Multi-FPGA Co-optimization: Hybrid Routing and Competitive-based Time Division Multiplexing Assignment

Dan Zheng*
The Chinese University of Hong Kong
Hong Kong SAR
dzheng@cse.cuhk.edu.hk

Xiaopeng Zhang*
The Chinese University of Hong Kong
Hong Kong SAR
xpzhang@cse.cuhk.edu.hk

Chak-Wa Pui
The Chinese University of Hong Kong
Hong Kong SAR
cwpui@cse.cuhk.edu.hk

Evangeline F.Y. Young
The Chinese University of Hong Kong
Hong Kong SAR
fyyoung@cse.cuhk.edu.hk

ABSTRACT

In multi-FPGA systems, time-division multiplexing (TDM) is a widely used technique to transfer signals between FPGAs. While TDM can greatly increase logic utilization, the inter-FPGA delay will also become longer. A good time-multiplexing scheme for inter-FPGA signals is very important for optimizing the system performance. In this work, we propose a fast algorithm to generate high quality time-multiplexed routing results for multiple FPGA systems. A hybrid routing algorithm is proposed to route the nets between FPGAs, by maze routing and by a fast minimum terminal spanning tree method. After obtaining a routing topology, a two-step method is applied to perform TDM assignment to optimize timing, which includes an initial assignment and a competitive-based refinement. Experiments show that our system-level routing and TDM assignment algorithm can outperform both the top winner of the ICCAD 2019 Contest and the state-of-the-art methods. Moreover, compared to the state-of-the-art works [17, 22], our approach has better run time by more than 2× with better or comparable TDM performance.

1 INTRODUCTION

In recent years, systems with multiple field-programmable gate arrays (FPGAs) become very popular in applications that require high efficiency and frequent modifications, such as deep learning [13], data center [4], logic emulation and rapid prototyping of large designs [5], etc. In multi-FPGA systems, different FPGAs are connected by direct hardwired connections or programmable interconnection networks [12].

*These authors contributed equally to this work.

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 14202218 and CUHK 14209320).

ASPDAC '21, January 18–21, 2021, Tokyo, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431565

In multi-FPGA systems, the utilization of logic resources is limited by the routing resources between FPGAs. Time-division multiplexing (TDM) is a method that multiplexes the use of the I/O pins and inter-FPGA wires for inter-FPGA signals [1]. Since this technique can effectively increase the number of logical pins, the resulting logic resource utilization can be improved. However, each time-multiplexed signal has to wait for its turn of transmission and the inter-FPGA delay is lengthened.

In a modern compilation flow as shown in Figure 1, the negative effects of time-multiplexing on delay can be reduced by taking TDM delay into account during partitioning, routing and post-routing[6]. A recent work [3] proposes a framework that performs TDM assignment and partitioning simultaneously, which results in much faster system clock frequency compared to ordinary cut-driven partitioning. There are several works that optimize TDM ratios in the post-routing stage. TDM ratio is a metric related to the delay of an inter-FPGA net and its definition will be described in details in Section 2. The works [7–10] formulate the TDM assignment problem as an integer linear program (ILP), trying to put non-critical inter-FPGA nets in TDM wires to improve logic utilization without affecting the system timing. The works [18–20] propose a two-step analytical framework to solve the TDM assignment problem in modern multi-FPGA systems. These works consider all the practical TDM constraints and hence will be time consuming.

Routings for ASICs and FPGAs have been extensively studied [2, 14, 15], and many focus on wirelength reduction and removal of design rule violations. However, in system-level routing, we need to consider the overall performance of the whole system. For multi-FPGA systems, not only wirelength, TDM assignment is also very important since the system performance is largely affected by the delay of inter-FPGA nets. A recent work [17] explored this problem by a timing-aware ratio assignment algorithm. Another recent work [22] performed TDM assignment by Lagrangian relaxation, which achieved good TDM ratio but suffered from long run time.

In this work, we propose an effective algorithm that can generate time-multiplexed routing schemes for large scale multi-FPGA systems. The major contributions are summarized as follows:

- A system-level routing and TDM assignment algorithm is proposed to optimize timing of multi-FPGA systems, which considers both wirelength and TDM ratios. Most parts of our

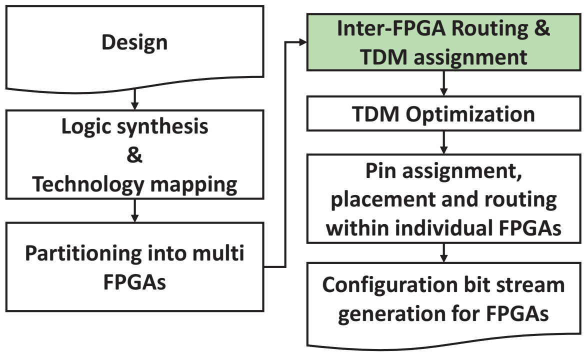

Design
Logic synthesis & Technology mapping
Partitioning into multi FPGAs
Inter-FPGA Routing & TDM assignment
TDM Optimization
Pin assignment, placement and routing within individual FPGAs
Configuration bit stream generation for FPGAs

Figure 1: A typical compilation flow for multi-FPGA systems [11] with TDM technique.

algorithms are designed to be multi-thread friendly, which can be parallelized with little or no quality loss.

- A hybrid routing algorithm is proposed to generate the routing topology of each net. In particular, we use maze routing to route the nets that are timing critical. The remaining nets are routed by a fast minimum terminal spanning tree method in parallel.
- A fast and effective two-step algorithm is proposed to determine the TDM ratio of each inter-FPGA routing edge.
- Experimental results show that our algorithm not only can satisfy all the TDM constraints but also has better performance than the top winner of the ICCAD 2019 Contest and the state-of-the-art works [17][22].

The remainder of this paper is organized as follows. Section 2 gives the preliminaries of the problem. Sections 3-5 first give an overview of our approach and then introduce the details. Section 6 shows the experiment results, and the conclusion is in Section 7.

2 PRELIMINARIES

In this section, we will first explain our target architecture. The problem definition will then be introduced. The notations used in the rest of this paper are shown in Table 1. Note that in our terminology, *wire* refers to the connection between two FPGAs while *edge* refers to the routing edge of a net. Besides, only inter-FPGA nets are considered in our problem.

2.1 Target Architecture

We consider a multi-FPGA system with time-multiplexed hardwired inter-FPGA connections where two FPGAs are adjacent logically if they are directly connected in the system. In our target system, two adjacent FPGAs are called an FPGA-pair.

Since the number of nets is much larger than the number of physical wires between FPGAs, time-multiplexed wires are used to connect different FPGAs. In such systems, each inter-FPGA net is assigned a TDM ratio that represents how many other nets are sharing one physical wire with this net. Only the nets in the same direction and with the same TDM ratio can be assigned to the same physical wire. When estimating the delay of a signal in our target architecture, the worst case scenario is assumed, i.e., an inter-FPGA signal needs to wait for an entire TDM cycle for its turn to be transmitted. Hence, the transmission delay is proportional to the TDM ratio of the wire. Due to architectural limitations, a TDM ratio can only be an even number in this work.

Table 1: Notations

e_i	The routing edge i in the routing graph.
E_i	The set of routing edges of net i.
E^w_{ij}	The set of routing edges through wire e^w_{ij}.
E^g_i	The set of routing edges belonging to net group i, which is $\bigcup_{net_j \in N_i} E_j$.
e^w_{ij}	The wire between FPGA i and FPGA j.
E^w	The set of wires of the FPGA system.
x_i	The TDM ratio of edge e_i.
G^n	The set of net groups.
N_i	The set of nets in net group i.
N	The set of all nets.
u_{ij}	The usage of wire e^w_{ij}, which is $\sum_{e_k \in E^w_{ij}} \frac{1}{x_k}$
r_k	The total TDM ratio of the routing edges in net group k and $r_k = \sum_{net_j \in N_k} \sum_{e_i \in E_j} x_i$.
r_{max}	The maximum total TDM ratio of all net groups such that $r_{max} = \max_{grp_k \in G^n}(r_k)$.

2.2 Problem Definition

In system-level routing, given a netlist and the FPGA-connectivity, we need to decide the routing topology of each net and estimate a TDM ratio of each inter-FPGA edge such that the system performance is maximized. To optimize timing, the total delay of the most critical signal paths in the netlist should be minimized. To model this, a number of net groups are defined, and each net group represents the set of nets on a critical path. Note that, every net belongs to at least one group. Since the delay of a signal path is mostly determined by the delays of the inter-FPGA signals on the path, it can be estimated by the total TDM ratio of the nets in the corresponding net group. A metric called *maximum group TDM sum* r_{max} is defined and is the objective to be minimized, as shown in Equations (1b)–(1c). In practice, the TDM ratios should satisfy all the constraints mentioned in Section 2.1. However, in system-level routing, we need a very fast method to estimate the timing of a routing solution, which can then be used to provide feedback for the partitioning step or the routing step. Therefore, in our problem formulation, as shown below in Equation (1), the limit on the actual number of physical wires in an FPGA-pair is modelled by Equation (1d), based on the number of nets between the FPGA-pair

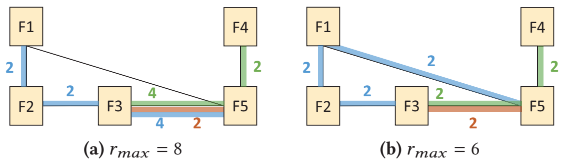


Figure 2: An example of the system-level routing problem.

and their TDM ratios.

$$\min \quad r_{max} \tag{1a}$$

$$s.t. \quad r_k = \sum_{net_i \in N_k} \sum_{e_j \in E_i} x_j, \quad \forall grp_k \in G^n \tag{1b}$$

$$r_{max} = \max_{grp_k \in G^n} r_k \tag{1c}$$

$$\sum_{e_k \in E^w_{ij}} \frac{1}{x_k} \le 1, \quad \forall e^w_{ij} \in E^w \tag{1d}$$

$$x_k \text{ is an even number} \tag{1e}$$

$$\text{Every net is connected by its routing edges.} \tag{1f}$$

An example is shown in Figure 2. In this example, the black lines denote the connections between FPGAs, while the blue, green and red lines represent three nets, net_1, net_2 and net_3 respectively. There are three net groups: grp_1 contains net_1, grp_2 contains net_2 and net_3, and grp_3 contains net_3. Given the routing scheme in Figure 2a, the optimal TDM assignment will result in $r_{max} = r_{grp_1} = 2+2+4 = 8$. However, if we move the routing edge of net_1 from $e^w_{3,5}$ to $e^w_{1,5}$ (Figure 2b), according to the wire usage constraints ($u_{ij} = \sum_{e_k \in E^w_{ij}} \frac{1}{x_k} \le 1$), the TDM ratio of the edge of net_2 between FPGA3 and FPGA5 can be reduced from 4 to 2. As a result, the optimal r_{max} ($r_{max} = r_{grp_1} = 2+2+2 = 6$) can be reduced to 6.

3 OVERVIEW

The overall flow of our algorithm, as shown in Figure 3, can be divided into two parts, routing and TDM assignment. During routing, the routing topology is generated such that the pins of each net are connected and the routing edges are distributed evenly among the connections between FPGAs. Given the routing result, we will determine the TDM ratio of each routing edge such that r_{max} is minimized. To be specific, a fast and effective method is first applied to assign an initial TDM ratio for each routing edge such that the resulted r_{max} is relatively good. Legalization is then performed to remove all constraint violations while disturbance to the initial assignment result is minimized. Finally, the TDM ratios will be further refined.

4 ROUTING

Given a netlist and the connections between the FPGAs, we need to find the routing topology of each net. As can be seen from Equation (1), both routing congestion and long wirelength will result in a large r_{max}. To balance routing congestion and wirelength, we set the cost of routing through a wire e^w_{pq} as in Equation (2).

$$cost_{pq} = \frac{|E^w_{pq}| \cdot |E^w|}{\sum_{e^w_{ij} \in E^w} |E^w_{ij}|} + C, \tag{2}$$

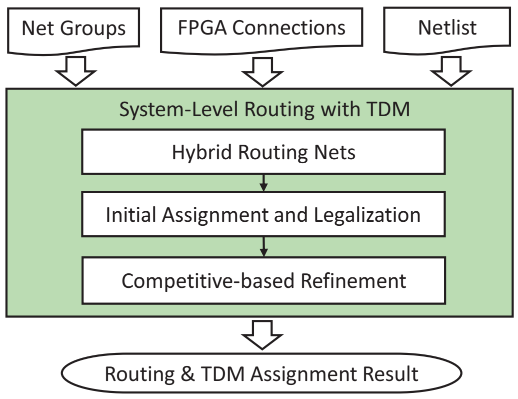

Net Groups
FPGA Connections
Netlist
System-Level Routing with TDM
Hybrid Routing Nets
Initial Assignment and Legalization
Competitive-based Refinement
Routing & TDM Assignment Result

Figure 3: Overall flow of our system-level routing and TDM assignment algorithm.

In Equation (2), the first item reflects congestion and is computed as the ratio between the number of routing edges on wire e^w_{pq} and the average number of routing edges on each wire. If this number is less than one, the wire is not congested and the cost of routing on this wire is small. The second term C is used to prevent long path length, which is set to be 10^5 in our implementation.

In our routing algorithm, all the routing groups will be classified as *dominant groups* or *non-dominant groups* according to the following procedure. First of all, all the routing groups will be sorted in a non-descending order of their total numbers of pins. The smallest index i is found such that the i^{th} group in this sorted list has its number of pins less than a fraction $\frac{1}{a}$ of that of the $(i+1)^{th}$ group. Then all the groups with indexes $i+1$ or above will be classified as dominant groups while others are non-dominant groups. In our implementation, the parameter a is set to 50. A dominant group will have many more pins than any non-dominant group. In our hybrid routing algorithm, we will first find the dominant and non-dominant groups according to their numbers of pins. To route all the nets, we have two schemes: maze routing and fast minimum terminal spanning tree (MTST) method. Since the nets in dominant groups are more timing critical, they are first routed using maze routing, which can achieve better quality compared to the fast MTST method. The rest of the nets are then routed by the fast MTST method, which can effectively speed up the routing process with little quality loss. Details of these two routing approaches will be discussed in the following sections.

4.1 Maze Routing

Maze routing is a widely adopted method [2, 14, 15] for its high flexibility and routability. In this work, we adopt the maze routing scheme from an open-source detailed router called Dr.CU [2]. For a multi-pin net, path search starts from a source pin. When reaching the first unvisited pin, all vertices on the path are regarded as source for searching the next unvisited pin, until all the pins are reached.

Algorithm 1 Fast MTST Method

```
Require: Non-dominant nets and the routing graph.
Ensure: Every non-dominant net is connected by its routing edges.
 1: evenly divide all nets into $m$ batches in a random manner;
 2: for each batch do
 3:     find the shortest paths of all FPGA pairs;
 4:     for each $net_i$ in the batch do
 5:         construct a complete graph $g_c$ for the pins of $net_i$;
 6:         find an MST in $g_c$;
 7:         unfold the MST edges into the corresponding shortest
            paths found on line 3;
 8:     end for
 9:     update the wire usages;
10: end for
```

4.2 Fast MTST Method

The nets belonging to non-dominant groups are named *non-dominant nets*. Compared to those in dominant groups, non-dominant nets have little impact on the objective value. Instead of using maze routing, we adopt a more efficient scheme called fast MTST, which is inspired by MTST [16]. We also propose a multi-threaded scheme to find a Steiner minimum tree (SMT) in a general graph quickly. Compared to the original MTST, our fast MTST is more compatible with multi-threaded implementation.

Algorithm 1 shows the details of our fast MTST method. First, the non-dominant nets are evenly divided into m batches in a random manner. For each batch, we use the Floyd-Warshall algorithm to find the shortest path between every FPGA pair in the system. As shown on lines 5-7, for each net in the batch, we will first construct a complete graph g_c of its pins and the edge weight is set to be the cost of the shortest path between the corresponding FPGA pair obtained by the Floyd-Warshall algorithm. A minimum spanning tree (MST) is then found on g_c, and the edges in the MST will be unfolded into their corresponding shortest paths. Note that, after unfolding, repeated routing edges will be eliminated.

Figure 4 gives an example of our fast MTST method. As shown in Figure 4a and 4b, the shortest paths of all the FPGA pairs are found and recorded. Given a net with three pins (F1, F3, and F8), a complete graph is constructed from the result of Figure 4b. An MST is then constructed on the complete graph, which is shown as blue lines in Figure 4c. Finally, the edges on the MST are unfolded into their corresponding shortest paths and repeated edges (between F3 and F6 in Figure 4d) are removed.

4.3 Parallel Implementation

Most parts of our routing algorithm can be parallelized with little quality degradation. For the dominant nets, maze routing can be parallelized among different nets but the wire usage should be updated in a synchronized manner. For the non-dominant nets, the batches are executed sequentially but the Steiner tree construction within the same batch can be parallelized. For each batch, the wire usage will be updated after the routing edges of all the nets are found. In our experiments, by using 8 threads, our routing algorithm can be 3X faster on average and can achieve up to 6X speedup in some designs. Besides, compared to the original MTST [16] in the same environment of 8 threads, our fast MTST has better run time by 1.9X with the same quality.

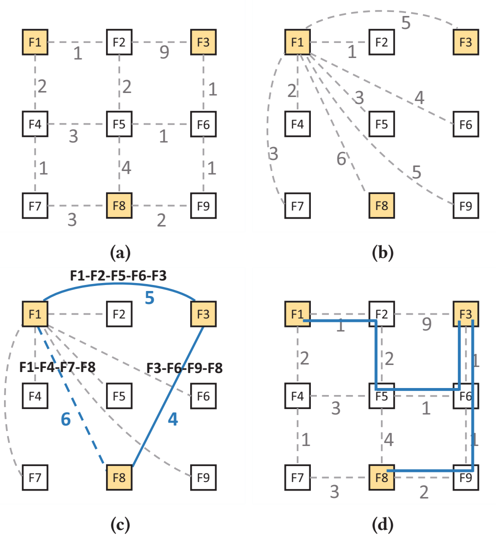


Figure 4: An example of the fast MTST method. (a) shows 9 FPGAs and the weights of the wires. (b) shows the shortest paths between F1 and the other FPGAs. (c) shows the complete graph constructed from (a) and the resulted MST. (d) shows the unfolded paths of the MST in (c).

5 TDM ASSIGNMENT

5.1 Initial Assignment

Since the objective is to minimize r_{max} which is the maximum TDM sum r_k among all the net groups, our method will calculate the TDM ratio of each routing edge from the net groups' perspective. Details of our initial assignment are shown in Algorithm 2. In our algorithm, there is a target TDM sum b, which represents the estimated r_{max} and will guide the TDM assignment of each routing edge. To initialize the TDM ratio of each routing edge, we will set the initial TDM sum target b to a large number, which is 5×10^7 in our implementation. On lines 5-7, we first assume that the routing edges in the same net group grp_i have the same TDM ratio ($x_i^g = \frac{b}{|E_i^g|}$). However, for routing edges belonging to several net groups, the x_i^g of those groups may vary. To minimize r_{max}, the TDM ratio of each routing edge is set to be the minimum x_i^g of the groups it belongs to. As shown on line 9, the target TDM sum b will be multiplied by the average usage of the wires at each iteration such that the usage of each wire will gradually converge to one. According to the experimental results, our algorithm usually converges in two iterations and the resulted r_{max} is relatively good compared with the r_{max} obtained in the first iteration.

5.2 Legalization

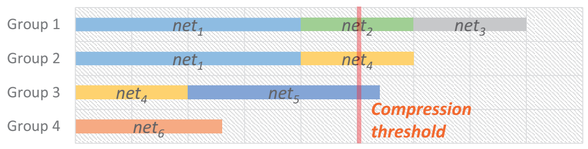
After the initial assignment step, a TDM ratio of integral type will have been assigned to each routing edge, which might not be a legal assignment according to the problem formulation (Section 2.2). A

Algorithm 2 Initial Assignment

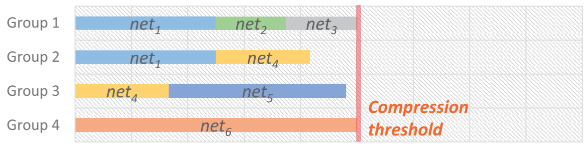
```
Require: The routing topology.
Ensure: An optimized and almost legalized TDM assignment result.
 1: let the TDM ratio $x_j$ of each routing edge be $\infty$;
 2: let the TDM sum target $b$ be a large number;
 3: let the threshold $\sigma$ be a small number;
 4: do
 5:     for each $grp_i \in G^n$ do
 6:         $x_j = \min(x_j, \frac{b}{|E_i^g|}), \quad \forall e_j \in E_i^g$;
 7:     end for
 8:     $u_{pq} = \sum_{e_k \in E_{pq}^w} \frac{1}{x_k}, \forall e_{pq}^w \in E^w$;
 9:     $b = b \cdot \frac{\sum_{e_{ij}^w \in E^w} u_{ij}}{|E^w|}$;
10: while $\exists e_{pq}^w \in E^w, |u_{pq} - 1| > \sigma$
```

Algorithm 3 Competitive-based Refinement (CR)

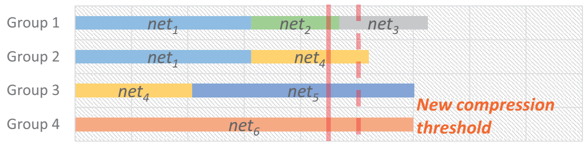
```
Require: The TDM ratio $x_j$ of each routing edge.
Ensure: A legal and optimized TDM assignment result.
 1: do
 2:     calculate the compression threshold $t = r_{max} \cdot \beta$;
 3:     let the compression ratio $c_i$ of each net $i$ be $\infty$;
 4:     for each $grp_k \in G^n$ do
 5:         $c_i = \min(c_i, \frac{t}{r_k}), \quad \forall net_i \in N_k$;
 6:     end for
 7:     for each $net_i \in N$ do
 8:         $x_j = x_j \cdot c_i, \quad \forall e_j \in E_i$;
 9:     end for
10:     apply legalization to each wire;
11: while the improvement in $r_{max}$ is not small
```



(a) Initial TDM sums and the target.



(b) The large groups are compressed.



(c) All groups are enlarged by legalization.

Figure 5: Examples of the CR algorithm.

legal assignment requires that the TDM ratios must be even numbers and the wire usage constraints ($u_{ij} = \sum_{e_k \in E_{ij}^w} \frac{1}{x_k} \leq 1$) must be satisfied. The assignment will be legalized in the legalization step. The TDM ratio of each routing edge is first scaled linearly by multiplying with the usage of the wire u_{ij} as shown in Equation (3).

$$x'_k = x_k \cdot u_{ij}, \quad \forall e_k \in E_{ij}^w, \forall e_{ij}^w \in E^w. \tag{3}$$

After scaling, the usage of each wire will be exactly one, satisfying the wire usage constraint:

$$\sum_{e_k \in E_{ij}^w} \frac{1}{x'_k} = \sum_{e_k \in E_{ij}^w} \frac{1}{x_k \cdot u_{ij}} = \frac{1}{u_{ij}} \cdot \sum_{e_k \in E_{ij}^w} \frac{1}{x_k} = 1, \forall e_{ij}^w \in E^w. \tag{4}$$

The TDM ratio of each routing edge will then be rounded to the smallest even number not less than the current value.

5.3 Competitive-based Refinement

In the following, we call the groups that have much larger r_k than the others as *critical groups*. The total TDM ratio r_k of a net group grp_k is determined by the number of routing edges in it and the TDM ratios of these edges. Without loss of generality, in a good TDM assignment result, there should be a large number of groups whose r_k are close to r_{max}. Otherwise, some routing edges in the non-critical groups might have been assigned too much wire resources. In such cases, we can usually transfer the wire resources from non-critical groups to critical groups such that the TDM ratios of the routing edges in critical groups can become smaller and the resulted r_{max} can be reduced. Since the basic idea of our refinement process is that routing edges compete for wire resources , we name our method competitive-based refinement (CR).

Details of CR are described in Algorithm 3. As shown on line 2, the groups with r_k larger than a compression target t are regarded as critical groups, where t is computed as $(r_{max} \cdot \beta)$ and $\beta(\leq 1)$ will increase based on the iteration number. On lines 4-9, the TDM ratios of the edges in critical groups will be reduced while the others will be increased. In other words, for the routing edges belonging to net groups with large r_k, their TDM ratios will be reduced since more wire resources should be assigned to them. In specific, for the routing edges in net i, we will use a scalar c_i, called *compression ratio*, to change their TDM ratios such that the r_k of each group k will not exceed t after the adjustment. However, the wire usage constraints ($\sum_{e_k \in E_{ij}^w} \frac{1}{x_k} \leq 1$) may be violated after such adjustment. To remove these violations, the legalization method mentioned in Section 5.2 will be applied. Note that, due to the characteristics of our legalization method, the relative TDM ratios among different routing edges will be maintained.

Figure 5 demonstrates the idea of Algorithm 3 in a simple example that consists of 4 net groups sharing 6 nets. The bars with different colors represent different nets, and the lengths of which represent their TDM sum. Figure 5a shows the result after initial assignment and legalization. To reduce r_{max}, the nets in the groups with TDM sum larger than the compression threshold t will be compressed (TDM ratios decrease), while the other nets will be extended (TDM ratios increase). From Figure 5b, we can see that net $1-5$ are compressed while net 6 is extended. Then, the legalization algorithm is performed on each wire to satisfy the wire usage constraint. The result is shown in Figure 5c. Compared with Figure 5a, r_{max} in Figure 5c is reduced significantly.

Table 2: Statistics of benchmarks.

Design	#FPGAs	#Wires	#Nets	#NetGroups
synopsys01	43	214	68456	40552
synopsys02	56	157	35155	56308
synopsys03	114	350	302956	334652
synopsys04	229	1087	551956	464867
synopsys05	301	2153	881480	879145
synopsys06	410	1852	785539	910739
hidden01	73	289	54310	50417
hidden02	157	803	610675	501594
hidden03	487	2720	720520	886720

5.4 Parallel Implementation

Note that our TDM assignment algorithm can be parallelized without losing any quality. In the initial assignment, the calculations of the TDM ratio x_i can be parallelized among all routing edges since b and $|E_i^g|$ are fixed at the time of computation. In legalization, the usage of each wire can also be obtained in parallel among all the wires because both the routing topology and the TDM ratio of each routing edge are determined at the time of execution. In the competitive-based refinement, the TDM sum r_k of each group k can also be calculated in parallel. Since a net may belong to several net groups, to avoid repeated calculation, the sum of the TDM ratios of each net is first computed in parallel. The sum of the TDM ratios of each net group can then be calculated by summing up the TDM ratios of its nets, which can also be performed in parallel. In our experiments, by using eight threads, our TDM assignment algorithm can be 3X faster on average and can achieve up to 5X speedup for some designs.

6 EXPERIMENTAL RESULT

In this work, all algorithms are implemented in C++ and tested on a Linux workstation with an Intel Xeon 2.2GHz CPU with 20 cores and 256GB memory.

6.1 Results on ICCAD'19 Contest Benchmarks

The ICCAD 2019 Contest benchmark [21] is used to evaluate the performance of our algorithm. The benchmark statistics are shown in Table 2.

In our experiment, we compare our proposed algorithm with the state-of-the-art works [17, 22] and the winner of the ICCAD 2019 Contest. The binaries of these methods under comparison are provided by the authors respectively, and are run in the same computing environment of 8 threads for fair comparisons. To quantify the performance, we employ the evaluation score used by the ICCAD 2019 Contest [21], as follows:

$$Score = r_{max} \times (\log_2(\frac{x}{X}) \times 0.01 + 1), \tag{5}$$

where x represents the runtime, while X is the medium runtime of all contestants, which is released by the contest organizer [21]. Note that a lower evaluation score implies a better performance.

The comparisons are shown in Table 3. We can see that our proposed algorithm achieves the best final scores which imply the best overall performance. Our algorithm also obtains the best evaluation scores for most test cases. More specifically, our method can achieve 4% better TDM ratio than the contest winner. Compared to the state-of-the-art work [22], we can achieve more than 6× better in run time and with comparable TDM ratio performance. Compared to the state-of-the-art work [17], we achieve more than 2× better run time with a slightly better TDM ratio performance.

6.2 Results on New Benchmarks

For data sets without dominant groups such as case4 and case5, the optimization is more difficult since the most critical group can vary. In order to evaluate the performance of different methods on this kind of benchmarks, we created four new benchmarks based on the original benchmarks by removing the dominant group. As a result, the number of pins in each group is close to each other. These new benchmarks are named "original benchmark_ex".

For these new benchmarks, we compare our method with [17, 22] and the contest winner. The results are shown in Table 4. We can observe that our algorithm obtains better TDM ratio for each case than the work [22] while the run time is 9× faster on average. Compared with the work the contest winner and [17], our run time is longer but we can produce much better TDM ratio assignment for each case. On average, our TDM ratio performance is 11.4% and 11.3% better than that of the contest winner and [17] respectively.

7 CONCLUSION

In this work, we propose an algorithm that can produce a time-multiplexed routing result for large scale FPGA systems. In particular, a hybrid routing algorithm is proposed to route the nets between FPGAs, which includes maze routing and the fast MTST method. Given the routing topology, a two-step method is used to produce a legal TDM assignment result with optimized timing, which includes initial TDM assignment and competitive-based refinement. Compared with the winner of the ICCAD'19 contest and the state-of-the-art works, experimental results show that our system-level routing and TDM assignment algorithm is effective.

REFERENCES

[1] Jonathan Babb, Russell Tessier, Matthew Dahl, Silvina Zimi Hanono, David M Hoki, and Anant Agarwal. 1997. Logic emulation with virtual wires. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 16, 6 (1997), 609–626.

[2] Gengjie Chen, Chak-Wa Pui, Haocheng Li, and Evangeline FY Young. 2019. 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* (2019).

[3] Shih-Chun Chen, Richard Sun, and Yao-Wen Chang. 2018. Simultaneous partitioning and signals grouping for time-division multiplexing in 2.5 D FPGA-based systems. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 4.

[4] George A. Constantinides. 2017. FPGAs in the Cloud. In *Proc. FPGA*. 167–167.

[5] Scott Hauck. 1998. The roles of FPGAs in reprogrammable systems. *Proc. IEEE* 86, 4 (1998), 615–638.

[6] William NN Hung and Richard Sun. 2018. Challenges in Large FPGA-based Logic Emulation Systems. In *Proceedings of the 2018 International Symposium on Physical Design*. ACM, 26–33.

Table 3: Comparison of performance with the contest winners and the state-of-the-art works.

Design	1st place			[22]			[17]			Ours		
	TDM ratio	Runtime(s)	Score	TDM ratio	Runtime(s)	Score	TDM ratio	Runtime(s)	Score	TDM ratio	Runtime(s)	Score
synopsys1	40480	0.404	39279	37084	0.915	**36421**	38468	1.046	38283	37132	5.062	37384
synopsys2	32008942	0.554	30910258	31646600	2.01	31148738	31666114	2.4	31540538	31740560	0.955	**30900446**
synopsys3	128213846	6.497	123763681	127208000	36.124	125941288	127023600	40.756	126709389	127697676	7.631	**123561811**
synopsys4	7048746	18.775	6831271	6191610	172.843	6198872	6227184	114.916	6218344	6263338	67.5595	**6185801**
synopsys5	5189460	41.398	5043829	4540370	554.065	4582874	4588410	245.177	4585094	4570334	137.299	**4521130**
synopsys6	15751203396	74.756	15213447930	15749700000	341.232	15556991726	15748078752	86.134	15595919905	15749653186	47.059	**15106787263**
hidden1	409292666	1.183	**393785153**	408785000	5.058	401865333	408881842	1.918	405076396	408868062	1.295	393910217
hidden2	45942083770	25.392	44181670827	45936400000	357.823	45929505449	45942167188	27.752	45429961597	45953903518	18.8128	**43994213761**
hidden3	4867170072	78.247	4701321836	4862190000	528.645	4830522746	4865534056	92.67	4819200323	4865357542	47.118	**4663968467**
Final Score			64656814064			66886793447			66419249869			**64324086280**
Ratio	1.041	0.845		0.997	6.11221		1.002917	2.0334		1	1	

The best score in each benchmark is highlighted in **bold**. *Score* is the evaluation score computed according to Equation (5). *Final Score* is obtained by adding the evaluation scores of all test cases, which is the evaluation criteria used by the ICCAD 2019 contest organizers. *Ratio* in the last row shows the average of the normalized *TDM Ratio* and *Runtime* (normalized based on our results).

Table 4: Comparison of performance with the state-of-the-art works based on the new test cases.

New Designs	1st place			[22]			[17]			Ours		
	TDM Ratio	Runtime	Score	TDM ratio	Runtime(s)	Score	TDM ratio	Runtime(s)	Score	TDM ratio	Runtime(s)	Score
synopsys06_ex	110232848	78.621	109211128	98851500	2428.578	102827484	110210030	95.213	109492970	**98572664**	203.728	**99013070**
hidden01_ex	5108544	1.429	5023325	4872160	7.654	4908850	5099040	1.419	5013463	**4625304**	8.05	**4663501**
hidden02_ex	66847668	26.999	65995666	62213700	871.199	64538926	66830822	33.492	66186819	**60133426**	97.1411	**60477770**
hidden03_ex	72716244	83.417	72038228	64905100	3490.989	67796452	72688214	98.28	72182408	**64799604**	220.117	**65102506**
Final Score			252268347			240071712			252875660			**229256847**
Ratio	1.114	0.305		1.023	9.425		1.113	0.359		1	1	

Ratio in the last row shows the average of the normalized *TDM Ratio* and *Runtime* (normalized based on our results. The medium runtime is calculated based on these four methods for Equation (5).

[7] Masato Inagi, Yuichi Nakamura, Yasuhiro Takashima, and Shin'ichi Wakabayashi. 2015. Inter-FPGA Routing for Partially Time-Multiplexing Inter-FPGA Signals on Multi-FPGA Systems with Various Topologies. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 98, 12 (2015).

[8] Masato Inagi, Yasuhiro Takashima, and Yuichi Nakamura. 2009. Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on.* IEEE, 212–217.

[9] Masato Inagi, Yasuhiro Takashima, and Yuichi Nakamura. 2010. Globally optimal time-multiplexing of inter-FPGA connections for multi-FPGA prototyping systems. *IPSJ Transactions on System LSI Design Methodology* 3 (2010), 81–90.

[10] Masato Inagi, Yasuhiro Takashima, Yuichi Nakamura, and Atsushi Takahashi. 2008. Optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA prototyping systems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 91, 12 (2008), 3539–3547.

[11] Mohammed AS Khalid. 1999. *Routing architecture and layout synthesis for multi-FPGA systems.* Ph. D. dissertation, Dept. of ECE, Univ. Toronto.

[12] Wan-Sin Kuo, Shi-Han Zhang, Wai-Kei Mak, Richard Sun, and Yoon Kah Leow. 2018. Pin Assignment Optimization for Multi-2.5 D FPGA-based Systems. In *Proceedings of the 2018 International Symposium on Physical Design.* ACM, 106–113.

[13] Andrew Ling and Jason Anderson. 2017. The Role of FPGAs in Deep Learning. In *Proc. FPGA.* 3–3.

[14] 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 TCAD* 32, 5 (2013), 709–722.

[15] Larry McMurchie and Carl Ebeling. 2008. PathFinder: a negotiation-based performance-driven router for FPGAs. In *Reconfigurable Computing.* Elsevier.

[16] Kurt Mehlhorn. 1988. A faster approximation algorithm for the Steiner problem in graphs. *Inform. Process. Lett.* 27, 3 (1988), 125–128.

[17] Zou Peng, Lin Zhifeng, Shi Xiao, Wu Yingjie, Chen Jianli, Yu Jun, and Chang Yao-Wen. 2020. Time-Division Multiplexing Based System-Level FPGA Routing for Logic Verification. In *Proceedings of the 2020 Design Automation Conferencen.*

[18] Chak-Wa Pui, Gang Wu, Freddy Y. C. Mang, and Evangeline F. Y. Young. 2019. An Analytical Approach for Time-Division Multiplexing Optimization in Multi-FPGA based Systems. In *Proc. SLIP.*

[19] Chak-Wa Pui and Evangeline F. Y. Young. 2019. Lagrangian Relaxation-Based Time-Division Multiplexing Optimization for Multi-FPGA Systems. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD).*

[20] Chak-Wa Pui and Evangeline F. Y. Young. 2020. Lagrangian Relaxation-Based Time-Division Multiplexing Optimization for Multi-FPGA Systems. *ACM TODAES* (2020).

[21] Yu-Hsuan Su, Richard Sun, and Pei-Hsin Ho. 2019. 2019 CAD Contest: System-level FPGA Routing with Timing Division Multiplexing Technique. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD).*

[22] Lin Tung-Wei, Tai Wei-Chen, Lin Yu-Cheng, and Iris Hui-Ru Jiang. 2020. Routing Topology and Time-Division Multiplexing Co-Optimization for Multi-FPGA Systems. In *Proceedings of the 2020 Design Automation Conferencen.*