Dynamic Multi-FPGA Prototyping Platforms with Simultaneous Networking, Placement and Routing

Xinshi Zang
The Chinese University of Hong Kong
xszang@cse.cuhk.edu.hk

Qin Luo
The Chinese University of Hong Kong
qluo22@cse.cuhk.edu.hk

Zhongwei Shao
S2C Limited, China
zhongweis@s2ceda.com

Jifeng Zhang
S2C Limited, China
zyeez@s2ceda.com

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

Martin D.F. Wong
Hong Kong Baptist University
mdfwong@hkbu.edu.hk

ABSTRACT

Large-scale multi-FPGA prototyping platforms play an indispensable role in the functional verification of complex IC designs. The process of compiling circuit designs typically entails tasks such as partitioning, global placement and routing using a fixed multi-FPGA network. However, different circuit designs often exhibit varying inter-FPGA communication requirements after compilation. Neglecting this distinction, the use of fixed multi-FPGA networks may impede the performance enhancement of circuit verification. In this study, we investigate dynamic networking for multi-FPGA platforms and propose a comprehensive framework, which integrates simultaneous networking and system-level placement and routing. Based on theoretical analysis, we formulate this dynamic networking problem as an Integer Linear Programming (ILP) problem. Additionally, we introduce two innovative techniques, namely two-level ILP optimization and edge grouping, to expedite the ILP-solving process. Compared to the baselines on Titan23 and ICEEC22 benchmarks, our method achieves remarkable 11% and 47% improvements in system frequency respectively.

ACM Reference Format:
Xinshi Zang, Qin Luo, Zhongwei Shao, Jifeng Zhang, Evangeline F.Y. Young, and Martin D.F. Wong. 2024. Dynamic Multi-FPGA Prototyping Platforms with Simultaneous Networking, Placement and Routing . In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649476.3658713

1 INTRODUCTION

Multi-FPGA prototyping platforms (MFP) have been widely applied in the industry for the functional verification of circuits. Notably, according to [8], verification time consumes a substantial portion, ranging from 60% to 80%, of the entire IC development cycle. Consequently, enhancing the performance of MFP assumes paramount importance for expediting IC development endeavors.

A conventional flow for compiling an RTL design into multi-FPGA platforms encompasses a series of stages, including synthesis, partitioning, *system-level* placement and routing, and *intra-FPGA* placement and routing [3]. After logic synthesis, a large circuit netlist is partitioned into several sub-circuits each of which is then placed on a separate FPGA. Subsequently, *system-level* routing is performed to route cut nets through the MFP network. Inter-FPGA nets are then assigned with a time-division multiplexing (TDM) ratio on physical wires. Finally, each sub-netlist is placed and routed within one FPGA.

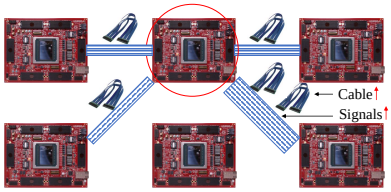


Figure 1: Customizable multi-FPGA platforms.

Based on the inter-FPGA connections, Tang et al. [16] classified multi-FPGA platforms into three categories, including hardwired off-the-shelf [15], cabling [2, 11], and custom [10] platforms. The hardwired off-the-shelf MFP is a ready-made multi-FPGA board, where all the FPGAs are connected by fixed printed wires. In contrast, the cabling and custom MFP will tailor the inter-FPGA cables or wires, which we call **networking** in this paper, for a specific design. For instance, Fig. 1 shows a cabling MFP where the number of external cables of the FPGA in the red circle is adjusted according to the inter-FPGA signals.

As not all pairs of FPGA have direct connections in large-scale MFP, the cut nets may need to be routed through intermediate FPGAs. Moreover, because the cut nets usually outnumber the available inter-FPGA physical wires, the Time-Division-Multiplexing (TDM) technique [3] is widely adopted to send inter-FPGA signals in a pipeline way. The system frequency of MFP is known to be related to the multiplexing ratio of the cut size (cut-net number) to the wire count between connected FPGAs and the routing hop (intermediate FPGA count) [1, 6, 17]. Because the inter-FPGA wire count is constrained in MFP, managing well the wire distribution and the cut distribution to minimize the multiplexing ratio and routing hop is crucial in MFP.

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. CUHK14210923).

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658713

In [16], Tang et al. proposed an automatic flow to optimize the wire distribution of MFP based on the pre-computed routing solution. Although it has achieved promising performance improvements, their flow assumes that all the FPGAs in MFP can be fully connected for the *system-level* routing, which may not hold in practice. In large-scale MFP, the FPGA connections can be constrained by the IO resource and distance [8]. Unaware of these constraints, there may not exist a feasible networking solution for the pre-computed routing solution.

Essentially, the problem of MFP networking and *system-level* routing is a chicken-and-egg problem. The routing task relies on the routing resources in the MFP architecture to distribute the cut nets, while the networking task requires the cut distribution to adjust the wire distribution. To optimize the system performance of MFP, it is essential to close the gap between these two tasks. To this end, we propose a simultaneous networking and *system-level* placement and routing framework for dynamic multi-FPGA prototyping platforms. The main contributions of this work are summarized as follows:

- We transform the original wire-constrained non-linear optimization problem into a linear wire-minimization problem. Given limited wire resources in MFP, the original networking, placement, and routing problem is a difficult non-linear optimization problem with complex constraints. However, we innovatively transform the hard wire constraints into an objective to minimize, which makes this problem linear and easy to solve. Furthermore, we theoretically prove the equivalence of the transformation.
- We propose a novel Integer Linear Programming (ILP) formulation for the wire-minimization problem to optimize networking and *system-level* placement and routing simultaneously.
- We propose two techniques, namely two-level ILP optimization and edge grouping, to prune the solution space of the ILP problem and make it efficiently solvable.
- We conduct extensive experiments on the Tian23 and ICEEC22 benchmarks, demonstrating that our method achieves 11% and 47% improvements, respectively, in system frequency compared to the baselines.

2 RELATED WORKS

Given a fixed MFP, to reduce the inter-FPGA communication delay, several methodologies have been proposed to improve the compilation flow [5, 9, 12, 13, 18]. Mak et al. [13] proposed the temporal logic replication to reduce the number of inter-FPGA signals. Liou et al. [12] proposed timing-driven partitioning by adjusting the weight of critical nets and iterative board-level placement by swapping FPGAs. Zheng et al. [18] proposed a TDM-aware system-level routing strategy to optimize both wirelength and TDM ratio. Chen et al. [5] proposed simultaneous partitioning and routing, which integrates the guided routing topologies in the multi-level partitioning framework. Inagi et al. [9] adopted ILP-based pin assignment to select signals to be time-multiplexed. Although considerable improvements have been achieved, these works can be impaired by inefficient networking [16].

The design of the networking of MFP is a time-consuming process and usually relies on the FPGA expertise of the engineers. To further improve the system frequency, recent works have delved into the problem of dynamic networking for MFP [1, 6, 16]. As discussed in Sec. 1, Tang et al. [16] proposed the first dynamic networking flow for MFP with simple networking constraints. Farooq et al. [6] experimentally studied different MFP architectures and concluded that the architectures with inter-FPGA wires closely matching the cut net requirements achieve better frequency results. The ICEEC 2022 contest [1] posed an industrial dynamic MFP networking problem with complex constraints, encouraging customization of MFP architectures for each circuit design. In this work, we focus on a dynamic MFP generation flow that supports practical networking constraints and tailors the wire distribution for better inter-FPGA communication delay.

3 PRELIMINARIES

The key terms utilized in this work are explained in Tab. 1.

Table 1: Terminology. The upper parts are abbreviated names, numbers or sets that can be pre-computed and the lower parts are variables in the ILP formulations.

Term	Description
$G, \hat{G}$	Partition graph and FPGA graph.
n_i, e_k	The ith G node and kth G edge.
$\hat{n}_j$, $\hat{e}_l$	The jth $\hat{G}$ node and lth $\hat{G}$ edge.
p_o	The oth routing path in $\hat{G}$.
α, β	The number of nodes and edges in G.
$\hat{\alpha}, \hat{\beta}$	The number of nodes and edges in $\hat{G}$.
γ	The number of routing paths in $\hat{G}$.
δ_k	The weight of e_k.
η_o	The routing hop count (number of intermediate $\hat{G}$ nodes) of p_o.
θ_{ol}	Whether p_o has $\hat{e}_l$ (1: yes, 0: no).
ω	The maximum number of wires for each FPGA.
μ	The maximum number of connected FPGAs for each FPGA.
ν	The maximum Manhattan distance for any connected FPGAs.
$\hat{E}_j$	The set of $\hat{G}$ edges connected to $\hat{n}_j$.
$P_{j_1 j_2}$	The set of routing paths between $\hat{n}_{j_1}$ and $\hat{n}_{j_2}$.
P_k	The set of candidate routing paths in $\hat{G}$ for e_k.
w_l	The wire number of $\hat{e}_l$.
v_l	Whether $\hat{e}_l$ exists ($w_l > 0$) (1: yes, 0: no).
a_{ij}	Whether n_i is assigned to $\hat{n}_j$ (1: yes, 0: no).
t_{ok}	Whether p_o is used by e_k (1: yes, 0: no).
c_l	The cut size, i.e., the total weight of G edges routed on $\hat{e}_l$.
r^*	The maximum ratio of cut to wire of the multi-FPGA system.
h^*	The maximum routing hop of the multi-FPGA system.

3.1 MFP Networks

Three typical MFP networks, namely 4-way, 8-way, and 1-hop [7], are illustrated in Fig 2. The 8-way and 1-hop networks provide twice the number of neighboring FPGAs compared to the 4-way network. In these networks, the IO pins of each FPGA are typically evenly distributed among the connected FPGAs. As a result, the 4-way network will have twice the number of wires between connected FPGAs compared to the 8-way and 1-hop networks.

3.2 MFP Frequency Evaluation

Compared to intra-FPGA communication, inter-FPGA communication incurs significantly higher delays, which directly impact the system clock frequency. As illustrated in Fig 3, the communication delay using TDM in a pipeline way mainly consists of the multiplexing delay (T_{mux}), routing hop delay (T_{hop}), and other constant

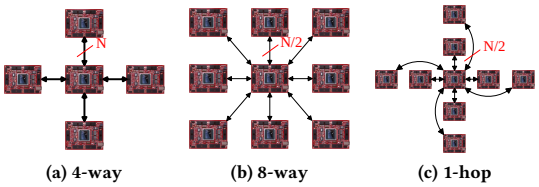


Figure 2: Typical networks of multi-FPGA platforms.

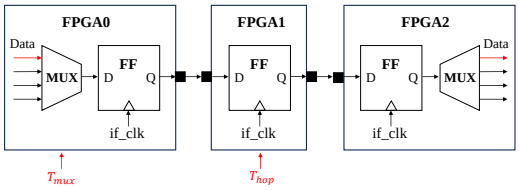


Figure 3: Timing analysis in MFP with TDM and one hop.

delays like flip flop delay. For simplicity, we estimate the system frequency using Eq. (1) in this work following [1, 6, 17], where r^* represents the maximum multiplexing ratio of signals to physical wires between each connected pair of FPGAs, and h^* denotes the maximum number of routing hops for all inter-FPGA signals. The inter-FPGA clock frequency, if_clk, is set to 100MHz in this study.

$$freq = \frac{if\_clk}{T_{mux} + T_{hop}} = \frac{if\_clk}{r^* + h^*} \tag{1}$$

3.3 Problem Formulation

In this work, we focus on the MFP networking and system-level placement and routing (NPR) problem. The circuit partitioning is assumed to be completed and regarded as input. The FPGA in MFP is assumed to be arranged in a 2D mesh and has a coordinate for distinction. We represent an MFP as an FPGA graph ($\hat{G}$) where the node and edge represent the FPGA and the direct connection. The task of networking is then to determine whether an edge exists between two nodes (v_l) and the wire count on the existing edge (w_l). Let $\hat{\alpha}$ and $\hat{E}_j$ be the node number and the edge set of the FPGA node in $\hat{G}$ respectively.

There are three common constraints for MFP networking. For each FPGA, the maximum number of wires (ω) and connected FPGAs (μ) need to be honored due to the FPGA IO constraints. Moreover, for each pair of connected FPGA, the maximum distance (ν) constraint is also specified because of the length limitation of wires. In summary, the objective of the NPR problem is to optimize the system frequency defined in Eq. (1) while satisfying the following constraints:

(1) Each circuit partition must be assigned to one FPGA and Each FPGA can only accommodate one circuit partition;
(2) Each cut net must be routed on inter-FPGA wires;
(3) The number of wires on each FPGA should not exceed ω, i.e., $\sum_{l \in \hat{E}_j} w_l \le \omega, \forall j \in [1, \hat{\alpha}]$;
(4) The number of FPGAs connected to each FPGA should not exceed μ, i.e., $\sum_{l \in \hat{E}_j} v_l \le \mu, \forall j \in [1, \hat{\alpha}]$;

To further illustrate the NPR problem, an example is provided in Fig. 4. There are four circuit partitions and four cut nets with different weights. Assume the networking constraint $\langle \omega, \mu, \nu \rangle$ is $\langle 8,$

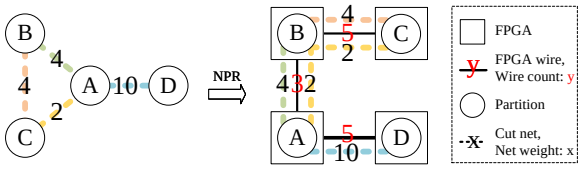


Figure 4: An example of the NPR problem.

$2, 1\rangle$. One optimal NPR solution is shown on the right, with r^* and h^* being 2 ($\frac{10}{5} = \frac{4+2}{3}$) and 1 for the yellow net, respectively.

4 METHODOLOGY

To optimize the frequency of multi-FPGA platforms, we propose a novel methodology that simultaneously addresses the networking and *system-level* placement and routing problem (NPR). The framework, outlined in Fig. 5, consists of three stages. In the initial stage, we construct the partition graph and the FPGA graph based on the input circuit partitioning solution. Next, we transform the original NPR problem to an Integer Linear Programming problem. To ensure the solvability of the ILP problem in a reasonable amount of time, we introduce a two-level ILP optimization framework along with an edge grouping technique for the partition graph. Finally, we construct a small-scale quadratic program (QP) to legalize the wire solution.

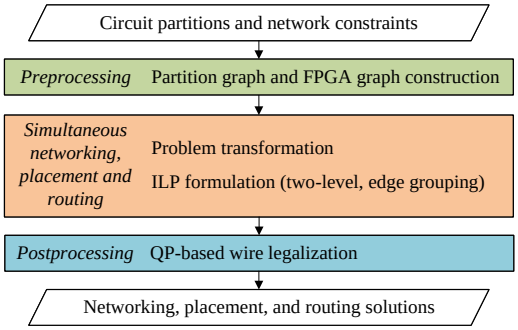
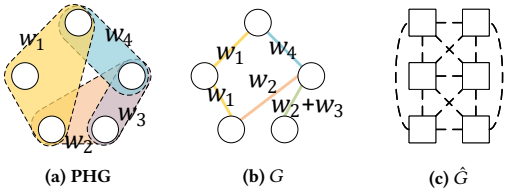


Figure 5: Main flow.

4.1 Partitioning Graph (G) and FPGA Graph ($\hat{G}$)

With the input circuit partitions, we can obtain a partition hypergraph where nodes represent partitions and hyperedges represent cut nets. To construct the general partitioning graph (G), we expand each hyperedge to $n-1$ source-to-sink edges, where n is the number of partitions connected by the hyperedge. We then merge edges between the same pair of partitions into a single edge with a weight equal to the total weight of all the merged edges. The total node number and edge number of the G are then denoted as α and β in the following. This transformation is illustrated in Fig. 6a and 6b, where a partition hypergraph (PHG) with four hyperedges is converted into a G with five edges. The purpose of this transformation is to ensure that the maximum routing hop count of the expanded edges is equal to that of the original hyperedge in the subsequent routing stage. However, this transformation may introduce inaccuracies in calculating the cut size. We will address this issue in the post-processing stage in Sec. 4.6.

Furthermore, we will construct a potential FPGA graph ($\hat{G}$), to define the solution space for the networking, placement, and routing. Given the G and $\hat{G}$, the NPR problem is then converted to determine (1) which $\hat{G}$ node to assign a G node to; (2) which $\hat{G}$ edges to route a G edge on; and (3) how many wires to allocate to a $\hat{G}$ edge. To provide efficient solution space, the $\hat{G}$ shape is determined as a rectangle with an aspect ratio within $[\frac{1}{2}, 1]$ and the $\hat{G}$ node number ($\hat{\alpha}$) also need be not less than the G node number (α). To meet the distance constraint, an edge can exist between any pair of $\hat{G}$ nodes if their distance does not exceed ν. As shown in Fig. 6c, considering the G in Fig. 6b and with a maximum distance constraint of $\nu = 2$, we determine that $\hat{\alpha} = 6$ and the $\hat{G}$ has a shape of 3×2. These dashed lines represent potential edges. We will decide which FPGAs in the $\hat{G}$ and which edges (connections) between the FPGAs will be used at the end. Once the $\hat{G}$ is determined, we will find all the shortest paths between every pair of $\hat{G}$ nodes. These shortest paths will be utilized as candidate routing paths in the subsequent routing stage. The total number of routing paths (γ) in $\hat{G}$ can be represented as $\frac{1}{2} * \sum_{j_1=1}^{\hat{\beta}} \sum_{j_2=1}^{\hat{\beta}} |P_{j_1 j_2}|$, where $P_{j_1 j_2}$ represents the path set between $\hat{G}$ nodes $\hat{n}_{j_1}$ and $\hat{n}_{j_2}$.



(a) PHG (b) G (c) $\hat{G}$

Figure 6: Examples of the construction of G and $\hat{G}$.

4.2 Problem Transformation

As summarized in Tab. 1, there are four main variables to be determined, including w_l, v_l, t_{ok}, a_{ij}, in the NPR problem. The wire count variable (w_l) and edge status variable (v_l) are used to determine the networking solution. The routing choice variable (t_{ok}) and the related mapping variable (a_{ij}) are used to describe the routing solution and the corresponding partition mapping solution. The NPR problem is then converted into determining the variables $\langle w_l, v_l, t_{ok}, a_{ij} \rangle$, with an optimization objective to minimize $r^* + h^*$ defined in Eq. (1). The r^* and h^* are further formulated in Eq. (2) and (3), where the descriptions of the terms can be found in Tab. 1. Specifically, the parameters, including G edge weight (δ_k), the path attribute (θ_{ol}), and the hop count (η_o) are all fixed values. The path set P_k for each e_k includes all paths in $\hat{G}$ and the cut size (c_l) on each $\hat{G}$ edge $\hat{e}_l$ depends on the routing variable (t_{ok}). In Eq. (2), the function $Flag(condition)$ will return 0 or 1 depending on the condition and the numerator calculates the total weight (cut size) of all G edges whose routing path passes through the $\hat{G}$ edge $\hat{e}_l$.

$$r^* = \max_{l=1}^{\hat{\beta}} \frac{c_l}{w_l} = \max_{l=1}^{\hat{\beta}} \frac{\sum_{k=1}^{\beta} \delta_k * Flag(e_k \text{ routes through } \hat{e}_l)}{w_l}$$
$$= \max_{l=1}^{\hat{\beta}} \frac{\sum_{k=1}^{\beta} \delta_k * (\sum_{o \in P_k} \theta_{ol} * t_{ok})}{w_l} \quad (2)$$

$$h^* = \max_{k=1}^{\beta} \sum_{o \in P_k} \eta_o * t_{ok} \quad (3)$$

In this section, we focus on discussing the maximum wire number ω which is a strict constraint for each $\hat{G}$ node $\hat{n}_j$. The full formulations of all the variable constraints in the original NPR problem will be discussed in Sec. 4.3. The objective r^* in Eq. (2) is non-linear w.r.t. the routing variable (t_{ok}) and wire variable (w_l) as shown in Eq. (2), resulting in optimizing r^* directly to be extremely challenging. Therefore, we transform the original NPR problem by switching the roles of the objective r^* and the constraint ω. In the transformation, **the term r^* is constrained to be one, i.e. $r^* = 1$, while the hard constraint ω is replaced by a new soft variable w^* to be minimized**, i.e. $\sum_{l \in \hat{E}_j} w_l \le w^*$. Once the value of w^* is determined, we can uniformly scale all the obtained wire counts in the new NPR problem by a factor of $\frac{\omega}{w^*}$. Since the r^* is fixed as one in the new NPR problem and with the same cut distribution, the value of r^* after wire scaling will equal $\frac{w^*}{\omega}$. Therefore, we then introduce a new objective, shown in Eq. (4), with $\frac{w^*}{\omega}$ as the r^* proxy, for the new NPR problem. This problem transformation allows us to remove the non-linearity and formulate the new NPR problem as an ILP problem.

$$\text{minimize} \quad \frac{w^*}{\omega} + h^* \quad (4)$$

We proved Theorem 4.1 to show the equivalence between the original and the transformed NPR problems. Suppose the wire number could be any real number, and let w_0^* and h_0^* be the minimum wire number and routing hop count obtained by solving the new NPR problem minimizing Eq. (4). The final r_0^* is then given by $\frac{w_0^*}{\omega}$.

Theorem 4.1. *The optimal value, $r_0^* + h_0^*$, achieved in the transformed NPR problem is also optimal for the original NPR problem.*

Proof. Suppose there exists a better NPR solution denoted as $S = \langle \overline{w}_l, \overline{v}_l, \overline{a}_{ij}, \overline{t}_{ok} \rangle$ for the original NPR problem, which has a smaller objective $\overline{r}^* + \overline{h}^*$, i.e. $\overline{r}^* + \overline{h}^* < r_0^* + h_0^*$. Since in the original problem, ω is the maximum wire number for each $\hat{G}$ node $\hat{n}_j$, we can have $\sum_{l \in \hat{E}_j} \overline{w}_l \le \omega, \forall j \in [1, \hat{\alpha}]$. Furthermore, according to the r^* definition in Eq. (2), we have $\overline{r}^* \ge \frac{\overline{c}_l}{\overline{w}_l}, \forall l \in [1, \hat{\beta}]$.

Based on S, we can construct a solution for the transformed problem denoted as $\widetilde{S} = \langle \widetilde{w}_l, \widetilde{w}^*, \overline{v}_l, \overline{a}_{ij}, \overline{t}_{ok} \rangle$, where $\widetilde{w}_l = \overline{w}_l * \overline{r}^*$ and $\widetilde{w}^* = \omega * \overline{r}^*$. Next, we will show that $\widetilde{S}$ is a feasible solution for the transformed NPR problem. As the differences between the original and the transformed problems are only the r^* and wire constraints, the $\langle \overline{v}_l, \overline{a}_{ij}, \overline{t}_{ok} \rangle$ in $\widetilde{S}$ are also feasible for the transformed problem. For the r^* constraint of $\widetilde{S}$, we have $\frac{\overline{c}_l}{\widetilde{w}_l} = \frac{\overline{c}_l}{\overline{w}_l * \overline{r}^*} = \frac{\overline{c}_l}{\overline{w}_l} * \frac{1}{\overline{r}^*} \le \frac{\overline{c}_l}{\overline{w}_l} * \frac{\overline{w}_l}{\overline{c}_l} = 1, \forall l \in [1, \hat{\beta}]$. For the wire constraint of $\widetilde{S}$, we have $\sum_{l \in \hat{E}_j} \widetilde{w}_l = \sum_{l \in \hat{E}_j} (\overline{w}_l * \overline{r}^*) = (\sum_{l \in \hat{E}_j} \overline{w}_l) * \overline{r}^* \le \omega * \overline{r}^* = \widetilde{w}^*, \forall j \in [1, \hat{\alpha}]$. Thus, $\widetilde{S}$ also satisfies the r^* and wire constraints of the transformed problem and therefore is a feasible solution.

Since the routing solution of $\widetilde{S}$ is the same as that of S, the maximum hop count h^* of $\widetilde{S}$ is also $\overline{h}^*$. According to Eq. (4), the objective of $\widetilde{S}$ is $\frac{\widetilde{w}^*}{\omega} + h^* = \overline{r}^* + \overline{h}^* < r_0^* + h_0^*$. However, this contradicts the fact that $r_0^* + h_0^*$ is the optimum of the transformed problem. Therefore, the assumption is incorrect and the proof is done. □

One example of the problem transformation is illustrated in Fig. 7. With the same netlist in Fig. 4, we can first obtain an optimal

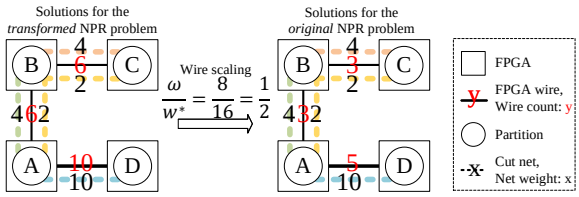


Figure 7: An example of the problem transformation. The networking constraint (ω, μ, ν) is (8, 2, 1).

solution for the transformed NPR problem with r^* and w^* being 1 and 16 respectively. After scaling all the wire counts by $\frac{\omega}{w^*} = \frac{1}{2}$, we can obtain a valid and optimal solution for the original NPR problem with the final r^* being 2 ($\frac{w^*}{\omega}$).

4.3 Basic ILP Formulation

Based on the above problem transformation, we will first formulate the four constraints defined in Sec. 3.3 with the variables (w_l, v_l, t_{ok}, a_{ij}). All necessary terms are described in Tab. 1 and for brevity, we may not explain all of them again in the following.

$$\sum_{l \in \hat{E}_j} w_l \le w^* \quad \forall j \in [1, \hat{\alpha}] \tag{5}$$

$$\sum_{l \in \hat{E}_j} w_l \le \sum_{i \in N_j} a_{ij} * M \quad \forall j \in [1, \hat{\alpha}] \tag{6}$$

$$\sum_{l \in \hat{E}_j} v_l \le \sum_{i \in N_j} a_{ij} * \mu \quad \forall j \in [1, \hat{\alpha}] \tag{7}$$

$$v_l \le w_l \le v_l * M \quad \forall l \in [1, \hat{\beta}] \tag{8}$$

$$\sum_{j \in \hat{N}_i} a_{ij} = 1 \quad \forall i \in [1, \alpha] \tag{9}$$

$$\sum_{i \in N_j} a_{ij} \le 1 \quad \forall j \in [1, \hat{\alpha}] \tag{10}$$

$$\sum_{o \in P_k} t_{ok} = 1 \quad \forall k \in [1, \beta] \tag{11}$$

$$\eta_o * t_{ok} \le \sum_{l \in \hat{E}'_o} v_l - 1 \quad \forall k \in [1, \beta], \forall o \in P_k \tag{12}$$

$$a_{i_1 j_1} + a_{i_2 j_2} \le \sum_{o \in P_{j_1 j_2}} t_{ok} + 1 \quad \forall k \in [1, \beta], \forall j_1 \ne j_2 \in [1, \hat{\alpha}], \; i_1, i_2 \in endpoint(e_k) \tag{13}$$

$$\sum_{o \in P_k} \eta_o * t_{ok} \le h^* \quad \forall k \in [1, \beta] \tag{14}$$

$$\sum_{k=1}^{\beta} \sum_{o \in P_k} \delta_k * \theta_{ol} * t_{ok} \le w_l \quad \forall l \in [1, \hat{\beta}] \tag{15}$$

$$w_l, w^* \in \mathbb{N}, v_l \in \{0, 1\} \quad \forall l \in [1, \hat{\beta}] \tag{16}$$

$$t_{ok} \in \{0, 1\} \quad \forall k \in [1, \beta], \forall o \in P_k \tag{17}$$

$$a_{ij} \in \{0, 1\} \quad \forall i \in [1, \alpha], \forall j \in [1, \hat{\alpha}] \tag{18}$$

There are three types of constraints related to networking, placement and routing. Firstly, the networking variables (w_l, v_l) must satisfy the constraints (5) - (8). As the core of the problem transformation, Eq. (5) requires that the wire number (w_l) of each $\hat{G}$ node can not exceed the variable w^*. In Eq. (6), M is a large constant. If $\hat{G}$ node $\hat{n}_j$ is used (i.e., there is one G node assigned to $\hat{n}_j$), Eq. (6) will have no constraint on the wire counts on $\hat{n}_j$. Otherwise, it will make the wire counts zero to avoid redundant wire allocation. Similarly, Eq. (7) ensure the actual edge number of $\hat{n}_j$ cannot exceed the maximum number (μ) if the $\hat{n}_j$ is used. Since the variable w_l and v_l are affected by each other, Eq. (8) ensures that if v_l is 0, w_l is also 0, and $w_l \ge 1$ otherwise.

Secondly, the placement variable (a_{ij}) and the routing variable (t_{ok}) must satisfy the constraints (9) - (12). Eq. (9) and (10) ensure that each G node is assigned to a $\hat{G}$ node and each $\hat{G}$ node can accommodate one G node at most. The candidate $\hat{G}$ node set $\hat{N}_i$ in Eq. (9) includes all $\hat{G}$ nodes for each G node in the basic ILP formulation. In Eq. (11), each G edge must select exactly one routing path. The routing path set P_k in the basic ILP formulation includes all the routing paths in $\hat{G}$ for each G edge. Eq. (12) and (13) align the routing solution with the networking solution and the placement solution respectively. In Eq. (12), $\hat{E}'_o$ denotes the $\hat{G}$ edge set containing in the routing path p_o. If p_o is selected by any G edge (i.e., $t_{ok} = 1$), all the $\hat{G}$ edges in $\hat{E}'_o$ must be used (i.e. $v_l = 1$). In Eq. (13), if the endpoints of G edge e_k, i.e. n_{i_1} and n_{i_2}, are assigned to $\hat{G}$ nodes $\hat{n}_{j_1}$ and $\hat{n}_{j_2}$, e_k must choose one routing path from the path set ($P_{j_1 j_2}$) between $\hat{n}_{j_1}$ and $\hat{n}_{j_2}$ (i.e. $\sum_{o \in P_{j_1 j_2}} t_{ok} = 1$).

In addition, there are also two constraints related to the r^* and h^* in Eq. (14) and (15). Eq. (14) ensures that h^* is the maximum hop count for all routing paths. As the other core of the transformation, the r^* in the transformed NPR problem is fixed as one, which is realized by Eq. (15).

In the basic ILP formulation, the candidate $\hat{G}$ node set includes all $\hat{G}$ nodes for each G node. The candidate routing paths for each G edge then include all routing paths in $\hat{G}$. As a result, the total number of the routing variables is the product of the number of G edges (β) and the number of routing paths (γ), which significantly dominates the size of the other variables. For large-scale circuit designs, this can lead to a large number of routing variables, making it challenging to solve the ILP problem efficiently within a limited time. To address this issue and to prune the solution space, we propose a two-level ILP optimization and G edge grouping.

4.4 Two-level ILP Optimization

In the two-level ILP optimization, we first perform a coarse-level cut-driven placement and then a fine-level placement with simultaneous networking and routing. During the coarse level, we divide $\hat{G}$ into four quadrants and construct a coarsened FPGA graph (CFG). Each CFG node represents one quadrant of $\hat{G}$ and is connected with its two horizontal and vertical neighbors. Next, we apply a relatively smaller ILP model to assign G nodes to CFG nodes and minimize the maximum cut size on CFG edges.

After solving the coarse-level ILP problem, we obtain the candidate region, i.e. the $\hat{G}$ node set $\hat{N}_i$ for each G node n_i. The new size of $\hat{N}_i$ is approximately $\frac{1}{4}$ of the original size ($\hat{\alpha}$). As explained at the end of Sec. 4.1, the candidate routing paths for each G edge will then be approximately $\frac{1}{16}$ of all the routing paths in $\hat{G}$. As a result, the total number of routing variables can be reduced to $\frac{1}{16} * \beta * \gamma$. We then apply a fine-level ILP optimization with Eq. (4) - (18) to perform simultaneous networking, placement, and routing. The size of the fine-level ILP is significantly smaller than that of the basic ILP formulation.

4.5 Partition Edge Grouping

Although the coarse-level placement can reduce the number of routing variables for each G edge, the total number of routing variables can still be huge for ILP solving when the number of G edges (β) is large. Therefore, we propose an edge grouping technique to further reduce the total number of routing variables after the coarse-level placement. We divide the G edges into λ groups based on their weights. We use one-dimensional k-means clustering to put G edges with similar weights into the same edge group. The weight of an edge group g_q is set as the maximum weight of all the G edges in g_q. The routing variables, t'_{oq}, will be used to represent the choices of routing paths for g_q. We then replace the routing variable t_{ok} with t'_{oq} and make corresponding adaptations in Eq. (11) - (15). Let E_q denote the set of G edges in g_q and we need to select $|E_q|$ different routing paths for the G edges in g_q in alignment with the placement solution. In this work, we put λ as 10 to get a good balance between the number of edge groups and the weight similarity within an edge group.

4.6 Wire Legalization with Quadratic Programming

Since the wire constraints are relaxed in our basic ILP formulations, the final wire solution needs to be legalized into an integral solution satisfying the maximum wire constraint (ω). Besides, the cut size calculated on each $\hat{G}$ edge may not be accurate due to the hyperedge expansion and edge grouping operations. To address these issues, we propose a post-processing stage to compute the actual cut size and to legalize the wires using Quadratic Programming (QP). Based on the NPR solution, we first re-calculate the cut size c_l on each $\hat{G}$ edge $\hat{e}_l$ and minimize the final r^* in Eq. (19) while satisfying the constraints (20) - (22). Here, $\hat{E}$ denotes the set of $\hat{G}$ edges with non-zero wires in the original networking solution. It is important to note that the QP-based legalization only adjusts the wire numbers and will not change the network topology nor the placement and routing solution. Since the size of the QP is small, the optimal solution to the QP problem can be obtained quickly.

$$\text{minimize } r^* \tag{19}$$

$$\text{subject to } \sum_{l \in \hat{E}_j, l \in \hat{E}} w_l \le \omega \quad \forall j \in [1, \hat{\alpha}] \tag{20}$$

$$c_l \le r^* * w_l \quad \forall l \in \hat{E} \tag{21}$$

$$w_l \in [1, \omega], w_l \in \mathbb{Z} \quad \forall l \in \hat{E} \tag{22}$$

5 EXPERIMENTAL EVALUATION

In this study, we utilized C++ to implement all the methods, which were subsequently evaluated on a Linux server featuring an Intel Xeon 2.9GHz CPU and 256GB of memory. For solving ILP instances, we employed CPLEX version 22.1.0 as the ILP solver. The default configuration of the ILP solver was used, employing a thread count of 20 and a time limit of 1000 seconds. For circuit partitioning, we applied PaToH [4] to perform min-cut partitioning while satisfying FPGA resource constraints. The partitioning results will be used to test different methods of networking and routing for multi-FPGA platforms.

5.1 Benchmarks

We did experiments on the Titan23 [14] and ICEEC 2022 contest [1] benchmarks[1]. The circuit statistics for both the Titan23 and ICEEC22 benchmarks are presented in Tab. 2 and 3. There are a total of 10 circuits in the ICEEC22 benchmark, but the sizes of the case1 - case6 circuits are very small and we excluded them from our experiments. For the ICEEC22 benchmark, the networking constraints (ω, μ, ν) are specified individually for each design. For the Titan23 benchmark, the constraints are specified as (800, 32, 4) for all cases.

5.2 Overall Performance

We compared our method with two kinds of baselines on Titan23 circuit benchmarks, including the state-of-the-art general dynamic networking method [16] and the methods using fixed MFP networks. For the second baselines, the three networks described in Fig. 2 are used, and an ILP formulation based on Eq. (9) to (13) is employed for system-level routing. As presented in Tab. 2, among all fixed FPGA networks, the 1-hop network demonstrates the highest frequency. Compared with the 1-hop network and the method [16], our method still achieves approximately 49% and 11% improvements. It is worth noting that the method [16] requires any two FPGAs to be connected when there are signal nets connecting them, which hardly holds in practice. In contrast, our method can not only dynamically optimize wire distribution to align with the cut distribution but also adapt interconnections to accommodate diverse routing requirements.

Table 2: Frequency results of Titan23 benchmarks. The frequency is in MHz, which holds for all the following tables.

Cases	#Nodes	#Nets	4-way	8-way	1-hop	[16]	Ours
cholesky_mc	108,441	140,139	9.22	8.22	12.67	23.33	**24.36**
des90	110,517	138,853	12.07	11.59	12.49	23.81	**25.25**
bitonic_mesh	190,714	233,978	7.81	6.37	8.57	**12.50**	12.40
dart	202,354	223,301	28.86	35.37	28.86	44.02	**44.03**
openCV	212,449	279,104	4.68	3.60	4.49	5.41	**7.32**
minres	257,377	316,569	10.10	9.66	12.20	31.25	**35.63**
cholesky_bdti	255,410	331,676	4.48	5.19	6.31	12.50	**17.99**
gsm_switch	490,030	504,592	6.84	7.10	8.43	12.90	**15.41**
LU230	567,806	662,911	5.28	5.15	6.23	**12.12**	12.10
bitcoin_miner	1,087,505	1,446,377	3.82	2.84	4.26	9.09	**10.54**
Avg. Ratio	-	-	0.46	0.43	0.51	0.89	**1.00**

Table 3: Frequency results of ICEEC22 benchmarks.

Cases	#Nodes	#Nets	ω	μ	ν	1st	2nd	Ours
case7	54,970	76,258	800	8	2	100.00	50.17	**100.00**
case8	62,846	86,139	800	8	3	33.30	50.85	**58.42**
case9	784,814	871,588	1600	16	4	11.00	20.24	**40.00**
case10	3,066,539	3,324,963	3200	32	4	13.20	3.10	**45.00**
Avg. Ratio	-	-	-	-	-	0.53	0.49	**1.00**

We further conducted a comparative analysis against the top two teams[2] in the ICEEC 2022 contest, which represents state-of-the-art methods for ICEEC22 benchmarks. As depicted in Tab. 3, our method exhibits the most remarkable performance in all cases

[1]All benchmarks are stored in https://drive.google.com/drive/folders/1-uzy-BMXmmN4EqrSgG5VKIN9Z58bZeOs?usp=sharing.
[2]The results of the top two teams were obtained from the contest organizer.

and outperforms the top team by an impressive 47% in terms of the average per-case ratio of the system frequency.

5.3 Impact of Networking Constraints

To investigate the impact of networking constraints, we conducted a verification of our method on the Titan23 benchmark with three additional networking constraints in terms of μ and ν. Larger values of μ and ν indicate that each FPGA can be connected to a greater number of FPGAs and to FPGAs at a larger distance apart. As shown in Tab. 4, our method with larger μ and ν values exhibits superior performance, attributed to the expanded solution space for networking. Moreover, comparing the results in Tab. 2 and 4, our method with `e4-d1` and `e8-d2` constraints still outperforms the 4-way, 8-way, and 1-hop networks that have the same interconnection scopes. This justifies the effectiveness of our co-optimization of networking and routing.

Table 4: Frequency results of our method with different networking constraints. (m, n) refers to $\mu = m$ and $\nu = n$.

Cases	(4, 1)	(8, 2)	(12, 3)	(32, 4)
cholesky_mc	13.43	19.01	19.62	**24.36**
des90	17.15	20.17	**25.25**	**25.25**
bitonic_mesh	9.22	11.22	**12.44**	12.40
dart	29.42	**44.03**	**44.03**	**44.03**
openCV	5.09	6.41	**7.34**	7.32
minres	14.01	20.73	24.27	**35.63**
cholesky_bdti	7.33	11.91	15.32	**17.99**
gsm_switch	8.66	11.85	13.41	**15.41**
LU230	6.61	10.71	11.50	**12.10**
bitcoin_miner	5.63	8.15	**10.63**	10.54
Avg. Ratio	0.58	0.80	0.92	**1.00**

5.4 Ablation Study

In this section, we explore the effects of the two-level ILP optimization and the edge grouping techniques. We propose two variants: Ours-v1, which does not employ neither the TL nor the EG techniques, and Ours-v2, which excludes the TL technique only. As shown in Tab. 5, Ours-v1 not only fails to find feasible solutions in five cases but also generates a sub-optimal solution for the minres design. Although Ours-v2 can find feasible solutions for most of the cases, the frequency is slightly worsened in certain cases like LU230 and bitcoin_miner.

Table 5: Ablation study. For the ILP solving status, 'O', 'F', and '-' mean optimal, feasible, and infeasible respectively.

Cases	Ours-v1		Ours-v2		Ours	
	freq	*status*	*freq*	*status*	*freq*	*status*
cholesky_mc	**24.36**	O	**24.36**	O	**24.36**	O
des90	**25.25**	O	**25.25**	O	**25.25**	O
bitonic_mesh	-	-	12.39	F	**12.40**	F
dart	**44.03**	O	**44.03**	O	**44.03**	O
openCV	-	-	7.26	F	**7.32**	F
minres	26.27	F	35.48	F	**35.63**	F
cholesky_bdti	-	-	-	-	**17.99**	F
gsm_switch	**15.41**	F	**15.41**	F	**15.41**	O
LU230	-	-	11.26	F	**12.10**	F
bitcoin_miner	-	-	10.00	F	**10.54**	F
Avg. Ratio	0.96	-	0.99	-	**1.00**	-

6 CONCLUSION

In this work, we present a dynamic networking framework for multi-FPGA platforms. The transformation of the wire-constrained networking, placement, and routing problem and the formulation as an ILP problem, along with the proposed speed-up techniques, ensure an efficient solving process. Experimental results demonstrate the effectiveness of our method, surpassing both the other dynamic networking methods and the approaches employing pre-designed and fixed networks. Our work offers a valuable contribution to dynamic multi-FPGA platforms, enabling higher system frequencies and efficient customization.

REFERENCES

[1] 2022. Dynamic Networking Iterative Partitioning Algorithm Design. https://eda.icisc.cn/en/file/cacheFile/d5997d47f1ba4e05a27bc782dc2fae05.pdf. [Online; accessed 17-Sep-2023].

[2] Sameh Asaad, Ralph Bellofatto, Bernard Brezzo, Chuck Haymes, Mohit Kapur, Benjamin Parker, Thomas Roewer, Proshanta Saha, Todd Takken, and José Tierno. 2012. A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multi-core processor simulation. In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. 153–162.

[3] 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* (1997).

[4] Ümit V Çatalyürek and Cevdet Aykanat. 2011. Patoh (partitioning tool for hypergraphs). In *Encyclopedia of Parallel Computing*.

[5] Ming-Hung Chen, Yao-Wen Chang, and Jun-Jie Wang. 2021. Performance-driven simultaneous partitioning and routing for multi-FPGA systems. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1129–1134.

[6] Umer Farooq, Roselyne Chotin-Avot, Moazam Azeem, Maminionja Ravoson, and Habib Mehrez. 2018. Novel architectural space exploration environment for multi-FPGA based prototyping systems. *Microprocessors and Microsystems* (2018).

[7] Scott Hauck, Gaetano Borriello, and Carl Ebeling. 1998. Mesh routing topologies for multi-FPGA systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (1998).

[8] William NN Hung and Richard Sun. 2018. Challenges in large FPGA-based logic emulation systems. In *International Symposium on Physical Design*.

[9] 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), 2572–2583.

[10] Helena Krupnova. 2004. Mapping multi-million gate SoCs on FPGAs: industrial methodology and experience. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Vol. 2. IEEE, 1236–1241.

[11] Ari Kulmala, Erno Salminen, and Timo D Hämäläinen. 2007. Evaluating large system-on-chip on multi-FPGA platform. In *Embedded Computer Systems: Architectures, Modeling, and Simulation: 7th International Workshop, SAMOS 2007, Samos, Greece, July 16-19, 2007. Proceedings 7*. Springer, 179–189.

[12] Sin-Hong Liou, Sean Liu, Richard Sun, and Hung-Ming Chen. 2020. Timing driven partition for multi-fpga systems with tdm awareness. In *Proceedings of the 2020 International Symposium on Physical Design*. 111–118.

[13] Wai-Kei Mak and Evangeline FY Young. 2002. Temporal logic replication for dynamically reconfigurable FPGA partitioning. In *Proceedings of the 2002 international symposium on Physical design*. 190–195.

[14] Kevin E Murray, Scott Whitty, Suya Liu, Jason Luu, and Vaughn Betz. 2013. Titan: Enabling large and complex benchmarks in academic CAD. In *International Conference on Field Programmable Logic and Applications*.

[15] Qingshan Tang, Habib Mehrez, and Matthieu Tuna. 2012. Design for prototyping of a parameterizable cluster-based Multi-Core System-on-Chip on a multi-FPGA board. In *2012 23rd IEEE International Symposium on Rapid System Prototyping*.

[16] Qingshan Tang, Matthieu Tuna, and Habib Mehrez. 2014. Performance comparison between multi-fpga prototyping platforms: Hardwired off-the-shelf, cabling, and custom. In *International Symposium on Field-Programmable Custom Computing Machines*.

[17] Mariem Turki, Habib Mehrez, Zied Marrakchi, and Mohamed Abid. 2013. Partitioning constraints and signal routing approach for multi-fpga prototyping platform. In *International Symposium on System on Chip*.

[18] Dan Zheng, Xiaopeng Zhang, Chak-Wa Pui, and Evangeline FY Young. 2021. Multi-FPGA Co-optimization: Hybrid Routing and Competitive-based Time Division Multiplexing Assignment. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 176–182.