TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/10 pp902–911
DOI: 10.26599/TST.2021.9010092
Volume 27, Number 6, December 2022

A Two-Stage Method for Routing in Field-Programmable Gate Arrays with Time-Division Multiplexing

Peihuang Huang, Longkun Guo*, Long Sun, and Xiaoyan Zhang

Abstract: Emerging applications widely use field-programmable gate array (FPGA) prototypes as a tool to verify modern very-large-scale integration (VLSI) circuits, imposing many problems, including routing failure caused by the limited number of connections among blocks of FPGAs therein. Such a shortage of connections can be alleviated through time-division multiplexing (TDM), by which multiple signals sharing an identical routing channel can be transmitted. In this context, the routing quality dominantly decides the performance of such systems, proposing the requirement of minimizing the signal delay between FPGA pairs. This paper proposes algorithms for the routing problem in a multi-FPGA system with TDM support, aiming to minimize the maximum TDM ratio. The algorithm consists of two major stages: (1) A method is proposed to set the weight of an edge according to how many times it is shared by the routing requirements and consequently to compute a set of approximate minimum Steiner trees. (2) A ratio assignment method based on the edge-demand framework is devised for assigning ratios to the edges respecting the TDM ratio constraints. Experiments were conducted against the public benchmarks to evaluate our proposed approach as compared with all published works, and the results manifest that our method achieves a better TDM ratio in comparison.

Key words: Field-programmable gate array (FPGA) routing; time-division multiplexing; minimum Steiner tree; exact algorithm; approximation algorithm

1 Introduction

With the developments in science and technology, the size of integrated circuits (IC) is rapidly shrinking. By increasing the number of components on a single chip and reducing its feature size of the chip, the density and integration of the chip are increased. With the increasing level of development and complexity of very large scale integrated (VLSI) circuits[2], the design and manufacturing costs of VLSI circuits are gradually increasing[3]. Hence, it is necessary to find an effective method to verify the effectiveness and efficiency of the VLSI.

The first strategy of logic verification is software simulation[4], which provides visibility and debugging capabilities. However, it requires each logic gate to be simulated individually, and would have a considerable cost and a very long runtime when the circuit size is very large. The second strategy is hardware emulation[5, 6], which greatly reduces the runtime but suffers a high implementation cost. The third method

• Peihuang Huang is with the College of Mathematics and Data Science, Minjiang University, Fuzhou 350116, China, and with the School of Mathematics Science, Nanjing Normal University, Nanjing 210024, China. E-mail: peihuang.huang@foxmail.com.
• Longkun Guo and Long Sun are with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China. E-mail: Longkun.guo@gmail.com.
• Xiaoyan Zhang is with the School of Mathematics Science and Institute of Mathematics, Nanjing Normal University, Nanjing 210024, China. E-mail: zhangxiaoyan@njnu.edu.cn.
• A preliminary version of this paper has been presented at the 21st International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2020)[1].
$*$ To whom correspondence should be addressed.
Manuscript received: 2021-07-10; revised: 2021-09-27; accepted: 2021-10-31

of logic verification is to use a field-programmable gate array (FPGA) prototype system. The FPGA prototype verifies the circuit through a configurable FPGA system, achieving a better tradeoff between the cost and runtime. Therefore, the FPGA prototype system has been widely used in the industry[7–9].

To adapt to the design of the FPGA prototype system, a large VLSI circuit must be divided into multiple sub-circuits[10], each of which corresponds to an FPGA. As the number of input/output (I/O) pins in FPGAs is fixed and limited, routing signals usually exceed the number of I/O pins. Babb et al.[11] introduced a time-division multiplexing (TDM) technique that can transmit multiple routing signals in one system clock cycle as shown in Fig. 1. The technique increases the signal capability in one FPGA and the high routability for the prototyping system. However, this technique also slows down the inter-FPGA signal delay[11].

Several works have targeted on the optimization of inter-FPGA connections for logic verification. In Ref. [12], an integer linear programming (ILP) based method is presented to select I/O signals to achieve a high frequency in 2-FPGA systems. However, due to the sharp increase in the size of the internal signal of the FPGA, the ILP-based method cannot easily generate a promising solution within a reasonable runtime. To optimize the TDM ratio, Pui et al.[13] presented an analytical framework for a multi-FPGA based system. However, in this architecture, only cross-FPGA networks with the same direction and TDM ratio can be allocated to the same line. In addition, the above work did not consider the TDM multiplexing ratio as an even number, which is an impractical hardware implementation of multiple channels[14]. Therefore, the development of an effective and efficient TDM based FPGA routing algorithm is desirable.

1.1 Problem model

We model the system-level FGPA routing problem with the following parts:

- An undirected graph $G(V, E)$, where V and E are the set of FPGA components and the connections between the FPGAs, respectively.
- A set of net accommodating the routing requirements ("nets" for short when no confusion arises) $\mathcal{N}$, where each net $N \in \mathcal{N}$ is a subset of V indicating the terminals (vertices) to be connected by the routing of N, respectively
- A set of groups $\mathcal{P}$, in which each group $P \in \mathcal{P}$ is composed by a set of nets. Clearly, $P \subseteq \mathcal{N}$ holds.

Without loss of generality, we assume that all the nets of the groups exactly compose exactly the set of nets $\mathcal{N}$. In the above setting, a vertex can be connected by one or multiple nets, a net may belong to multiple groups, and conversely a group can have many nets with identical sets of terminals.

The challenge of system-level FPGA routing lies in the side-effect of TDM. That is, the FPGA routing can always be complete, but the inter-FPGA signal delay could be huge as it depends on the TDM ratio[15]. Formally, the TDM ratio of edge e can be defined as follows:

$$r_e = \frac{\text{demand}_e}{\text{capacity}_e} \tag{1}$$

where demand_e and capacity_e are the demand of edge e and the capacity of edge e, respectively, for which we have $\text{demand}_e \leqslant \text{capacity}_e$. In the problem, the actual capacity of each edge is defined as 1, and the TDM ratio is defined as a positive even integer number because of the requirement of multiplexing hardware implementation. The TDM ratio of each net is defined as the sum of ratios of its edges, and the TDM ratio of each group is defined as the sum of ratios of all the nets

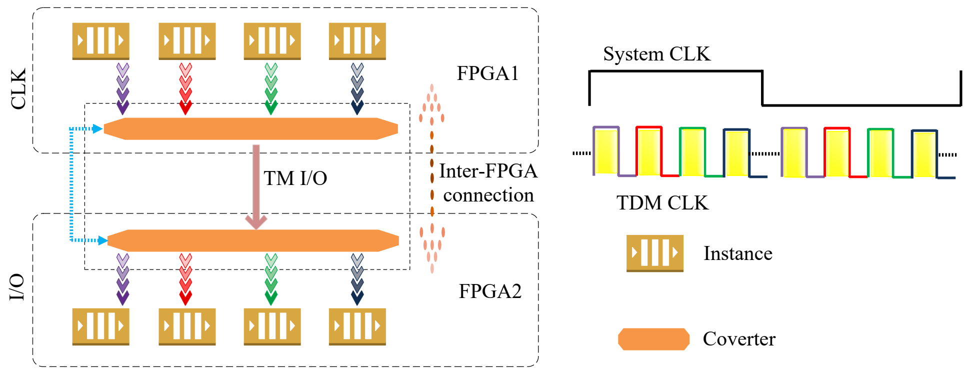

Fig. 1 Sample illustration of the TDM technique. CLK indicates clock.

it contains. Thus, we have

$$\begin{cases} r_N = r(N) = \sum_{e \in N} r_e, \\ r_P = r(P) = \sum_{N \in P} r_N \end{cases} \tag{2}$$

where r_N and r_P denote the total TDM ratio of net N and the total TDM ratio of group P, respectively.

Therefore, the problem can be formally described as follows: Given a multi-FPGA platform with TDM wires between FPGA connection pairs as $G = (V, E)$, a list of nets $\mathcal{N}$ which is to be routed, a collection of groups $\mathcal{P}$, the aim is to route the nets and assign a TMD ratio for each connection in each routed net, such that the TDM constraints are strictly satisfied and the maximum total TMD ratio among the groups is minimized.

1.2 Algorithm flow

In this paper, we propose a two-stage efficient TDM-based FPGA routing algorithm, where the first part computes the topology accommodating each net of the given groups, and the second part assigns ratio to each edge of the nets according to the computed topology. The detailed algorithm flow is as illustrated in Fig. 2.

1.3 Results

In this paper, we propose a TDM-based FPGA routing method with a two-stage framework. We present new techniques for both topology computation and ratio assignment. The major contributions of our work can be summarized as follows:

• For each iteration of processing a single group, we introduce a simple but equivalent version of the problem with respect to one group, and devise convex programming (CP) for the computation against such group with nets of identical source (or destination) terminals. The CP has only a polynomial number of constraints, which favorably compares to the exponential size constraints of the existing mathematical programs

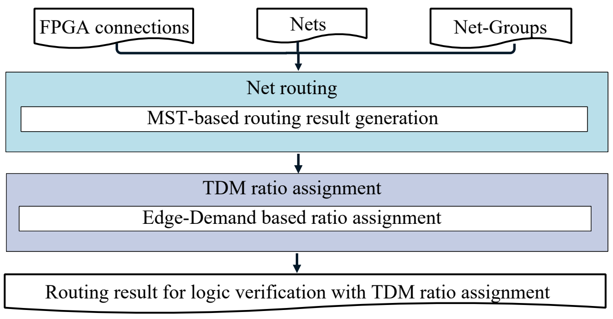
FPGA connections
Nets
Net-Groups
Net routing
MST-based routing result generation
TDM ratio assignment
Edge-Demand based ratio assignment
Routing result for logic verification with TDM ratio assignment

Fig. 2 Flow chart of the algorithm, where MST indicates minimum spanning tree.

(MPs) originally designated for Steiner trees.

• We proposed a combinatorial algorithm for computing disjoint paths, which can be combined with the CP-based algorithm to alleviate its high time complexity. Moreover, we used the combinatorial algorithm as a building block to directly solve the problem and achieve a solution quality better than that of all existing algorithms.

• We proposed a ratio assignment algorithm based on the dual ascending technique, which slightly compromises the solution quality but has a better runtime performance compared to the previous method based on Lagrangian relaxation.

• We performed numerical experiments to evaluate our approach with the results of the top two teams of the 2019 Computer Aided Design (CAD) Contest at International Conference On Computer Aided Design (ICCAD) and the state-of-art algorithms.

The rest of this paper is organized as follows. Section 2 introduces our CP and the combinatorial algorithm for one single group and then presents an algorithm to solve the problem with many groups. Section 3 introduces the dual-ascending algorithm for ratio assignment. Section 4 demonstrates the experimental results. Lastly, Section 5 concludes the paper.

2 Computation of Near-Optimal Topology for the Nets of a Group

In the section, we shall first consider a special case of the problem where there is only one group and accordingly propose the mathematical programming formulation. Then we devise an algorithm for constructing the topology for routing the netlists by employing the minimum Steiner tree algorithm as a key ingredient. Lastly, by taking into consideration that there are many groups, we use the algorithm for one group as a building block to eventually construct the topology for all the groups in iterations.

2.1 MPs for finding the topology of one single group

We present a simple but equivalent version of our problem with one single group and present MPs, solving of which could result in a high quality solution.

Formally, the problem of constructing the topology for routing the netlists of one single group can be stated as follows:

Definition 1 (The soft-disjoint Steiner tree problem, SDST): For a directed graph $G = (V, E)$ with two sets of terminals $S, \hat{S} \subseteq V$, and a set of netlists $\mathcal{N}$ in which each $N \in \mathcal{N}$ is a requirement of connecting a terminal of S to a subset of terminals of $\hat{S}$, the problem is to compute a set of soft-disjoint Steiner trees $\mathcal{T}$, such that there exists a mapping between $\mathcal{N}$ and $\mathcal{T}$, such that $T(N) \in \mathcal{T}$ connects the terminals of netlist $N \in \mathcal{N}$. The aim is to minimize $\sum_{e \in G} \left(\sum_{T: T \supseteq \{e\}, T \in \mathcal{T}} c_T(e) \right)$, where $c_T(e)$ is the cost of edge e that appears on the tree T and $\sum_{T \in \mathcal{T}} \frac{1}{c_T(e)} \leqslant 1$. In particular, $c_T(e) = 0$ holds for $\forall T \in \mathcal{T}$, when e does not appear on any $T \in \mathcal{T}$.

Apparently, we have the following property for the optimization objective:

Lemma 1 For any feasible topology for the set of netlists $\mathcal{N}$, the objective function $\sum_{e \in G} \left(\sum_{T: T \supseteq \{e\}, T \in \mathcal{T}} c_T(e) \right)$ attains the minimum only if $c_T(e) = c_{T'}(e)$ for any pair of T and T' containing e.

Therefore, the objective of SDST can be equivalently stated as follows:

- To minimize $\sum_{e \in G} \left(\sum_{T: T \supseteq \{e\}, T \in \mathcal{T}} c_T(e) \right)^2$ where

$c(e)$ is the number of Steiner trees containing edge e in $\mathcal{T}$.

The SDST problem is NP-hard, because when $|\mathcal{N}| = 1$, i.e., there is only one netlist in $\mathcal{N}$, it reduces to the classical Steiner tree problem that is NP-hard. We can directly devise MPs for SDST following the structure of known Linear Programs (LPs) for Steiner trees as in Ref. [16]. However, it is undesirable to directly determine SDST by solving the programs because of the exponentially many constraints of the MPs inhered from the LPs. Inspired by the mainstream algorithms for computing Steiner trees[16], we shall use soft-disjoint shortest paths as a building block so as to eventually construct SDST.

Definition 2 (The soft-disjoint shortest path problem, SDSP): For a directed graph $G = (V, E)$ with a set of source vertex $S = \{s_1, \ldots, s_k\} \subseteq V$ and a set of destination vertices $\hat{S} = \{t_1, \ldots, t_k\} \in V$, the problem aims to compute a set of soft-disjoint paths $P_1, P_2, \ldots, P_k$ with P_i connecting s_i to t_i, such that $\sum_{e \in G} (c(e))^2$ attains the minimum, for $c(e)$ being the number of times edge e appears on the k paths. In particular, $c(e) = 0$ when e does not appear on any of the $k$paths.

When $c(e) \leqslant 1$ is forced additionally, the problem reduces to the classical disjoint path problem, a fundamental problem in graph theory and networking that remains NP-hard even when $k = 2$[17]. For the case of unconstrained $c(e)$, we propose the following formulation (MP1):

$$\min \quad \sum_{e \in G} \left(\sum_{i=1}^{k} x_i(e) \right)^2 \tag{3}$$

$$\text{s.t.} \quad \forall i: \sum_{e \in \delta^+(v)} x_i(e) - \sum_{e \in \delta^-(v)} x_i(e) = \begin{cases} 1, & v = s_i; \\ -1, & v = t_i; \\ 0, & v \in V \setminus \{s_i, t_i\} \end{cases} \tag{4}$$

$$x_i(e) \in \{0, 1\}$$

where $x_i(e)$ indicates whether e appears on path P_i or not. Constraint (4) is the flow conservation constraint borrowed from flow theory[18].

Lemma 2 The MP1 is convex when $x_i(e)$ is relaxed to [0, 1].

Following the above lemma, we can solve MP1 by employing existing MP solvers, such as the *Gurobi* solver[†]. Moreover, for the performance guarantee of MP1, we have Lemma 3.

Lemma 3 An optimal solution to MP1 is an optimal solution to SDSP.

Thus, when there is only one group and each net within the group contains at most two terminals, we can optimally solve the problem using MP1. We can directly solve MP1 using the *Gurobi* solver and obtain a high-quality solution. However, it requires a high runtime to run the solver against the above MP even when it is with a polynomial number of constraints and variables.

2.2 Fast combinatorial algorithm for finding the topology against netlists

By observing the runtime hardness of the solver-based algorithm, we proposed a fast algorithm by approximating minimum Steiner trees and building up the topology for the netlists. For the task, we first presented a method to assign weight to the edges according to the ratio. Second, we proposed an approach incorporating the exact algorithm and two approximation algorithms for the minimum Steiner tree problem. Lastly, we gave an example for analyzing the gap between our

[†] https://www.gurobi.com/.

solution and the optimum.

2.2.1 Setting the weight of the edges

Considering that the ratio sum of an edge e quadratically depends on the number of topology of net topologies that contain e, we devised a simple method for setting the weight of e to capture the property: (1) We set the weight of each edge e to 2. (2) Once an edge e is routed by a net, we increased the weight of e by 2.

2.2.2 Combinatorial algorithms for computing the topology

In this subsection, we employed two different algorithms for computing the topology of Steiner trees for routing the nets. On the one hand, for the small-size nets, we used the exact algorithm for Steiner trees with a time complexity of $O(3^{|V'|}|V|+2^{|V'|}(|V|\log|V|+|E|))$; on the other hand, for large-size nets, for which the runtime of the exact algorithm is unacceptable, we proposed two improved forms of the factor-2 approximation algorithm to ensure that the algorithm terminates in a reasonable runtime. We only presented the details of our tuned factor-2 approximation algorithm, since the exact algorithm is the same as that in literature.

Algorithm 1 demonstrates the details of our factor-2 approximation algorithm. First, we constructed a set U containing the set of vertices that are already routed; while U' consists of the set of vertices that have not yet been routed (as in Line 1). Inside each loop, we first added auxiliary edges from the source vertex s to vertices of U and the auxiliary edges leaving vertices of U' to t. Then, we employed Dijkstra's algorithm to compute the shortest st-path (Lines 4–7). Lastly, we updated U, U', and E' and the edge weight of G (Lines 8–11). For edge e used in the shortest path, we updated its weight by setting the new weight as $w_e = w_e + 2$.

Algorithm 1 Approximation algorithm for computing steiner trees with increasing edge weights

```
Input: A weighted undirected graph $G(V, E, W)$ with a terminal vertex set $V' \subseteq V$
Output: A set of edges $E'$ for connecting $V'$
1: Select a vertex $v' \in V'$ and set $U \leftarrow \{v'\}'$, while set $U' \leftarrow V' - U$;
2: Set $P \leftarrow \varnothing, U' \leftarrow V' - U$;
3: Add two auxiliary vertices $s$ and $t$;
4: while $U' \neq \varnothing$ do
5:     Add to $E_s$ the edges from $s$ to the vertices in $U$ with weight 0;
6:     Add to $E_t$ the edges from the vertices in $U'$ to $t$ with weight 0;
7:     Find the shortest path from $s$ to $t$ by Dijkstra's algorithm, and store the path in $P$;
8:     Remove the edges of $E_s$ and $E_t$ from $P$;
9:     Select a vertex of $P$ without $s, t$ as $u$;
10:    Set $E' \leftarrow E' + P$;
11:        $U \leftarrow U + u$;
12:        $U' \leftarrow U' - u$;
13:    Update the weight of edges of $P$ through increasing its weight by 2;
14: end while
15: return $E'$.
```

Figure 3 shows the routing process for a 3-FPGA net using our approximation algorithm.

For the time complexity and correctness of Algorithm 1, we have Lemma 4.

Lemma 4 Algorithm 1 runs in $O(n^2 \log n + mn)$ and deserves an approximation ratio of 2.

Proof The approximation ratio can be proven following a similar line as for the factor-2 approximation algorithm for minimum Steiner trees as in Ref. [16].

For the time complexity, the while-loop iterates for at most $O(n)$ times, each of which consumes $O(m + n \log n)$ time for finding the shortest path. Other steps consume a relatively trivial time. Therefore, the total runtime of the algorithm is $O(n^2 \log n + mn)$.

Algorithm 2 illustrates another algorithm based on the shortest disjoint path algorithm that routes multiple nets at the same time. In Lines 1 and 2, we first construct a set U_k and U'_k for each net, which represents the set of routed vertices and the set of vertices that have not yet been routed, respectively. For each loop, in Lines 8–13 we added the auxiliary edge from the source vertex s_k to vertices in U_k and also add the auxiliary edge from vertexes in U'_k to sink vertex t_k for each net. Afterward, we added another s and t connected to all s_k and t_k. Then we used the shortest disjoint path algorithm[19] to find a path P_k for each net. In Lines 15–17, we update U_k, U'_k, and E'_k.

We set a constant M. When the vertex number of the net exceeds M, we used Algorithm 1; otherwise, we used Algorithm 2. In our experiment M is equal to 2. In addition, in the implementation of our algorithms, we used a Fibonacci heap[18] to accelerate the Dijkstra's algorithm. Similar to Lemma 4, we have Lemma 5.

Lemma 5 Algorithm 2 runs in $O(\beta\,(n \log n + m))$ and deserves an approximation ratio of 2, where $\beta = \sum_{V \in \mathcal{V}} |V|$.

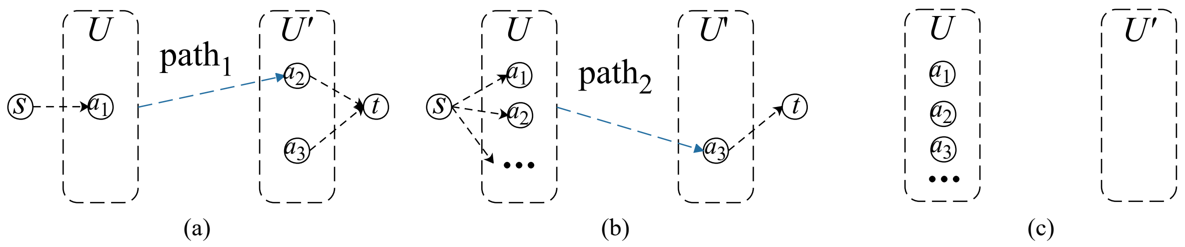

Fig. 3 Routing procession for a 3-FPGA net.

Algorithm 2 Approximation algorithm combining KMB algorithm with shortest disjoint path

```
Input: An undirected and weighted graph $G(V, E, W)$, a set
    of net $N = \{n_1, n_2, \ldots, n_k\}$ with their vertex set $\mathcal{V} =$
    $\{\mathcal{V}_\infty, \mathcal{V}_\in, \ldots, \mathcal{V}_{||}\}$
Output: Sets of connecting edges $\{E'_k\}$
1:  Set $V' = \cap\mathcal{V}$;
2:  For each net, set $U_k = \varnothing$, $U'_k = V_k - U_k$ and $E'_k = \varnothing$;
3:  while $V' \neq \varnothing$ do
4:      $\forall v' \in V'$, $V' \leftarrow V' - v'$;
5:      if $v' \in V_k$, then
6:          $U_k \leftarrow U_k + v'$;
7:          $U'_k \leftarrow U'_k - v'$;
8:          add vertex $s_k$ and a set of edges $E_{sk}$ from $s_k$ to vertices
    in $U_k$
            with weight 0;
9:          add vertex $t_k$ and a set of edges $E_{tk}$ from vertices in
    $U'_k$ to
          $t_k$ with weight 0;
10:         add vertex $s$ and a set of edges $E_s$ from $s$ to all $s_k$
            with weight 0;
11:         add vertex $t$ and a set of edges $E_t$ from all $t_k$ to $t$ with
            weight 0;
12:         $\{P_k\}$ ←shortest disjoint paths from $s$ to $t$;
13:         $\{P_k\} \leftarrow \{P_k - e_{s \to s_k} - E_{sk} - e_{t_k \to t} - E_{tk}\}$;
14:         $u_k$ ←vertices in $P_k$;
15:         $E'_k \leftarrow E'_k + P_k$;
16:         $U_k \leftarrow U_k + u_k$;
17:         $U'_k \leftarrow U'_k - u_k$;
18: end while
19: return $\{E'_k\}$.
```

2.2.3 Bad example

The proposed algorithms cannot guarantee to find an optimal solution even when the requirements are between two vertices (i.e., SDSP) and all source terminals are identical, i.e., $s_i = s_1 = s_2 = \cdots = s_k$. As illustrated in Fig. 4a, the origin graph is presented, with two netlists requiring the connection of s to t. The current cost of each edge is 2, and the remaining bandwidth is uniformly $1/2$. Figure 4b presents the results outputted by the existing algorithm for finding the two nets connecting s to t[20], where the orange path is the first computed path and the green one is the second. Figure 4c presents an optimal solution. Compared with Fig. 4b, Fig. 4c has a smaller total ratio, i.e., 16 whereas that of Fig. 4b is 18, with the shared edge suffering a higher ratio 4 for each sharing. Nonetheless, it is worth noting that the gap between our produced solution and the optimum is small.

3 Ratio Assignment

After computing the topology for the nets, we commence the stage of the TDM ratio assignment. The task is to assign TDM ratio for each net edge, such that the maximum TDM ratio in a group is minimized while the ratio constraints are strictly satisfied.

We can easily model the optimization problem as in the following formulations:

$$\begin{aligned} \min \quad & \max_{P \in \mathcal{P}} \sum_{N \in P} \sum_{e \in N} r_e \\ \text{s.t.} \quad & R_e(r_e) = \sum \frac{1}{r_e} \leqslant 1, \quad \forall e \in E \\ & \operatorname{mod}(r_e, 2) = 0, \quad \forall e \in E \\ & r_e \geqslant 2, \quad \forall e \in E \end{aligned} \tag{5}$$

where r_e and R_e denote the TDM ratio of edge e in net N and the reciprocal of the TDM ratio of the same edge in all nets, respectively. Note that the value of R_e is

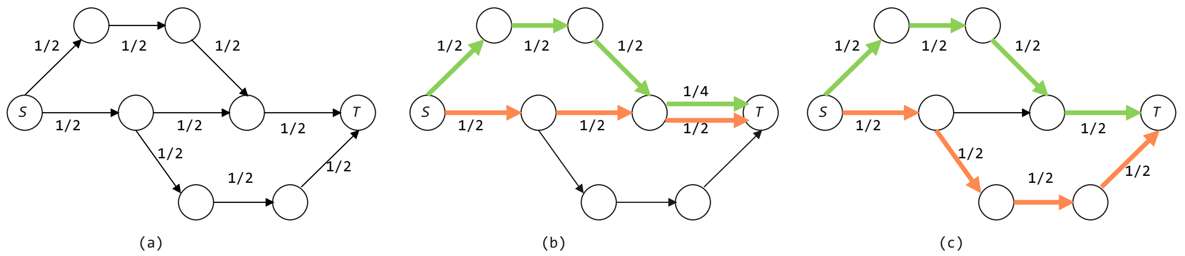

Fig. 4 A solution outputted by our algorithm vs. an optimal solution to SDSP. The orange path is the first computed path and the green one is the second.

bounded by 1.

Because modern circuits have a huge scale, using the ILP-based algorithms[21] for the above assignment problem would take a terrible runtime. Therefore, we proposed a TDM ratio assignment algorithm based on the group demand.

3.1 Group demand based ratio assignment

To prevent the distinct scales between different groups from causing unbalanced ratios, we proposed a ratio assignment scheme based on dual ascending with respect to the group demand. We first defined the demand of the net-group P as follows:

$$d_P = \sum_{N \in P} d_N \tag{6}$$

where d_N is the vertex number of net n.

We used d_P to predict the relative importance of group P based on the observation that a group with a large demand is more likely to eventually have a large TDM ratio. Therefore, we presented the process priority (i.e., to assign ratio) to groups with large demands. More precisely, we sorted all groups according to their demand d_P, and then repeated to find the group with the largest demand and assign ratio to the edges of its nets. The aim was to minimize the maximum ratio of the groups, so we shall assign the ratio of each group in a way that the gap between the largest group ratio and the ratio of any other group is as small as possible.

We initially found the a group with the largest d_N, i.e., $P_{\max}$. Then we set the TDM ratio of edge e in $P_{\max}$ as the number of nets containing e in $P_{\max}$.

$$r_{e,P_{\max}} = \sum_{N \in P_{\max}} 1 \tag{7}$$

Then after the ratio assignment of $P_{\max}$, we assigned the ratio to every other groups according to $r_{e,P_{\max}}$, and for the already processed groups. For a group P, we assumed $\mathcal{N}_a(P)$ and $\mathcal{N}_{na}(P)$ as the sets of nets within P that are assigned and unassigned. We introduce the following formulation for predicting the relationship between P and $P_{\max}$:

$$\text{prio}_P = \frac{r_{P_{\max}} - \sum\limits_{N \in \mathcal{N}_a(P)} r_N}{\sum\limits_{e:\, e \in N_{na} \text{ and } e \text{ appears in } P_{\max}} r_{e,P_{\max}}} \tag{8}$$

where $r_{e,P_{\max}}$ represents the ratio assigned to edge e in $P_{\max}$. Then we calculated the ratio of the edge e in net N as follows:

$$r_{e,N} = r_{e,P_{\max}} \cdot \text{prio} \tag{9}$$

Algorithm 3 summarizes the assignment of the ratio to each edge of each net.

Algorithm 3 Ratio assignment respecting group demands

```
Input: A set of groups $\mathcal{P} = \{P_1, P_2, \ldots, P_l\}$ and a set of net
    $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$, where for each $P \in \mathcal{P}$ we have
    $P \subseteq \mathcal{N}$.
Output: All edges of the nets $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ with
    assigned ratio
1: Sort the groups of $\mathcal{P}$ according to their demand $d_P$ in
   descending order and denote the sorted $\mathcal{P}$;
2: Select the first group of $\mathcal{P}'$, say $p_{\max}$;
3: $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \{P_{\max}\}$;
4: Set $r_{P_{\max}}$ as the TDM ratio of $P_{\max}$;
5: for each $N \in P_{\max}$ do
6:     Assign ratio to each edge of $N$ by Eq. (7);
7:     update $R_N$;
8: end for
9: for each $P \in \mathcal{P}'$ re-ordered by their prio;
10:    for each net $N \in P$ do
11:        if $N$ is not assigned then
12:            $assign$ ratio to each edge routed by $N$ according
                 to Eq. (9);
13:            $update$ $R_N$;
14:        end if
15:    end for
16:    if all nets of $P$ are assigned then
17:        Set $\mathcal{P}' \leftarrow \mathcal{P}' - P$;
18: end for
19: return $\mathcal{R}$.
```

3.2 Ratio legalization

We noted that $\sum\limits_{N \in \mathcal{N}} r_{e,N} > 1$ may hold, which implies a breach of the TDM constraints. For this problem, we needed only to normalize the ratios of $\{r_{e,N} | N \in \mathcal{N}\}$ for each e, such that the TDM ratio constraints are satisfied.

More precisely, for each edge e and for each net N containing e, we set the new ratio as

$$r_{e,N} = \left\lceil \frac{\sum\limits_{N \in \mathcal{N}} \frac{1}{r_{e,N}}}{\frac{1}{r_{e,N}}} \right\rceil \tag{10}$$

After the reassignment of the ratio, we could guarantee the following property:

Lemma 6 For each edge e, we have $\sum\limits_{N \in \mathcal{N}} \frac{1}{r_{e,N}} \leqslant 1$.

Proof The proof can be done via a calculation using Eq. (10):

$$\sum_{N \in \mathcal{N}} \frac{1}{r_{e,N}} = \sum_{N \in \mathcal{N}} \frac{1}{\left\lceil \frac{\sum\limits_{N \in \mathcal{N}} \frac{1}{r_{e,N}}}{\frac{1}{r_{e,N}}} \right\rceil} \leqslant$$

$$\sum_{N \in \mathcal{N}} \frac{1}{r_{e,N} \sum\limits_{N \in \mathcal{N}} \frac{1}{r_{e,N}}} = \frac{1}{\sum_{N \in \mathcal{N}} \frac{1}{r_{e,N}}} \sum_{N \in \mathcal{N}} \frac{1}{r_{e,N}} = 1.$$

There can be odd $r_{e,N}$, for which we need only to set: $r_{e,N} = r_{e,N} + 1$. Because $r_{e,N}$ can only increase during the above operation, $\sum\limits_{N \in \mathcal{N}} \frac{1}{r_{e,N}} \leqslant 1$ remains true. Then, we had completed the assignment of the TDM ratio for all groups satisfying the constraints of the TDM ratio.

4 Simulation Experiments

In this section, we evaluate our routing approach using the benchmarks that simulate multi-FPGA systems provided by the 2019 CAD Contest at ICCAD with statistics parameters as shown in Table 1. The code was implemented using C++ programming language and run on ubuntu 18.04 with Intel-Xeon(R) E5-2620 v4 @ 2.10 GHz CPU and 12 GB memory.

We compared our experimental TDM ratio results and runtime with the top two teams of the 2019 ICCAD CAD contest. As demonstrated in Table 2, our algorithm achieves a better performance compared with "1st place" and "2nd place"—we had achieved an average improvement of 0.1% and 2.6%, respectively, in the min-max TDM ratio. For pursuing a high solution quality, our algorithm has a slightly long runtime, but it is acceptable for the practical usage of large-scale multi-FPGA systems.

5 Conclusion

In this paper, two effective approaches are proposed for solving the system-level routing problem of multi-FPGA systems, aiming to strictly satisfy the TDM constraints while minimizing the maximum delay of the groups. For computing the topology of accommodating the routing requirement by the nets, we presented two approaches: The first is based on our proposed CP formulation and the second is based on the renowned MST-based approximation algorithm but with the enhancement of disjoint path. Then for assigning ratios according to the topology, we designed a group-demand ratio assignment scheme based on dual ascending by reassigning the ratio to strictly meet the TDM constraints. Lastly, we evaluated our algorithms against ICCAD benchmark suites and demonstrate its practical performance gain compared to the previous baselines.

Acknowledgment

This work was supported by the Natural Science Foundation of Fujian Province (No. 2020J01845), the National Natural Science Foundation of China (Nos. 61772005 and 11871280), the Outstanding Youth Innovation Team Project for Universities of Shandong Province (No. 2020KJN008), and Qinglan Project.

References

[1] L. Sun, L. Guo, and P. Huang, System-level FPGA routing for logic verification with time-division multiplexing, in *Proc. 21st Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, Shenzhen, China, 2020, pp. 210–218.

[2] C. Constantinescu, Trends and challenges in VLSI circuit reliability, *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[3] W. N. N. Hung and R. Sun, Challenges in large FPGA-

Table 1 Statistics of benchmarks provided by 2019 ICCAD.

Benchmark	Number of vertexes	Number of edges	Number of nets	Number of groups	Averaged number of vertexes	Averaged number of nets
Synopsys01	43	214	68 456	40 552	2	27
Synopsys02	56	157	35 155	56 308	10	51
Synopsys03	114	350	302 956	334 652	11	36
Synopsys04	229	1087	551 956	464 867	11	37
Synopsys05	301	2153	881 480	879 145	11	37
Synopsys06	410	1852	785 539	910 739	48	54
Hidden01	73	289	54 310	50 417	17	54
Hidden02	157	803	610 675	501 594	31	54
Hidden03	487	2720	720 520	886 720	50	54

Table 2 Performance comparison with the other two schemes comparison with the other two schemes.

Team	Benchmark									
	Synopsys01	Synopsys02	Synopsys03	Synopsys04	Synopsys05	Synopsys06	Hidden01	Hidden02	Hidden03	Normalized
2nd place	**40 190**	32 093 856	129 290 206	**6 334 724**	**4 613 110**	15 759 302 934	409 891 196	45 952 594 974	4 872 933 150	1.001
1st place	40 498	32 006 748	128 206 488	7 049 316	5 190 132	15 751 028 904	409 300 116	45 942 030 192	4 867 325 852	1.026
Ours	40 436	**31 630 660**	**127 132 204**	6 419 814	4 672 398	**15 743 366 622**	**408 575 164**	**45 933 427 024**	**4 860 390 872**	**1.000**

based logic emulation systems, in *Proc. 2018 Int. Symp. on Physical Design*, Monterey, CA, USA, 2018, pp. 26–33.

[4] S. Asaad, R. Bellofatto, B. Brezzo, C. Haymes, M. Kapur, B. Parker, T. Roewer, P. Saha, T. Takken, and J. Tierno, A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multi-core processor simulation, in *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, Monterey, CA, USA, 2012, pp. 153–162.

[5] W. T. Wang, Z. X. Shen, and V. Dinavahi, Physics-based device-level power electronic circuit hardware emulation on FPGA, *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2166–2179, 2014.

[6] P. S. Graham, Logical hardware debuggers for FPGA-based systems, PhD dissertation, Brigham Young University, Provo, UT, USA, 2001.

[7] G. Schelle, J. Collins, E. Schuchman, P. Wang, X. Zou, G. Chinya, R. Plate, T. Mattner, F. Olbrich, P. Hammarlund, et al., Intel nehalem processor core made FPGA synthesizable, in *Proc. 18th Annual ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, Monterey, CA, USA, 2010, pp. 3–12.

[8] J. H. Han, Z. L. Li, W. M. Zheng, and Y. H. Zhang, Hardware implementation of spiking neural networks on FPGA, *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, 2020.

[9] J. Q. Huang, W. T. Han, X. Y. Wang, and W. G. Chen, Heterogeneous parallel algorithm design and performance optimization for WENO on the Sunway Taihulight supercomputer, *Tsinghua Science and Technology*, vol. 25, no. 1, pp. 56–67, 2020.

[10] M. Inagi, Y. Takashima, and Y. Nakamura, Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems. in *Proc. 2009 Int. Conf. on Field Programmable Logic and Applications*, Prague, Czech Republic, 2009, pp. 212–217.

[11] J. Babb, R. Tessier, and A. Agarwal, Virtual wires: Overcoming pin limitations in FPGA-based logic emulators, in *Proc. 1993 IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, USA, 1993, pp.142–151.

[12] M. Inagi, Y. Takashima, Y. Nakamura, and A. Takahashi, Optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA prototyping systems, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E91.A, no. 12, pp. 3539–3547, 2008.

[13] C. W. Pui, G. Wu, F. Y. C. Mang, and E. F. Y. Young, An analytical approach for time-division multiplexing optimization in multi-FPGA based systems, in *Proc. 2019 ACM/IEEE Int. Workshop on System Level Interconnect Prediction*, Las Vegas, NV, USA, 2019, pp. 1–8.

[14] T. W. Lin, W. C. Tai, Y. C. Lin, and I. H. R. Jiang, Routing topology and time-division multiplexing co-optimization for multi-FPGA systems. in *Proc. 2020 57th ACM/IEEE Design Automation*, San Francisco, CA, USA, 2020, pp. 1–6.

[15] M. Inagi, Y. Takashima, and Y. Nakamura, Globally optimal time-multiplexing of inter-FPGA connections for multi-FPGA prototyping systems, *IPSJ Trans. Syst. LSI Des. Methodol.*, vol. 3, pp. 81–90, 2010.

[16] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th ed. Berlin, Germany: Springer, 2012.

[17] S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem, *Theor. Comput. Sci.*, vol. 10, no. 2, pp. 111–121, 1980.

[18] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan, Faster algorithms for the shortest path problem, *J. ACM*, vol. 37, no. 2, pp. 213–223, 1990.

[19] J. W. Suurballe and R. E. Tarjan, A quick method for finding shortest pairs of disjoint paths, *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[20] P. Zou, Z. F. Lin, X. Shi, Y. J. Wu, J. L. Chen, J. Yu, and Y. W. Chang, Time-division multiplexing based system-level FPGA routing for logic verification, in *Proc. 2020 57th ACM/IEEE Design Automation*, San Francisco, CA, USA, 2020, pp. 1–6.

[21] M. Inagi, Y. Takashima, Y. Nakamura, and A. Takahashi, ILP-based optimization of time-multiplexed I/O assignment for multi-FPGA systems, in *Proc. 2008 IEEE Int. Symp. on Circuits and Systems*, Seattle, WA, USA, 2008, pp. 1800–1803.

Peihuang Huang received the BS degree in mathematics from Fujian Normal University, Fuzhou, China, in 2005, and the PhD degree in intelligent systems from Fuzhou University, Fuzhou, China, in 2019. She is currently an associate professor at the College of Mathematics and Data Science, Minjiang University, Fuzhou, China. She has published more than 20 academic papers with major research interest including sensor networks and networking science.

Long Sun received the BS degree in computer science from Fuzhou University in 2018. His major research interest includes algorithm design for field-programmable gate array (FPGA) and very-large-scale integration (VLSI), combinatorial optimization and graph algorithm. He is currently pursuing the ME degree of computer science at Fuzhou University, Fuzhou, China.

Longkun Guo received the BS and PhD degrees in computer science from University of Science and Technology of China (USTC), Hefei, China, in 2005 and 2011, respectively. He was a research associate of the University of Adelaide, Adelaide, Australia, from 2015 to 2016, and is currently a full professor at the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. He has published more than 80 academic papers in reputable journals/conferences such as *Algorithmica*, *IEEE TMC*, *IEEE TC*, *IEEE TPDS*, IEEE ICDCS, IJCAI, and SPAA. His major research interest includes efficient algorithm design and computational complexity analysis, particularly for optimization problems in high performance computing systems and networks, and very-large-scale integration (VLSI) etc.

Xiaoyan Zhang received the first PhD degree in applied mathematics from Nankai University, Tianjin, China, in 2006 and received the second PhD degree in computer science from Twente University, Enschede, the Netherlands, in 2014. He is currently a full professor in the School of Mathematics Science and the Institute of Mthematics, Nanjing Normal University, Nanjing, China. He has published more than 50 academic papers in reputable journals such as *Siam J. Computing*, *Siam J. Scientific Omputing*, *Siam J. Discrete Math*, and *J. Gtaph Theoty*. His major research interest includes efficient algorithm design and computational complexity analysis, particularly for combinatorial optimization, graph algorithms, networks, and VLSI.