct-by-construction path search algorithm is proposed. Besides, an efficient bulk synchronous parallel scheme is adopted to further reduce the runtime usage.

3.2.2 Bus Routing. We routes all the bits in a bus concurrently, instead of processing bit after bit. Such concurrency directly captures topology consistency constraint together with other objectives (e.g. wirelength) and constraints (e.g. spacing) in a correct-by-construction manner. A hierarchical framework, consisting of a topology-aware path planning (TAP) and a track assignment for bits (TAB), is designed for the efficiency. TAP is efficient as it works on a coarse-grained solution space (see Figure 2 (c)). TAB generates fine-grained routing solution, which gains efficiency by searching on the regions provided by TAB only (see Figure 2 (d)). We also utilize an effective rip-up and reroute scheme to further improve the routing solution quality.

3.3 Routability

3.3.1 Routability-Driven FPGA Packing and Placement. In a typical FPGA implementation flow, after logic synthesis, the netlist consists of lookup tables (LUTs), flip-flops (FFs), and some coarse-grained cores. During packing, LUTs and FFs are grouped together into basic logic elements (BLEs) and then further clustered into configurable logic blocks (CLBs). After packing, placement legally maps all the CLB blocks onto the FPGA. A proper packing can fully utilize the intra-BLE and intra-CLB routing resources and leave sufficient inter-CLB routing to global connections. However, packing without a physical view of placement may cause many long inter-CLB connetions and routing hotspots. Not only the convectional flow (pack-place) but also some recent works (place-pack-place [35], place-pack [42]) suffer from this artificial separation of packing and placement, as Figure 6(a) shows.

We propose a stair-step framework that interleaves the packing and placement stages (briefly illustrated by Figure 6(b)). It not only makes the optimization flow more smooth and integrated, but also enables fast feedback of accurate estimation (e.g., routability) from a final state to an intermediate state. Under this framework, we further adopt some routing-architecture-aware techniques including partition allocation, CLB slot assignment and alignment optimization, together with the ASIC-like congestion alleviation methods in global and detailed placement stages.

3.3.2 Routability-Driven Cell Legalization. VLSI placement usually start with a global placement, where cells may overlap with each other and may not align with placement sites. A legalization has to minimize the disturbance to global placement, where routability has been optimized with a global view. We first devise a incremental legalization method for multi-row-height cells. It inserts a cell optimally into a window, minimizing an average displacement of all the cells in the region from their global placement positions. Besides, a few routability issues, including edge spacing, pin short, and pin access (shown by Figure 7), are either incorporated into the cost function or converted to hard constraints in the formulation. A bipartite graph matching is then adopted to minimize the maximum displacement among a group of cells that can exchange their positions without creating additional violations. Finally, the minimum-cost flow formulation of the fixed-row-and-order problem optimizes a weighted sum of the maximum and average displacement, with range constraints on the cell movements to avoid pin short and pin access violations.

4 RESULTS AND CONTRIBUTIONS

4.1 Single-Net Routing

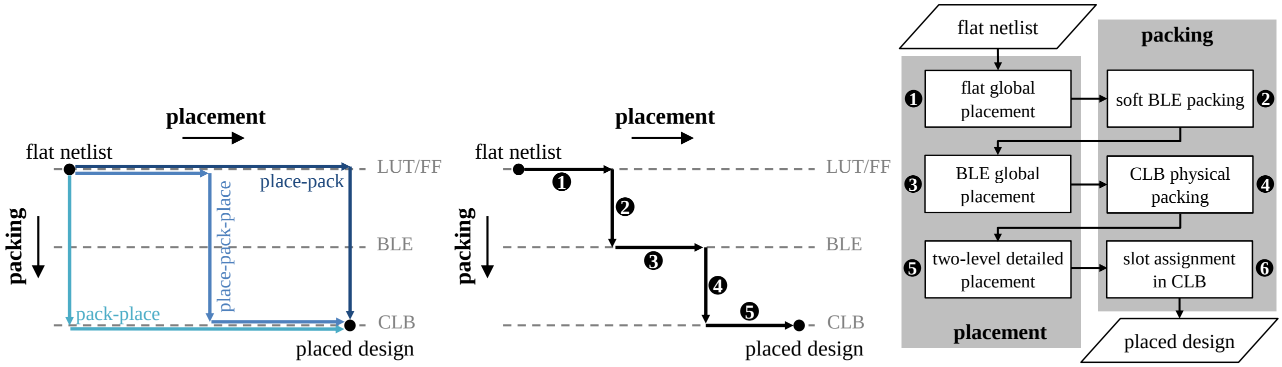
For trade-off between wirelength and path length, SALT shows superior performance, compared with both classical and recent routing tree construction methods (including FLUTE [18], CL [21], ABP [5], KRY [32], ES [25], Bonn [46], PD [4], and PD-II [3]), which is illustrated by Figure 8. It can be clearly observed that our method has the best Pareto frontier between RSMT and RSMA.

Figure 9(a) shows the result of BST construction (with # sinks being 16384). Here, our method shows better Pareto frontiers compared with BST/DME [20]. In general, as the skew bound becomes larger, a smooth decrease of wirelength from ZST to RSMT can also be observed. Moreover, our method is faster than BST/DME (Figure 9(b)). For relatively large nets, a 10× speed-up stably exists.

4.2 Multiple-Net Routing

Compared with the most recent works on detailed routing [31, 48], our detailed router achieves superior quality of result on ISPD 2018 benchmarks [39]. We have 13.7×–394× reduction in design rule violations together with shorter wirelength and fewer vias used. Moreover, our detailed routers is 8.2 and 32 times faster than the two works on average respectively.

For our bus router, we compare it with the winners in a recent contest [1]. Compared with them, it not only reduces spacing and short violations greatly but also achieves topology consistency for all buses. Under the contest evaluation scheme, it achieves the best total cost in seven out of eight cases. On average, our score is 1.97, 3.28, and 6.90 times better than the first, second and third places. At



(a) Three types of flows in previous work.

(b) Our proposed flow of simultaneous packing and placement.

Figure 6: Different FPGA packing and placement flows.

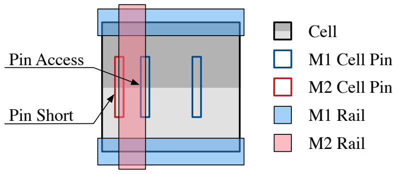


Figure 7: Pin short and pin access issues in cell legalization.

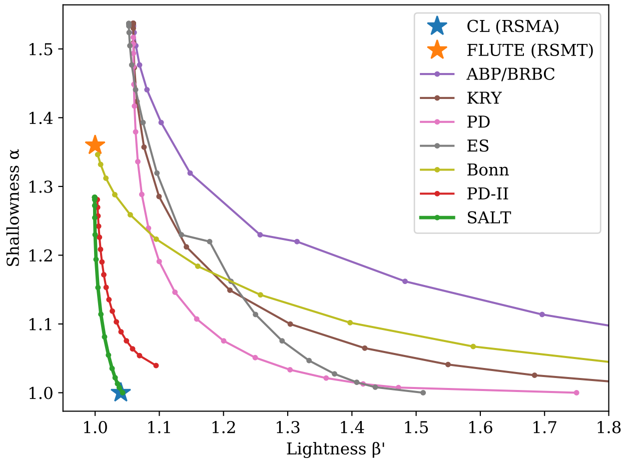
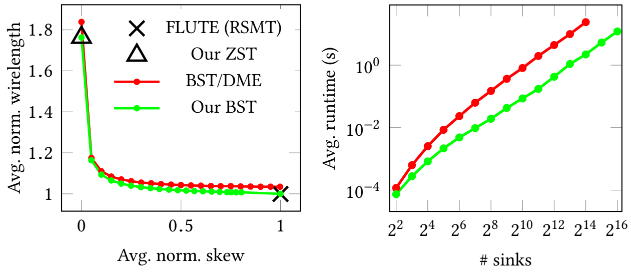


Figure 8: Comparing SALT with other methods.

the same time, our bus router runs significantly faster than the first place (with 1.6×–631× speed-up). This indicates the effectiveness and efficiency of the concurrent and hierarchical scheme for bus routing.

4.3 Routability

We use the benchmarks and evaluation flow provided by Xilinx [52] to evaluate our routability-driven packing and placement for FPGAs. The routed wirelength reported by the commercial FPGA router is the quality metric, which requires a truly routing-friendly packing and placement. Compared with a state-of-the-art work [35], our approach has 5.1% better routed wirelength and 5.5× speedup.



(a) Wirelength on 100 random nets with 16384 sinks.

(b) Runtime on random nets with various sizes.

Figure 9: Comparison our BST method with BST/DME.

The proposed routability-driven legalizer is first compared with some state-of-the-art placers [15, 17, 49] for total displacement minimization only. Here, 20%, 17% and 9% improvement can be observed respectively. When routability constraints and objectives are considered, we compared with the first place of ICCAD 2017 Contest [22]. Our method gets rid of all the edge spacing violations, improves the pin access by 8.3%, and achieves 18% and 12% reduction in average and maximum displacement at the same time.

4.4 Research Impact

My PhD study so far has led to 6 first-authored [8–13] and 4 coauthored [14, 34, 43, 44] publications about VLSI routing in premier design automation journals and conferences. Our algorithms tackle the three aspects of challenges, single-net routing, multiple-net routing, and early-stage routability optimization, with the considerations of both practical VLSI design needs and mathematical rigorousness. Our theoretical research on shallow-light tree construction has been recognized by the **Best Paper Award** of International Conference on Computer-Aided Design (ICCAD) in 2017. The method has also been applied in the tool of Synopsys, a major vendor of electronic design automation. Our dedication in building effective and efficient routers as well as routability-driven placers has been recognized by several **first and second place awards** in ICCAD and International Symposium on Physical Design (ISPD) contests, hosted by leading industrial companies including IBM, Xilinx, Cadence, Synopsys, and Mentor Graphics.

REFERENCES

[1] 2018. ICCAD 2018 Contest. http://iccad-contest.org/2018/.

[2] Rajat Aggarwal. 2014. FPGA place & route challenges. In *ACM International Symposium on Physical Design (ISPD)*. 45–46.

[3] Charles J Alpert, Wing-Kai Chow, Kwangsoo Han, Andrew B Kahng, Zhuo Li, Derong Liu, and Sriram Venkatesh. 2018. Prim-Dijkstra Revisited: Achieving Superior Timing-driven Routing Trees. In *ACM International Symposium on Physical Design (ISPD)*. 10–17.

[4] Charles J Alpert, TC Hu, JH Huang, Andrew B Kahng, and D Karger. 1995. Prim-Dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 14, 7 (1995), 890–896.

[5] Baruch Awerbuch, Alan Baratz, and David Peleg. 1992. *Efficient broadcast and light-weight spanners*. Technical Report.

[6] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and AB Kahng. 1992. Zero skew clock routing with minimum wirelength. *IEEE Transactions on Circuits and Systems II* 39, 11 (1992), 799–814.

[7] Moses Charikar, Jon Kleinberg, Ravi Kumar, Sridhar Rajagopalan, Amit Sahai, and Andrew Tomkins. 2004. Minimizing wirelength in zero and bounded skew clock trees. *SIAM Journal on Discrete Mathematics (SIDMA)* 17, 4 (2004), 582–595.

[8] Gengjie Chen, Jian Kuang, Zhiliang Zeng, Hang Zhang, Evangeline FY Young, and Bei Yu. 2017. Minimizing thermal gradient and pumping power in 3D IC liquid cooling network design. In *ACM/IEEE Design Automation Conference (DAC)*. 70.

[9] Gengjie Chen, Chak-Wa Pui, Wing-Kai Chow, Ka-Chun Lam, Jian Kuang, Evangeline FY Young, and Bei Yu. 2018. RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 37, 10 (2018), 2022–2035.

[10] Gengjie Chen, Chak-Wa Pui, Haocheng Li, Jinsong Chen, Bentian Jiang, and Evangeline FY Young. 2019. Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 754–760.

[11] Gengjie Chen, Peishan Tu, and Evangeline FY Young. 2017. SALT: provably good routing topology by a novel steiner shallow-light tree algorithm. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 569–576.

[12] Gengjie Chen and Evangeline FY Young. 2019. Dim Sum: Light Clock Tree by Small Diameter Sum. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*.

[13] Gengjie Chen and Evangeline FY Young. 2019. SALT: provably good routing topology by a novel steiner shallow-light tree algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* (2019).

[14] Jingsong Chen, Jinwei Liu, Gengjie Chen, Dan Zheng, and Evangeline FY Young. 2019. MARCH: Maze Routing Under a Concurrent and Hierarchical Scheme for Buses. In *ACM/IEEE Design Automation Conference (DAC)*.

[15] Jianli Chen, Ziran Zhu, Wenxing Zhu, and Yao-Wen Chang. 2017. Toward Optimal Legalization for Mixed-Cell-Height Circuit Designs. In *ACM/IEEE Design Automation Conference (DAC)*. 52:1–52:6.

[16] Sheng-Yen Chen and Yao-Wen Chang. 2015. Routing-architecture-aware analytical placement for heterogeneous FPGAs. In *ACM/IEEE Design Automation Conference (DAC)*. 27:1–27:6.

[17] Wing Kai Chow, Chak Wa Pui, and Evangeline F. Y. Young. 2016. Legalization Algorithm for Multiple-Row Height Standard Cell Design. In *ACM/IEEE Design Automation Conference (DAC)*. 83:1–83:6.

[18] Chris Chu and Yiu-Chung Wong. 2008. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 27, 1 (2008), 70–83.

[19] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H Madden. 1996. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal* 21, 1-2 (1996), 1–94.

[20] Jason Cong, Andrew B Kahng, Cheng-Kok Koh, and C-W Albert Tsao. 1998. Bounded-skew clock and Steiner routing. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 3, 3 (1998), 341–388.

[21] Javier Córdova and Yann-Hang Lee. 1994. *A heuristic algorithm for the rectilinear Steiner arborescence problem*. Technical Report.

[22] Nima Karimpour Darav, Ismail S. Bustany, Andrew Kennings, and Ravi Mamidi. 2017. ICCAD-2017 CAD Contest in Multi-Deck Standard Cell Legalization and Benchmarks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 867–871.

[23] Masato Edahiro. 1993. A clustering-based optimization algorithm in zero-skew routings. In *ACM/IEEE Design Automation Conference (DAC)*. 612–616.

[24] Michael Elkin and Shay Solomon. 2011. Steiner Shallow-Light Trees are Exponentially Lighter than Spanning Ones. In *IEEE Symposium on Foundations of Computer Science (FOCS)*. 373–382.

[25] Michael Elkin and Shay Solomon. 2015. Steiner shallow-light trees are exponentially lighter than spanning ones. *SIAM Journal on Computing (SICOMP)* 44, 4 (2015), 996–1025.

[26] Hadi Esmaeilzadeh, Emily Blem, Renée St Amant, Karthikeyan Sankaralingam, and Doug Burger. 2013. Power challenges may end the multicore era. *Commun. ACM* 56, 2 (2013), 93–102.

[27] Michael R Garey and David S. Johnson. 1977. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics (SIAP)* 32, 4 (1977), 826–834.

[28] Michael Gester, Dirk Müller, Tim Nieberg, Christian Panten, Christian Schulte, and Jens Vygen. 2013. BonnRoute: Algorithms and data structures for fast and good VLSI routing. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 18, 2 (2013), 32.

[29] Padmini Gopalakrishnan, Xin Li, and Lawrence Pileggi. 2006. Architecture-aware FPGA placement using metric embedding. In *ACM/IEEE Design Automation Conference (DAC)*. 460–465.

[30] Xu He, Tao Huang, Linfu Xiao, Haitong Tian, and Evangeline F. Y. Young. 2013. Ripple: A robust and effective routability-driven placer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 32, 10 (2013), 1546–1556.

[31] Andrew B Kahng, Lutong Wang, and Bangqi Xu. 2018. TritonRoute: an initial detailed router for advanced VLSI technologies. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 81:1–81:8.

[32] Samir Khuller, Balaji Raghavachari, and Neal Young. 1995. Balancing minimum spanning trees and shortest-path trees. *Algorithmica* 14, 4 (1995), 305–321.

[33] Hui Kong, Tan Yan, and Martin D. F. Wong. 2010. Optimal simultaneous pin assignment and escape routing for dense PCBs. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 275–280.

[34] Haocheng Li, Wing-Kai Chow, Gengjie Chen, Evangeline FY Young, and Bei Yu. 2018. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In *ACM/IEEE Design Automation Conference (DAC)*. 150.

[35] Wuxi Li, Shounak Dhar, and David Z. Pan. 2016. UTPlaceF: a routability-driven FPGA placer with physical and congestion aware packing. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 66:1–66:7.

[36] Derong Liu, Bei Yu, Vinicius Livramento, Salim Chowdhury, Duo Ding, Huy Vo, Akshay Sharma, and David Z Pan. 2018. Synergistic Topology Generation and Route Synthesis for On-Chip Performance-Critical Signal Groups. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* (2018), 18.

[37] Wen-Hao Liu, Wei-Chun Kao, Yih-Lang Li, and Kai-Yuan Chao. 2013. 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 (TCAD)* 32, 5 (2013), 709–722.

[38] Qiang Ma, Evangeline F. Y. Young, and Martin D. F. Wong. 2011. An optimal algorithm for layer assignment of bus escape routing on PCBs. In *ACM/IEEE Design Automation Conference (DAC)*. 176–181.

[39] Stefanus Mantik, Gracieli Posser, Wing-Kai Chow, Yixiao Ding, and Wen-Hao Liu. 2018. ISPD 2018 Initial Detailed Routing Contest and Benchmarks. In *ACM International Symposium on Physical Design (ISPD)*. 140–143.

[40] Igor L Markov. 2014. Limits on fundamental limits to computation. *Nature* 512, 7513 (2014), 147–154.

[41] David Papa, Charles Alpert, Cliff Sze, Zhuo Li, Natarajan Viswanathan, Gi-Joon Nam, and Igor Markov. 2011. Physical synthesis with clock-network optimization for large systems on chips. *IEEE/ACM International Symposium on Microarchitecture (MICRO)* 31, 4 (2011), 51–62.

[42] Ryan Pattison, Ziad Abuowaimer, Shawki Areibi, Gary Gréwal, and Anthony Vannelli. 2016. GPlace: a congestion-aware placement tool for ultrascale FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 68:1–68:7.

[43] Chak-Wa Pui, Gengjie Chen, Wing-Kai Chow, Ka-Chun Lam, Jian Kuang, Peishan Tu, Hang Zhang, Evangeline FY Young, and Bei Yu. 2016. RippleFPGA: A routability-driven placement for large-scale heterogeneous FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 67.

[44] Chak-Wa Pui, Gengjie Chen, Yuzhe Ma, Evangeline FY Young, and Bei Yu. 2017. Clock-aware ultrascale FPGA placement with machine learning routability prediction. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 929–936.

[45] Sailesh K Rao, P Sadayappan, Frank K Hwang, and Peter W Shor. 1992. The rectilinear Steiner arborescence problem. *Algorithmica* 7, 1-6 (1992), 277–288.

[46] Rudolf Scheifele. 2017. Steiner Trees with Bounded RC-Delay. *Algorithmica* 78, 1 (2017), 86–109.

[47] Weiping Shi and Chen Su. 2005. The rectilinear Steiner arborescence problem is NP-complete. *SIAM Journal on Computing (SICOMP)* 35, 3 (2005), 729–740.

[48] Fan-Keng Sun, Hao Chen, Ching-Yu Chen, Chen-Hao Hsu, and Yao-Wen Chang. 2018. A multithreaded initial detailed routing algorithm considering global routing guides. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 82:1–82:7.

[49] Chao-Hung Wang, Yen-Yi Wu, Jianli Chen, Yao-Wen Chang, Sy-Yen Kuo, Wenxing Zhu, and Genghua Fan. 2017. An effective legalization algorithm for mixed-cell-height standard cells. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 450–455.

[50] Rui Xu and Donald Wunsch. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks (TNN)* 16, 3 (2005), 645–678.

[51] Xiaoqing Xu, Nishi Shah, Andrew Evans, Saurabh Sinha, Brian Cline, and Greg Yeric. 2017. Standard cell library design and optimization methodology for ASAP7 PDK. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 999–1004.

[52] Stephen Yang, Aman Gayasen, Chandra Mulpuri, Sainath Reddy, and Rajat Aggarwal. 2016. Routability-Driven FPGA Placement Contest. In *ACM International Symposium on Physical Design (ISPD)*. 139–143.

[53] Alexander Z Zelikovsky and Ion I Mandoiu. 2002. Practical approximation algorithms for zero-and bounded-skew trees. *SIAM Journal on Discrete Mathematics (SIDMA)* 15, 1 (2002), 97–111.

[54] Yanheng Zhang and Chris Chu. 2013. RegularRoute: An efficient detailed router applying regular routing patterns. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)* 21, 9 (2013), 1655–1668.