2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) | 979-8-3503-7243-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/FCCM60383.2024.00024

A Routability-Driven Ultrascale FPGA Macro Placer with Complex Design Constraints

Qin Luo[1], Xinshi Zang[1], Qijing Wang[1], Fangzhou Wang[1], Evangeline F.Y. Young[1], Martin D.F. Wong[2]
[1]The Chinese University of Hong Kong, [2]Hong Kong Baptist University
{qluo22, xszang, qjwang21, fzwang fyyoung}@cse.cuhk.edu.hk, mdfwong@hkbu.edu.hk

***Abstract*—Macro placement significantly influences the performance of the FPGA placement. However, constraints in modern designs like relative placement constraint (RPC) and regional constraint (RC) are often overlooked in existing routability-driven FPGA placers during macro placement. These constraints introduce challenges in optimizing routability during global placement and macro legalization stages. In this paper, we propose a novel macro placer that specifically addresses these constraints while optimizing routability. Our macro placer integrates macro size-aware pseudo nets, RC guided spreading, and multi-stage look-ahead legalization techniques to enhance routability with specified design constraints. Experimental results show that compared with DreamplaceFPGA-MP and the macro placer in Vivado, our proposed approach achieves 6% and 8% total routing score reduction on the MLCAD2023 contest benchmark. Moreover, the place and route time is reduced by 3.5% on average and up to 43% after our macro placer is integrated into Vivado. These compelling results demonstrate the efficiency gains and superior routability optimization achieved through our approach.**

I. Introduction

FPGAs play a pivotal role in circuit prototyping [1] and heterogeneous computing [2] for the reconfiguration capabilities. Within the FPGA implementation process encompassing logic synthesis, technology mapping, placement and routing, FPGA placement holds utmost importance. It maps the cells in the logical netlist onto the locations on FPGA boards and significantly determines the success of the routing. Specifically, FPGA placement is divided into global placement and legalization, where the global placement determines rough locations with the minimization of the wirelength and the cell density, while the legalization shifts each cell to the legal location with slight displacement. In contrast to ASIC placement, FPGA placement is challenging due to its heterogeneous nature and the limited resources for place and route [3].

Look-up tables (LUT), flip-flops (FF), digital signal processors (DSP), block memories (BRAM), and input/output buffers (IO) are five essential resources to be placed within modern FPGAs [4]. Several LUTs and FFs are packed into one Configurable Logic Block (CLB). The sites containing these five resources differ in physical size on FPGA boards with different importance to the placement. For example, on the Xilinx Ultrascale+ xcvu30 board [5], the BRAM block is five times larger than the CLB, and the DSP block is twice the size of the CLB. The locations of the BRAMs and DSPs have the most significant influence on the overall performance of the placement, distinguished by their significantly larger sizes and intricate interconnections with the other cells. Therefore, in the usual categorization, BRAMs and DSPs are regarded as macros, and LUTs and FFs are treated as standard cells in the placement [6], [7]. Macro placement serves as an individual stage in the existing quadratic [8], [9] or nonlinear mixed-size placers [10], [11], which firstly determine the positions of the macros and then place the standard cells with macros fixed.

Modern circuit designs incorporate distinctive and challenging constraints that significantly influence the FPGA macro placement. One is relative placement constraint (RPC), stemming from the hierarchical design methodology [12]–[14]. RPC requires that the cells

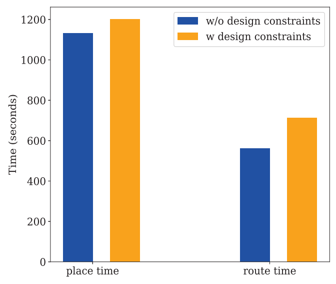


Fig. 1: Average placement and routing time using Vivado for the designs with and without constraints on the MLCAD2023 benchmark

with the same constraint must maintain a specific relative position to each other. Another is regional constraint (RC) [15]–[17], where the back-end engineers group related logic and assign it to a region to achieve timing closure easily. While both RPC and RC are supported in the commercial FPGA tool Vivado (like $rloc$ and $pblock$) [18], most academic FPGA placers do not take these two kinds of constraints simultaneously into the routability optimization. Among these placers, a few simply consider the RPC by amalgamating the macros in the same RPC into a large cascaded macro that participates in the overall placement [19]–[22], without no further techniques to improve routability. Actually, the occurrence of the RPC and RC increases difficulties in finding macro placement solutions with less congestion. Fig. 1 provides a comparative analysis of place and route times using Vivado for the designs without and with the constraints on the MLCAD2023 benchmark. A 20% increase in place and route time is observed on average for designs with the constraints, caused by the increased difficulties in routing. The scarce candidate sites to place the large cascaded macros and the increased cell density in constrained regions collectively hinder the macro placer's ability to find an optimal routable solution. In response to the need for improved macro placement solutions under design constraints, an academic contest organized by MLCAD2023 committee and AMD Corporation [6] focused on FPGA macro placement with routability optimization and realistic design constraints.

In this paper, we propose a novel macro placer based on the SimPL framework [23] that incorporates advanced techniques for better routability optimization under the RPC and RC. It is worth noting that these two constraints are not considered in the other refined SimPL-based FPGA placers, such as RippleFPGA [9] and Liquid [24]. In the global placement, we modify the pseudo nets in the original framework to be macro size-aware and the cell spreading to be guided by RC, to reduce the congestion level under RPC and RC. In the legalization, we categorize the macros into different groups and legalize them sequentially with look-ahead techniques, to ensure a valid macro placement solution under the two complex constraints. The contributions of the papers are summarized as follows:

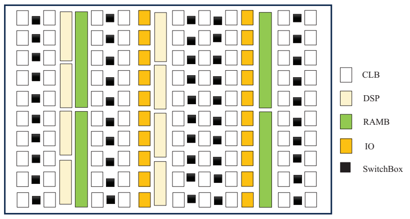


Fig. 2: The layout of the Xilinx Ultrascale series board

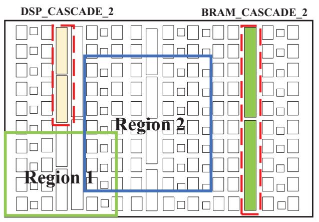


Fig. 3: Relative placement constraints and regional constraints

- We propose the macro size-aware pseudo nets and RC guided spreading to enhance the routability in the global placement when both the RPC and RC are considered.
- A multi-stage look-ahead legalization algorithm is proposed to extend the progressive macro legalization to the scenarios with intensive RPC and RC.
- We conduct comparative evaluations of our macro placer against other solutions in both academic and commercial tools. Additionally, ablation studies are performed to demonstrate the effectiveness of the proposed techniques.

II. Preliminary

In this section, we firstly introduce the architectures of the Ultrascale series FPGAs that our macro placer targets and RPC and RC in modern designs. Then we mathematically formulate the FPGA macro placement problem.

A. FPGA Architecture

Our macro placer targets at the Xilinx Ultrascale series FPGAs [5] with the column-wise structure shown in Fig. 2. This FPGA board offers five common site types for flexible configuration: CLB, DSP, BRAM, IO, and switchbox. Each CLB can accommodate several LUTs and FFs. The sites in the same columns have the same types. The switchboxes are used to interconnect different sites with the prefabricated wires.

B. Design Constraints

Two categories of design constraints play a significant role in modern FPGA-based design. The first constraint is RPC, which exists in the advanced architectures like carry chains [22], systolic arrays [25] and memory cascades [26] demanding specific relative placements. In Fig. 3, the blocks in the dotted red rectangles indicate that the cells in the cascaded DSPs and the cascaded BRAMs should be placed adjacently, according to RPC. RC arises when back-end engineers confine certain cells within predefined regions. Fig. 3 shows two constrained regions with overlapping. Nodes lacking RC enjoy the flexibility to be placed at any location on the FPGA.

C. Problem Formulation

The FPGA macro placement problem can be formulated as follows. Let a hypergraph $H = (V, E)$ represent the design where macros are denoted as $V_M = \{v_M^1, v_M^2, v_M^3, ..., v_M^n\}$ and standard cells as $V_S = \{v_S^1, v_S^2, v_S^3, ..., v_S^k\}$. Both the macros and standard cells are movable in the design. The macros with the same RPC are clustered into one cascaded macro, where the set of macros V_M can be converted to a smaller set $V_{CM} = \{v_{CM}^1, v_{CM}^2, v_{CM}^3, ..., v_{CM}^m\}$ ($n \geq m$). Several rectangular regions with the boundary denoted as $R = \{(x_r^l, x_r^u, y_r^l, y_r^u)\}_{r=1}^{|R|}$ are pre-defined to constrain some of the cells in them. The FPGA macro placer aims to determine the locations of the macros V_{CM} with the minimization of the routing congestion. Additionally, the macro placement must ensure all resources are overflow-free across all FPGA sites and adhere to RPC and RC.

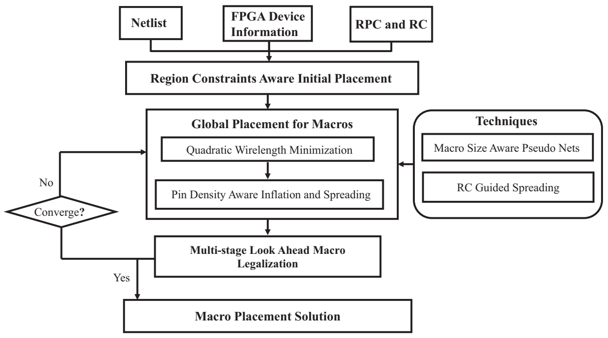

Netlist
FPGA Device Information
RPC and RC
Region Constraints Aware Initial Placement
Global Placement for Macros
Quadratic Wirelength Minimization
Pin Density Aware Inflation and Spreading
Techniques
Macro Size Aware Pseudo Nets
RC Guided Spreading
No
Converge?
Yes
Multi-stage Look Ahead Macro Legalization
Macro Placement Solution

Fig. 4: Overall flow of our macro placer

III. Methodology

Fig. 4 illustrates the overall flow of our macro placement framework. The framework is built upon the SimPL-based framework, with specific modifications tailored to address design constraints and enhance overall routability. Given the netlist, FPGA device information and the design constraints, the macros are initially placed according to the RC. The macro placer refines the macro locations by iterative global placement and macro legalization. The global placement involves quadratic wirelength minimization for all the connected cells, and the cell inflation and spreading to reduce the pin density. Since there are massive overlaps for macros after the global placement, multiple legalization stages are implemented for various macro categories. Note that the macro size-aware pseudo nets and the RC guided spreading are two important techniques introduced in our macro placer to achieve better routability under RPC and RC. In addition, the look-ahead technique is proposed to guarantee a valid macro placement solution during the progressive legalization under intensive constraints.

A. Regional Constraints Aware Initial Placement

In our macro placer, the initial placement of macros considers the RC, thereby establishing a more favorable starting point for global placement and easing macro legalization. For macros with RC whose boundaries are denoted as $(x_r^l, x_r^u, y_r^l, y_r^u)$, their locations are randomly sampled from a uniform distribution within the specified constrained regions:

$$\begin{aligned} x &\sim U(x_r^l + \sigma, x_r^u - \sigma) \\ y &\sim U(y_r^l + \sigma, y_r^u - \sigma) \end{aligned} \tag{1}$$

In Eq. (1), σ is the distance to the boundary of the region, and it takes 0.1 in our macro placer. For macros without any RC, the locations are initialized to any location within the FPGA board.

B. Global Placement for Macros

Although the primary focus is on placing macros, getting the distribution of standard cell locations is still crucial for achieving

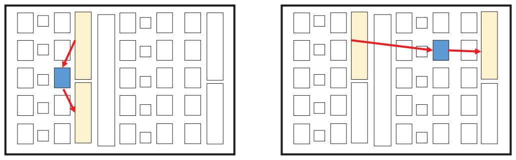

(a) With LUT-FF Cluster Guidance (b) Without LUT-FF Cluster Guidance

Fig. 5: BRAM placement with and without the guidance from LUT-FF Cluster

optimal macro placement. On one hand, plenty of indirect connections exist between macros through a few standard cell cluster hops [27]. In Fig. 5, two BRAMs are connected via a single LUT-FF cluster. The BRAMs are placed adjacently with a minimization of the distance between them in Fig. 5(a). Without considering these indirect connections, the BRAMs might be placed far apart, like what is shown in Fig. 5(b). On the other hand, the standard cell clusters occupy a substantial portion of the FPGA board. Therefore, both the macros and the standard cells participate in the global placement in our macro placer.

During the wirelength minimization stage in the previous SimPL-based framework, pseudo nets connecting the cells and their location in the last iteration are introduced to speed up the convergence of the placement. In this process, the total quadratic wirelength of all the nets and the weighted sum of the length for all the pseudo nets are minimized.

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{e\in E} W_e + \sum_{e_p\in E_P} [w_{e_p} W_{e_p}] \tag{2}$$

$$W_e = \sum_{i,j\in e} [w_{x,ij}^{B2B}(x_i - x_j)^2 + w_{y,ij}^{B2B}(y_i - y_j)^2] \tag{3}$$

where $E_p = \{e_p\}$ is the set of pseudo nets and W_{e_p} is the quadratic approximation of the pseudo net e_p. (x_i, y_i) and (x_j, y_j) are the coordinates of the i-th and j-th pin of the net e. $w_{x,ij}^{B2B}$ and $w_{y,ij}^{B2B}$ are weights determined by the Bound2Bound net module [28]. In the refined SimPL-based FPGA placer like RippleFPGA [9] and the Liquid [24], the weight of the pseudo net increases as the placement approaches convergence. It is calculated as:

$$w_{e_p,v^i} = \frac{\alpha}{d(v^i)} \tag{4}$$

In Eq. (4), α represents a customizable coefficient, $d(v^i)$ denotes the distance between the locations of cells in the current and last placement iteration. Following the quadratic wirelength minimization procedure, macros that share either a direct connection or an indirect connection through a limited number of standard cell clusters are adjusted to be in closer proximity.

The pin density is computed to gauge the utilization of routing resources in the FPGA after wirelength minimization. The calculation of the pin density follows the approach in [9]. Initially, the FPGA board is divided into multiple bins, and the pin density within each bin is determined by calculating the weighted sum of intersections between the bin and the bounding boxes of all nets.

$$PinDensity_i = \sum_{m\in N_i} \frac{w_m \cdot HPWL_m}{\text{\#bins covered by the net m}} \tag{5}$$

In Eq. (5), N_i represents the set of nets where the bounding boxes overlap with the bin i. w_m is the assigned weight proportional to the number of pins located in the bin i of the net m. $HPWL_m$ is the half perimeter wirelength of the net m, where $HPWL_m = (\max_j x_j - \min_j x_j) + (\max_j y_j - \min_j y_j)$.

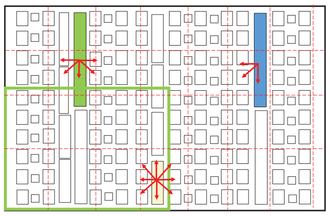


Fig. 6: The spreading directions for different macros with the same regional constraint (The constrained region is inside the green rectangle)

The cells are inflated according to the pin density of the bin, with a higher pin density resulting in a greater inflation rate. The cells in the bins with resource overflow caused by the inflation would relocate to the bins without overflow. This relocation serves to decrease the overall pin density across the FPGA board, while the wirelength between the cells may enlarge. In the subsequent iterations, the cells including macros would adjust their positions with the minimization of the wirelength. This iterative process results in an improved macro placement solution, optimizing both wirelength and pin density.

1) Handling the Relative Placement Constraints: RPC imposes restrictions on the relative positions of the macros and it is the common practice to amalgamate the macros with the same RPC into a large cascaded macro in the related works. However, these cascaded macros exhibit complicated interconnections with the other cells, and even a slight movement can result in subtle changes in wirelength. Frequent adjustments over a larger range can lead to unstable convergence in global placement. Moreover, the candidate sites for placing large cascaded macros are scarce in the FPGA board, particularly when some cascaded macros span half of the columns and the RC are taken into consideration simultaneously. Therefore, the sizes of the cascaded macros should be considered in the global placement. Here we modify the weights of the pseudo nets connecting macros to be highly related to their sizes as:

$$w_{e_p,v^i_{CM}} = \frac{\alpha}{d(v^i_{CM})} \times p(v^i_{CM}) \tag{6}$$

where $p(v^i_{CM})$ denotes the number of the pins of the macros. Larger cascaded macros with more pins are assigned the pseudo nets with larger weights and move relatively slower in the wirelength minimization.

2) Handling the Regional Constraints: The RC are considered in the stage of cell inflation and spreading. The cells with the RC are directed with the calculated distance to the constrained regions before and after the spreading. Let x_r^l and x_r^u be the x-coordinates of the left and the right boundary of the constrained region, then the horizontal distance d_x from the location x to the constrained region is calculated as:

$$d_x = \begin{cases} x - x_r^u, & x > x_r^u \\ x_r^l - x, & x < x_r^l \\ 0, & x_r^l \le x \le x_r^u \end{cases} \tag{7}$$

The vertical distance, denoted as d_y, is also calculated in a manner similar to Eq. (7). The spreading of the cells with RC ensures that the horizontal distance d_x and the vertical distance d_y do not increase. Illustrated in Fig. 6, the cells within and outside the constrained regions have different directional selections of spreading.

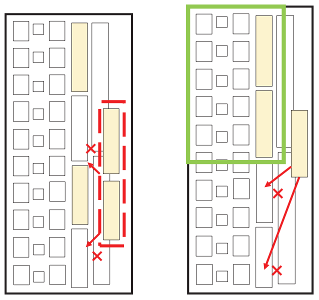

(a) Failure 1 (b) Failure 2

Fig. 7: Two failed macro legalization with the progressive minimum cost bipartite matching

C. Multi-stage Look Ahead Macro Legalization

The introduction of RPC and RC poses notable complexities in the macro legalization. RPC leads to the formation of excessively large cascaded macros with limited available sites to place. Meanwhile, the RC restricts the freedom for the macros to pursue a solution with less HPWL increase. Minimum cost bipartite matching is widely used in macro legalization, where one set of cells are macros, and the other set of cells are the candidate sites for each macro. The directed edges connect the macros and the candidate sites with the weight as the displacement after the legalization. Progressive legalization is proposed in [20] to speed up the traditional bipartite matching algorithms in the designs with thousands of macros and candidate sites. This approach involves iterative bipartite matching with a gradually increasing displacement threshold. However, in the cases of high site utilization and numerous constraints, progressive legalization may yield invalid results. Fig. 7 demonstrates two failures in progressive macro legalization. Fig. 7(a) shows a failure that the cascaded macro with two macros cannot find candidate sites for placement after two single macros are placed. Fig. 7(b) illustrates another failed case that a macro with the RC cannot be legalized within the region because all the candidate sites in the region are occupied by the other macros.

In order to eliminate the legalization failures caused by the design constraints, we categorize the macros into three distinct groups and perform sequential legalization on these categories. Alg. 1 outlines the procedures in the multi-stage legalization. The macros with RPC are legalized first, given the scarcity of candidate sites for placing such macros. Subsequently, the macros with the RC and those without the RC are legalized respectively. The macros without RPC and RC have the highest degree of freedom, as any site on the FPGA board is considered valid for their placement. These three categories of macros are placed with the progressive legalization algorithm in Alg. 2. However, the previously placed macros have the probability of occupying the sites in the constrained region, making it difficult for the macros with the RC to find the candidate sites in the same region. To solve this problem, some "look ahead" techniques are added to the progressive legalization. Specifically, the number of the sites occupied by the macros with the RC would be calculated initially (Line 1 in Alg. 1). This calculation ensures that the first-placed cascaded macros would not select sites in regions with resource overflow. In addition, the selection of candidate sites (Line 5 in Alg. 2) and the update of macro locations (Line 7-11 in Alg. 2) would also consider whether the constrained regions are resource overflow after placing the macro.

Algorithm 1: Multi-stage Look Ahead Macro Legalization

```
Input: List of the macros with RPC $I_{rpc}$, list of the macros
       with RC $I_{reg}$, list of the macros without any constraint
       $I_{sim}$, the sites $S$ on the FPGA board, the set of RC $R$,
       the locations after the global placement for all the
       macros $L_{gp} = \{x_i^{gp}, y_i^{gp}\}_{i=1}^M$
Output: The locations of the macros after the legalization
        $L_{leg} = \{x_i^{leg}, y_i^{leg}\}_{i=1}^M$
1 PreOccupyConstrainedRegions($R$)
2 ProgressiveLookaheadLegalize($I_{rpc}$, $L_{gp}$, $S$, $R$)
3 ProgressiveLookaheadLegalize($I_{reg}$, $L_{gp}$, $S$, $R$)
4 ProgressiveLookaheadLegalize($I_{sim}$, $L_{gp}$, $S$, $R$)
```

Algorithm 2: ProgressiveLookaheadLegalize

```
Input: The set of the macros to be legalized $I$, the sites $S$ on
       the FPGA board, the locations after the global
       placement for macros $L_{gp}$, the set of RC $R$, the initial
       threshold for the displacement $\theta_{init}$, the incremental of
       the displacement threshold $\Delta\theta$
Output: The locations of all the macros in $I$ after the
        legalization $L_{leg}^I$
1  Macro2CandidateSites = $\phi$
2  $\theta = \theta_{init}$
3  while there are macros in $I$ do
4      Macro2CandidateSites = FindCandidateSites($I$, $S$, $\theta$, $L_{gp}$)
5      RemoveSitesWithRegionsOverflow(Macro2CandidateSites,
        $I$, $S$, $R$)
6      Solve the Bipartite Matching
        $I \rightarrow$ Macro2CandidateSites and obtain the set of
        macro-site pairs Macro2Site
7      for each pair (macro, site) in Macro2Site do
8          if site does not cause resource overflow in $R$ then
9              UpdateMacroLocation(macro, site, $L_{leg}^I$)
10             UpdateRegionResourceUsage(site, $R$)
11             $I$.remove(macro)
12     $\theta = \theta + \Delta\theta$
```

IV. Experiment

A. Experiment Setups

We assess the performance of our proposed macro placer on the MLCAD 2023 benchmark[1], which is one of the largest benchmarks in the FPGA and EDA society, with 140 designs in total. It is targeted at the Xilinx Ultrascale+ xcvu3p FPGA board. Because of the page limit, we select 70 representative designs containing both the RPC and RC for evaluation. The selected designs exhibit a notable scale, with node counts ranging from 560k to 720k and net counts between 600k and 800k. Distinguishing itself from the ISPD 2016 [4] and ISPD 2017 [29] benchmarks, the MLCAD 2023 benchmark is characterized by large quantity and a high site utilization for the macros. The utilization of the sites placing macros rises from 50% to 85%. In addition, the benchmark contains plenty of realistic constraints in modern designs. In the selected designs, the percentage of the macros following RPC varies from 3% to 12%. The number of macros with the same RPC ranges from 2 to 60. The number of the constrained

[1]https://www.kaggle.com/datasets/ismailbustany/updated-mlcad-2023-contest-benchmark

Tab. I: Comparison of the routing scores after integrating different macro placers

Design	Vivado			MP			Ours			Design	Vivado			MP			Ours		
	s_i	s_d	s_t	s_i	s_d	s_t	s_i	s_d	s_t		s_i	s_d	s_t	s_i	s_d	s_t	s_i	s_d	s_t
Design_2	1	5	**6**	1	6	7	1	5	**6**	Design_90	1	7	**8**	2	7	9	2	8	10
Design_5	1	6	7	1	5	**6**	1	5	**6**	Design_92	2	7	9	1	7	8	1	6	**7**
Design_7	1	6	**7**	5	6	11	1	6	**7**	Design_95	1	7	8	1	6	**7**	1	6	**7**
Design_10	1	6	**7**	1	9	10	1	8	9	Design_97	1	7	8	3	8	11	1	6	**7**
Design_12	1	6	7	1	5	**6**	1	6	7	Design_100	5	8	13	1	6	**7**	1	7	8
Design_15	1	6	**7**	1	6	**7**	1	6	**7**	Design_102	6	9	15	1	7	**8**	1	7	**8**
Design_17	1	6	**7**	6	8	14	5	7	12	Design_105	1	7	8	1	6	**7**	1	7	8
Design_20	1	6	**7**	2	7	9	1	9	10	Design_107	3	13	16	1	10	11	2	8	**10**
Design_22	1	5	**6**	1	6	7	1	6	7	Design_110	5	11	16	10	10	20	1	9	**10**
Design_25	1	6	**7**	1	6	**7**	1	6	**7**	Design_112	1	9	10	1	8	**9**	1	8	**9**
Design_27	1	7	**8**	1	7	**8**	1	7	**8**	Design_115	1	8	9	1	6	**7**	1	6	**7**
Design_30	1	6	**7**	2	6	8	1	6	**7**	Design_117	2	15	17	6	10	16	5	8	**13**
Design_32	1	6	**7**	1	7	8	1	6	**7**	Design_120	22	23	45	9	15	24	1	8	**9**
Design_35	1	5	**6**	1	7	8	1	6	7	Design_122	1	6	**7**	1	6	**7**	1	6	**7**
Design_37	1	9	**10**	10	11	21	1	10	11	Design_125	1	6	**7**	1	6	**7**	1	6	**7**
Design_40	1	6	**7**	1	6	**7**	5	8	13	Design_127	1	7	**8**	3	8	11	1	7	**8**
Design_42	1	6	**7**	1	7	8	1	6	**7**	Design_130	1	7	**8**	1	8	9	1	7	**8**
Design_45	1	7	8	1	6	**7**	1	6	**7**	Design_132	1	6	**7**	1	8	9	1	6	**7**
Design_47	2	8	10	2	9	11	1	8	**9**	Design_135	1	9	10	1	6	**7**	1	7	8
Design_50	1	7	**8**	1	7	**8**	1	7	**8**	Design_137	17	9	26	5	8	**13**	16	8	24
Design_52	1	6	**7**	1	7	8	1	6	**7**	Design_140	16	13	29	14	13	27	9	9	**18**
Design_55	1	9	10	1	5	**6**	1	6	7	Design_142	1	7	8	1	7	8	1	5	**6**
Design_57	4	8	12	3	8	**11**	6	8	14	Design_145	1	8	9	1	8	9	1	6	**7**
Design_60	1	14	15	1	7	**8**	1	8	9	Design_147	3	10	13	2	8	**10**	5	7	12
Design_62	1	6	**7**	1	6	**7**	1	7	8	Design_150	2	10	12	2	9	11	1	8	**9**
Design_65	1	6	**7**	1	6	**7**	1	7	8	Design_152	1	7	**8**	1	8	9	1	7	**8**
Design_67	2	6	8	2	8	10	1	6	**7**	Design_155	1	6	**7**	1	7	8	1	6	**7**
Design_70	1	6	**7**	6	8	14	1	9	10	Design_160	3	8	**11**	14	11	25	11	10	21
Design_72	1	6	**7**	1	6	**7**	1	6	**7**	Design_162	2	9	11	1	7	8	1	6	**7**
Design_75	1	8	9	1	7	8	1	6	**7**	Design_165	1	8	9	1	7	**8**	1	8	9
Design_77	1	8	9	1	7	**8**	1	8	9	Design_167	6	18	24	2	11	**13**	19	12	31
Design_80	2	8	10	2	7	9	2	6	**8**	Design_170	17	37	54	11	12	**23**	19	23	42
Design_82	1	6	**7**	1	8	9	1	6	**7**	Design_172	1	8	**9**	1	8	**9**	2	7	**9**
Design_85	1	5	**6**	1	6	7	1	5	**6**	Design_175	1	9	10	1	7	**8**	1	10	11
Design_87	7	9	16	2	8	10	1	7	**8**	Design_180	7	27	34	19	36	55	6	13	**19**
Geomean											1.57	7.83	9.70	1.68	7.51	9.53	1.45	7.08	**8.88**
Ratio											1.0	1.0	1.0	1.07	0.96	0.98	0.92	0.9	**0.92**

regions varies from 2 to 22, with the percentage of cells subject to RC averaging at 24% and peaking at 42%. Our code has been open-sourced in https://github.com/tomqingo/CUMPLE_MLCAD.

In the evaluation flow, the FPGA macro placer would read the netlists and determine the positions for each macro. Subsequently, the standard cells are placed and the overall design is routed using Vivado ML 2021.1. The routing congestion report is generated in the end. Note that all the placements are in non-timing-driven mode. The experiments are all conducted on the server with Intel(R) Xeon(R) Platinum 8268 CPU. The number of threads used is 8.

B. Comparison with Academic and Commercial Tools

1) Routing Scores: We adopt the routing score proposed in [6] as a quantitative metric to assess the congestion level and the routability of the macro placement solution. The routing score consists of two components: the initial routing score and the detailed routing score.

$$s_t = s_i + s_d \tag{8}$$

The initial routing score s_i is derived from the interconnect congestion report provided by Vivado. This report encompasses the utilization of routing resources from the North, South, East, and West directions across the interconnect tile grid. Routing congestion is identified when resource utilization exceeds 0.9. Based on the interconnect wire length, congestion is further categorized into Global, Short, and Long Congestion. The congestion level ranges from 1 to 8. Since the congestion level under 3 has little degradation on the quality of routing, the initial routing score is the summation of the Global and Short congestion levels above 3 for all the directions,

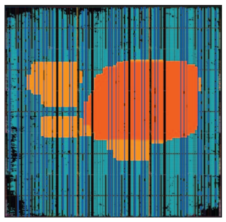
$$\begin{aligned} s_i = & 1 + \sum_{j=1}^{4} [max(0, short_j - 3)^2] \\ & + max(0, global_j - 3)^2] \end{aligned} \tag{9}$$

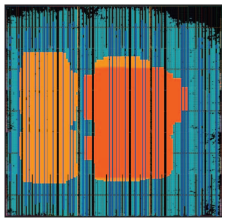
where $i = 1\ to\ 4$ stands for four directions. The detailed routing score s_d is calculated as the number of iterations for eliminating the nodes with overlap during the rip-up and reroute. Larger routing scores indicate larger congestion levels and more difficulties in fulfilling the FPGA routing.

We compare our macro placer with the default macro placer in Vivado ML 2021.1 and the DreamplaceFPGA-MP [30], which is the first place in the MLCAD2023 FPGA Macro-Placement Contest. Tab. I shows the routing scores for different designs and macro placers. "MP" denotes DreamplaceFPGA-MP. All the macro placement solutions satisfy RPC and RC, checked by the placement validity checker in Vivado. We calculate the geometry mean for the routing scores of all the designs. Compared with the default macro placer in Vivado, our placer could achieve 8% reduction in the initial routing score, 10% reduction in the detailed routing score, and 8% reduction in the total routing score on average. When compared with DreamplaceFPGA-MP, our placer could achieve 14% decrease in the initial routing score, 6% reduction in the detailed routing score, and 6% reduction in the total routing score on average.

It is also worth noticing that for some designs like Design_140 and Design_180, our macro placer can reduce the routing scores by nearly 40%. Fig. 8 demonstrates different macro placement solutions and the corresponding interconnect congestion levels above 3 in Design_140. The visualization shows the congestion level is lower and the congested area is smaller with our macro placement solution in this design. The comparative results above show that our macro placer could have better routability optimization for the designs with intensive constraints.

2) Place and Route Time: We also analyze the total place and route time after substituting the default macro in Vivado ML 2021.1 with our macro placer. The place time includes the time for macro placement and standard cell placement. Fig. 9 shows that after integrating our


(a) Vivado, s = 29


(b) MP, s = 27

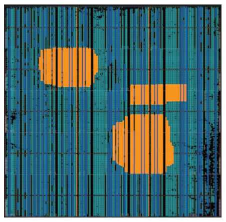

(c) Ours, s = 18

Fig. 8: The visualization of different macro placement solutions and the interconnect congestion levels above 3 for Design_140 (Arrays in deep blue represent the placed macros, arrays in light blue represent the placed CLBs, and area in orange represents the congested region)

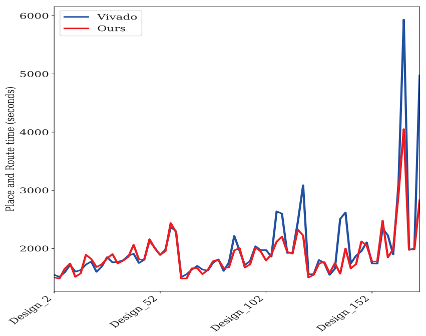


Fig. 9: Place and route time after integrating our macro placer in Vivado

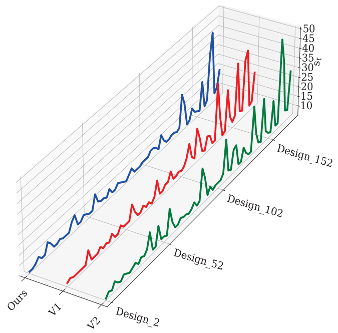


Fig. 10: Comparison of the total routing scores after introducing the techniques to align with design constraints in global placement

macro placer, the place and route time in Vivado could be reduced by 3.5% on average and the maximum drop reaches 43%. Although there are some overheads introduced by macro placement, the routing stage would be accelerated because of better routability.

C. Ablation Study

1) Techniques in Global Placement: More discussions are conducted on the techniques for adapting to the RPC and the RC. The two techniques used for achieving better routability within the design constraints are the macro size aware pseudo nets and the RC guided spreading. The former aims to reduce the frequent adjustments of the large cascaded macros, and the latter aims to ensure the macros with the RC would not spread out of the constrained regions. For evaluating these two techniques, we propose two versions by removing these two techniques respectively:

- V1 - the spreading does not consider decreasing the distance to the constrained regions.
- V2 - the weights of the pseudo nets connecting the macros are calculated as Eq. (4)

Fig. 10 shows the total routing scores for the macro placement solutions generated by V1, V2 and our macro placer. After integrating all of the two techniques, our macro placer performs best on most of the designs with fewer peaks, as shown in Fig. 10. We also calculate the geometry mean of the total routing scores. The geometric mean of the routing scores for V1, V2 and our macro placer are 9.80, 10.09 and 8.88 respectively. The introduction of the macro size aware pseudo nets and the RC guided spreading can bring the reductions in total routing scores by 12% and 9% respectively.

2) Techniques in Macro Legalization: We also evaluate the look-ahead techniques in multi-stage macro legalization. For comparison, we remove all the look-ahead techniques proposed in Section III-C, including Line 1 in Alg. 1 and Line 5, 8, 10 in Alg. 2. We rerun the versions with and without the look-ahead techniques on the selected designs. The time limit for the macro placement is set 10 minutes, with any macro placement that cannot finish in 10 minutes regarded as unsuccessful. The results show that the version with the look-ahead techniques succeeds in the placement for all 70 designs, while the version without the techniques fails on 8 designs. The unsuccessful macro placements stem from the unsuccessful bipartite graph construction in the progressive legalization. Some macros with RC cannot find the candidate sites that are already occupied by the other placed macros.

V. Conclusion

We propose a macro placer tailored for the Xilinx Ultrascale series FPGA boards addressing both routability optimization and realistic design constraints. It could be integrated into the Vivado environment with the introduction of the macro size aware pseudo nets and RC guided spreading in the global placement, and the multi-stage look ahead legalization. The experiment results on MLCAD2023 benchmark demonstrate that our macro placer can reduce congestion levels and facilitate routing, compared with the default macro placer in the Vivado and DreamplaceFPGA-MP. Additionally, the place and route time could be reduced by 3.5% on average and up to 43% after our macro placer is integrated in the Vivado. Ablation Study confirms the effectiveness of our techniques in achieving superior routability under RPC and RC.

Acknowledgement

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 14210923). We sincerely appreciate Tingyuan Liang for his help and guidance in our work.

References

[1] Q. Tang, M. Tuna, and H. Mehrez, "Performance comparison between multi-fpga prototyping platforms: Hardwired off-the-shelf, cabling, and custom," in *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2014, pp. 125–132.

[2] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai, "Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus?" in *2010 43rd annual IEEE/ACM international symposium on microarchitecture*. IEEE, 2010, pp. 225–236.

[3] S.-C. Chen and Y.-W. Chang, "Fpga placement and routing," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 914–921.

[4] S. Yang, A. Gayasen, C. Mulpuri, S. Reddy, and R. Aggarwal, "Routability-driven fpga placement contest," in *Proceedings of the 2016 on International Symposium on Physical Design*, 2016, pp. 139–143.

[5] Xilinx, "Ultrascale architecture and product data sheet: Overview," 2022.

[6] I. Bustany, G. Gasparyan, A. Gupta, A. B. Kahng, M. Kalase, W. Li, and B. Pramanik, "The 2023 mlcad fpga macro placement benchmark design suite and contest results," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. IEEE, 2023, pp. 1–6.

[7] C. Xu, W. Zhang, and G. Luo, "Analyzing the impact of heterogeneous blocks on fpga placement quality," in *2014 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2014, pp. 36–43.

[8] M. Gort and J. H. Anderson, "Analytical placement for heterogeneous fpgas," in *22nd international conference on field programmable logic and applications (FPL)*. IEEE, 2012, pp. 143–150.

[9] G. Chen, C.-W. Pui, W.-K. Chow, K.-C. Lam, J. Kuang, E. F. Young, and B. Yu, "Ripplefpga: Routability-driven simultaneous packing and placement for modern fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 10, pp. 2022–2035, 2017.

[10] W. Li, Y. Lin, and D. Z. Pan, "elfplace: Electrostatics-based placement for large-scale heterogeneous fpgas," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[11] R. S. Rajarathnam, M. B. Alawieh, Z. Jiang, M. Iyer, and D. Z. Pan, "Dreamplacefpga: An open-source analytical placer for large scale heterogeneous fpgas using deep-learning toolkit," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 300–306.

[12] D. Vercruyce, E. Vansteenkiste, and D. Stroobandt, "How preserving circuit design hierarchy during fpga packing leads to better performance," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 629–642, 2017.

[13] Y.-L. Chuang, G.-J. Nam, C. J. Alpert, Y.-W. Chang, J. Roy, and N. Viswanathan, "Design-hierarchy aware mixed-size placement for routability optimization," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2010, pp. 663–668.

[14] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings, "Hmflow: Accelerating fpga compilation with hard macros for rapid prototyping," in *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2011, pp. 117–124.

[15] R. Cofer and B. F. Harding, *Rapid System Prototyping with FPGAs: Accelerating the Design Process*. Elsevier, 2006.

[16] D. Park, Y. Xiao, and A. DeHon, "Fast and flexible fpga development using hierarchical partial reconfiguration," in *2022 International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 2022, pp. 1–10.

[17] L. Witschen, T. Wiersema, M. R. Nafchi, A. Bockhorn, and M. Platzner, "Timing optimization for virtual fpga configurations," in *International Symposium on Applied Reconfigurable Computing*. Springer, 2021, pp. 50–64.

[18] Xilinx, "Vivado design suite properties reference guide (ug912)," 2023.

[19] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. Eldafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. Walker *et al.*, "Vtr 8: High-performance cad and customizable fpga architecture modelling," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 2, pp. 1–55, 2020.

[20] T. Liang, G. Chen, J. Zhao, S. Sinha, and W. Zhang, "Amf-placer: High-performance analytical mixed-size placer for fpga," in *2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2021, pp. 1–6.

[21] Z. Huang, H. Sun, H. Wang, Z. Zhu, J. Yu, and J. Chen, "Late breaking results: An effective legalization algorithm for heterogeneous fpgas with complex constraints," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1362–1363.

[22] J. Mai, Y. Meng, Z. Di, and Y. Lin, "Multi-electrostatic fpga placement considering slicel-slicem heterogeneity and clock feasibility," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 649–654.

[23] M.-C. Kim, D.-J. Lee, and I. L. Markov, "Simpl: An effective placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 50–60, 2011.

[24] D. Vercruyce, E. Vansteenkiste, and D. Stroobandt, "Liquid: High quality scalable placement for large heterogeneous fpgas," in *2017 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2017, pp. 17–24.

[25] H. Hu, D. Fang, W. Li, B. Yuan, and J. Hu, "Systolic array placement on fpgas," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[26] N. Kapre, "Implementing fpga overlay nocs using the xilinx ultrascale memory cascades," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2017, pp. 40–47.

[27] X. Zhao, T. Wang, R. Jiao, and X. Guo, "Standard cells do matter: Uncovering hidden connections for high-quality macro placement," in *2024 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2024.

[28] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1398–1411, 2008.

[29] S. Yang, C. Mulpuri, S. Reddy, M. Kalase, S. Dasasathyan, M. E. Dehkordi, M. Tom, and R. Aggarwal, "Clock-aware fpga placement contest," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 159–164.

[30] Z. Xiong, R. S. Rajarathnam, Z. Jiang, H. Zhu, and D. Z. Pan, "Dreamplacefpga-mp: An open-source gpu-accelerated macro placer for modern fpgas with cascade shapes and region constraints," *arXiv preprint arXiv:2311.08582*, 2023.