Standard Cells Do Matter: Uncovering Hidden Connections for High-Quality Macro Placement

Xiaotian Zhao*, Tianju Wang*, Run Jiao†, and Xinfei Guo*‡§
*University of Michigan – Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University
†HiSilicon Technologies, ‡State Key Laboratory of Integrated Chips and Systems (SKLICS), §Corresponding author
{xiaotian.zhao, xinfei.guo}@sjtu.edu.cn

***Abstract*—It becomes increasingly critical for an intelligent macro placer to be able to uncover a good dataflow for a large-scale chip to reduce churns from manual trials and errors. Existing macro placers or mixed-size placement engines that were equipped with dataflow awareness mostly focused on extracting intrinsic relations among macros only, ignoring the fact that standard cell clusters play an essential role in determining the location of macros. In this paper, we identify the necessity of macro-cell connection awareness for high-quality macro placement and propose a novel methodology to extract all "hidden" relationships efficiently among macros and cell clusters. By integrating the discovered connections as part of the placement constraints, the proposed methodology achieves an average of 2.8% and 5.5% half perimeter wire length (HPWL) improvement for considering one-hop macro-cell and two-hop macro-cell-cell dataflow connections respectively, when compared against a recently proposed dataflow-aware macro placer. A maximum of 9.7% HPWL improvement has been achieved, incurring only less than 1% runtime penalty. In addition, the congestion has been improved significantly by the proposed method, yielding an average of 62.9% and 73.4% overflow reduction for one-hop and two-hop dataflow considerations. The proposed dataflow connection extraction methodology has been demonstrated to be a significant starting point for macro placement and can be integrated into the existing design flows while delivering better design quality.**

***Index Terms*—Macro placement, Standard cells, Dataflow awareness, QoR**

I. Introduction

Macro placement is typically manual and heavily relies on human experiences. Even experienced physical designers need to go through multiple iterations to get the macros placed "right" to some extent. It is challenging to fully automate the macro placement since it is almost impossible to accurately predict timing and other design information required by the subsequent placement and routing (P&R) processes based on very limited design and physical information. Though Reinforcement Learning (RL) or Deep Learning (DL)-based methods such as [1]–[4] have been proposed to offer benefits of automating the macro placement actions, they incurred legalization problems, huge computational, and the training design dataset is not easy to access. In addition, some of these techniques required a good starting point to work with [4]. Therefore, to achieve human-quality macro placement results, other techniques are still of great importance.

Recently, automatic macro placers have integrated design features to imitate the manual process, with dataflow being a prominent feature used by engineers to optimize macro positioning. Dataflow defines the moving of data between the macros and standard cells and influences the timing and power consumption of the circuit. The importance of dataflow awareness is reflected in massive efforts spent by physical designers for flyline analysis and frequent interactions with logic designers to understand detailed dataflow information.

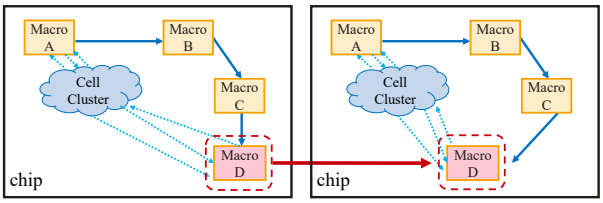


Fig. 1. Illustration of impact of standard cell clusters on macro placement result. The left figure illustrates the placement result that does not take into account the relationship between macros and cells. The right figure shows that the location of macros can be optimized by taking macro cluster-cell cluster relationships into consideration.

Since the interactions between macros and other modules are already available and ready to be analyzed from the synthesized netlist, and the macro count is still much less compared to standard cells in modern designs, thus extracting the detailed dataflow connections that involve macros is feasible computationally. This has inspired dataflow-aware macro placers [5]–[10], where the dataflow is analyzed and used as constraints for further placement of the macros. In [5], optimized data path layout was generated by utilizing rule information directly extracted from dataflow graphs. The approach in [7] considered dataflow and utilized the corner stitching data structure to address the problem of preplaced macros, which cannot be handled by simulated annealing (SA). A recently released macro placer, RTL-MP [8], considered the macro-to-macro dataflow and proved that such dataflow is helpful in getting a human-quality layout. An updated version of this placer was presented in [10]. Recently, commercial tool vendors also started to pay attention to dataflow analysis and equip some of their tools with such capability [11]–[13]. Therefore, the consideration of dataflow is critical for achieving high-quality macro placement results. Moreover, by minimizing the data movement, the placement of macros can reduce the overall design cost, as it reduces the routing process complexity and required interconnects number eventually.

Although the macro placers have been equipped with some forms of dataflow analysis capability, they mainly expose and utilize the virtual or direct connections among macros (i.e. macro-macro connections) because standard cell placement

This work was funded partially by a grant from Huawei HiSilicon Technologies (No. TC20220104476) and a SJTU Explore-X grant.

always tends to happen after the macro placement. While an important fact is that standard cells are usually placed in clusters based on their hierarchy. If the cluster is large enough, it can be modeled as a "macro". Thus the placement of standard cell clusters can in turn impact how macros are placed. This is illustrated by Fig. 1, where the routing resources can be optimized after moving the location of *Macro D* by taking into consideration of macro-cell connections and how data flows among these modules. A common practice in current design process, physical designers use feedback from frontend designers to create guides for standard cell clusters, integral to floorplanning and constraining cell placement.

Inspired by these, we propose a novel methodology to comprehensively extract all direct and indirect dataflow connections among macro clusters and cell clusters, and use them as an important guidance for macro placement. This methodology can be used in large-scale macro-dominant designs where engineers' experiences are limited. The major contributions are summarized as follows.

- The proposed methodology involves four different connection relationships including cell connections and placement in the macro placement process and defines a new dataflow-aware optimization target.
- The proposed dataflow extraction methodology offers a good starting point for any macro placers or mixed-size placers such as [14]. It complements the existing tools with newly discovered "hidden" connections that are essential for the final quality of results (QoR).
- Across a diverse group of benchmarks, the proposed work demonstrates an average of 2.8% and 5.5% in terms of HPWL improvement for discovering one-hop `macro-cell` connections and two-hop `macro-cell-cell` connections respectively, a maximum of 9.7% HPWL improvement has been achieved among all test cases. In addition, the congestion overflow has been improved 62.9% and 73.4% on average for considering one-hop and two-hop connections. The overall runtime penalty for extracting the full dataflow is only less than 1% of total runtime of the entire macro placement.
- The proposed methodology is able to uncover optimal macro placement results that otherwise violate engineering experiences such as push-boundary actions.

II. Overview of The Methodology

The overall macro placement methodology that incorporates the proposed dataflow connection extraction is depicted in Fig. 2, where key steps explained below.

During clustering, the synthesized netlist is firstly fed into a clustering engine, which is adapted from a recently proposed hierarchical auto-clustering method in [8]. This engine converts the structural netlist representation of the RTL design into a clustered netlist. We split and merge clusters when the number of macro or standard cells in it exceeds or is less than a preset threshold. Bundled IO pins are modeled as macro clusters in this process. The clustering also reduces the search cost for later connection extraction steps. There are

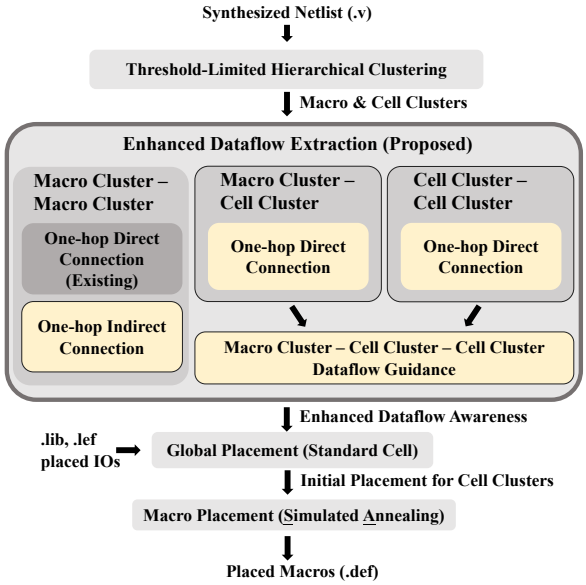


Fig. 2. The overall macro placement methodology. Light yellow rectangles outline all the newly proposed features in dataflow extraction.

only macros or only standard cells in each cluster and we define a cross-cluster connection as a one-hop.

With the generated macro or cell clusters, we explore the **dataflow connections extraction** in three categories shown in Fig. 2. The first category is among macro clusters, where most prior work focused on the one-hop direct connections between macros. We enhance this connection extraction and also consider potential indirect connections between macro clusters resulting from shared connections with the same cell clusters. The second category of dataflow connections is between macro clusters and cell clusters, where logical connections and their strengths are extracted and stored. The third category of connections is among cell clusters, this type of connection was usually ignored or hidden in previous macro placers, but it will still indirectly affect the macro locations. The connections of `macro cluster-cell cluster` and `cell cluster-cell cluster` will be combined into `macro cluster-cell cluster-cell cluster` dataflow guidance to further guide the macro placement.

In the proposed dataflow extraction method, the enhanced dataflow awareness provides a full picture of data movement in a design. With macro placement now highly dependent on cell clusters, we conduct a global placement (GP) to position the cell clusters initially. Macros are placed as the following step by incorporating the extracted dataflow information into a loss function. Then we leverage the Simulated Annealing (SA) algorithm in [8], [15] to optimize the loss function and represent the blocks in netlist by Sequence Pair [16]. Depending on the closeness to the converging point, the GP and macro placement can be run iteratively.

III. Proposed All-Connection Extraction Method

The existing macro placers mainly consider the connections among macros in the design. However, it is also crucial to analyze the logical relationships of the `macro cluster-cell cluster` and among the cell clusters. This section will introduce the proposed connection extraction processes

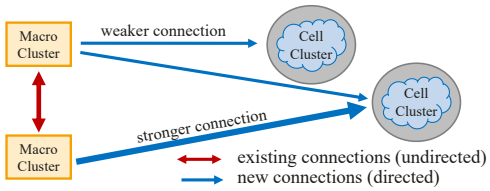


Fig. 3. One-hop direct dataflow connection of macro cluster-cell cluster. The strength is given based on the bit width of the extracted connections.

including indirect `macro cluster-macro cluster`, `macro cluster-cell cluster`, and `macro cluster-cell cluster-cell cluster` connections.

A. One-Hop Direct Macro Cluster-Cell Cluster Dataflow

The dataflow from the netlist is converted into a graph, where the prior work mostly modeled the `macro cluster-macro cluster` connections as undirected. As can be seen in Fig. 3, in addition to these undirected macro-macro connections, we also take direction into consideration when extracting the connections between macros and cells, enabling more types of logical relationships. Though breadth-first search (BFS) is used, we introduce pruning in the search process and optimize the storing data process during graph building and traversal. The strength of connections is defined based on the data bit-width. To incorporate the impact of the dataflow, a loss function is defined in Equation 1. This loss function is utilized to guide the subsequent solution search process, which is iterated to optimize the wirelenghth as the final quality indicator. In Equation 1, WL is half perimeter wire length (HPWL) and w_x is a weight factor which is defined by dataflow bit_width between the clusters.

$$w_0 = bit\_width(macro\_cluster_1, macro\_cluster_2)$$
$$w_1 = bit\_width(macro\_cluster, cell\_cluster)$$
$$loss = w_0 * WL_{macro-macro} + w_1 * WL_{macro-cell} \quad (1)$$

B. One-Hop Indirect Macro Cluster-Macro Cluster Dataflow

For certain designs, macro clusters may not have a direct connection among themselves, but they may share connections with the same cell clusters. Such "hidden" connection can indicate indirect dataflow among macro clusters and further guide the macro placement. This type of connection is challenging to find due to the scale of the search and the complexity involved in excluding common signals, such as clock and reset signals. In the proposed method, we extract these virtual connections in a hierarchical way. This has been illustrated in Fig. 4.

The first process is explained in line 2-12 of Algorithm 1. We first treat the cell cluster as the smallest unit. After traversing all the clusters, we use the $addconnection$ function to establish virtual connections between macros that share common connections to the same cell cluster (shown Fig. 4 left). Then we treat the single standard cell instance as the smallest unit and look for their macro fanouts. If a single cell drives multiple macros, we view these macros as being virtually connected (Fig. 4 right). Line 14-21 in Algorithm 1 details the extraction of such connections. By far, we have supplemented the relationship among macro clusters by considering all direct and indirect connections between macro clusters. We also define the new loss function for the indirect

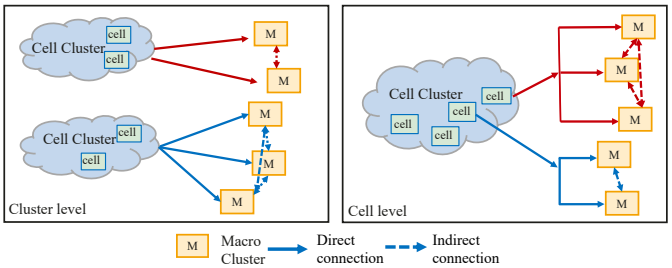


Fig. 4. Illustration of one-hop indirect dataflow connection from macro cluster-macro cluster. The left figure treats the cell cluster as the smallest unit, and the right figure treats the single cell instance as the smallest unit.

Algorithm 1 Indirect Macro Dataflow Connection Extraction Algorithm

Require: $.lef$, $.lib$, $.v$, $constraints$
Ensure: two types of indirect macro connections
1: ▷ At cell cluster level (Fig. 4 left)
2: **for** each cell cluster (in cell-macro connection) **do**
3: $vector$: $cell\_connected\_macro\_vec$
4: $num = cell\_connected\_macro\_vec.size$
5: **for** i = 0 to num **do**
6: **for** j = i to num **do**
7: $macro\_src = cell\_connected\_macro\_vec[i]$
8: $macro\_sink = cell\_connected\_macro\_vec[j]$
9: $addconnection\ (macro\_src, macro\_sink)$
10: **end for**
11: **end for**
12: **end for**
13: ▷ At cell instance level (Fig. 4 right)
14: **for** each vertex in cell fanin map **do**
15: $vector$: $same\_vertex\_macro\_vec$
16: **for** each pin (in vertex pin list) **do**
17: id = find the $macro\_id$ where pin located
18: **if** id is in a macro cluster **then**
19: $same\_vertex\_macro\_vec.push\_back(id)$
20: **end if**
21: **end for** ▷ Find macros connected to the same cell
22: $total = same\_vertex\_macro\_vec.size$
23: **for** i = 0 to $total - 1$ **do**
24: **for** j = i+1 to total **do**
25: $macro\_src = same\_vertex\_macro\_vec[i]$
26: $macro\_sink = same\_vertex\_macro\_vec[j]$
27: $addconnection\ (macro\_src, macro\_sink)$
28: **end for**
29: **end for**
30: **end for**
31: **return** indirect macro connections

macro cluster connection in Equation 3, where w_i is defined by the sum of dataflow bit_width between two macro clusters to cell cluster and WL represents the HPWL.

$$w_i = bit\_width(macro\_cluster_1, cell\_cluster) + bit\_width(macro\_cluster_2, cell\_cluster) \quad (2)$$

$$loss_{indirect\_macro} = w_i * WL_{indirect\ macro-macro} \quad (3)$$

C. Two-Hop Dataflow of Macro Cluster and Cell Cluster

The ultimate goal of this work is to establish dataflow of `macro cluster-cell cluster-cell cluster` to further analyze the impact of multi-hop dataflow connection. This dataflow can be divided into two parts, which are `macro cluster-cell cluster` connection that is already discussed in Section III-A and `cell cluster-cell cluster` connections. We also consider the size of the cell clusters based on the intuition that larger cell clusters usually have a greater impact on the position of virtually connected macros. Therefore, we use the product of constant k, bit_width, area, and instance number between the cell cluster which shown in Equation 4 to weight the connection between cell clusters.

$$w_j = k * bit\_width * cluster_{area} * cluster_{number} \quad (4)$$

After extracting the relationship among cell clusters shown in Algorithm 2, we convert them into two-hop connections

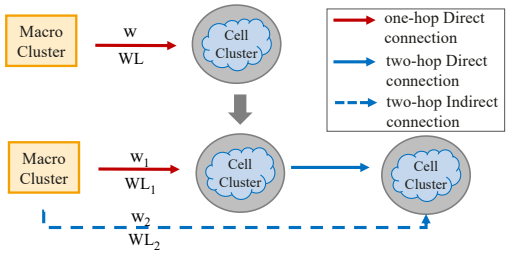


Fig. 5. Illustration of newly established two-hop dataflow connection macro cluster-cell cluster-cell cluster.

Algorithm 2 Macro Cluster-Cell Cluster-Cell Cluster Connection Extraction Algorithm (Fig. 5)

Require: $.lef$, $.lib$, $.v$, $constraints$
Ensure: connections of macro cluster-cell cluster-cell cluster
1: $map : macro\_cell\_connect$
2: **for** each connection **do**
3: **if** src is macro && sink is cell **then** ▷ First connect macro-cell
4: $addconnection\ (macro\_src, cell\_sink)$
5: $macro\_cell\_connect[sink.id].first = 1$
6: $macro\_cell\_connect[sink.id].second = src\_id$
7: **end if**
8: **if** src is cell && sink is cell **then** ▷ Then connect cell-cell
9: **if** macro_cell_connect[src_id].first == 1 **then**
10: ▷ Checking whether this cell is connected to a macro
11: **if** src_id != sink.first **then**
12: $macro\_id = macro\_cell\_connect[src\_id].second$
13: $addconnection\ (macro\_id, cell\_sink)$
14: $addconnection\ (cell\_src, cell\_sink)$
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **return** macro cluster-cell cluster-cell cluster connections

between macro clusters and cell clusters as shown in Fig. 5. The conversion is based on weighting two-hop virtual connections higher than the one-hop direct connection. In line 3-7, we first label the macro clusters that are directly connected to the cell cluster. After the connection between cell clusters is traversed, we return to see if there is a labeled macro cluster. Once such a macro is found, a `macro cluster-cell cluster-cell cluster` dataflow connection is established. We use the design `swerv_wrapper` to show the impact of a two-hop connection in Fig. 6. It shows that these green clusters which have second-hop connections with macros occupy a large area and will impact the macro positioning.

The final updated definition of the loss function is shown in Equation 5,

$$loss = w_0 * WL_{m-m} + w_1 * WL_{m-c} + w_2 * WL_{m-c-c} \quad (5)$$

where w_0, w_1 and w_2 are defined in Equation 2, Equation 1 and Equation 4, and WL_{m-m} refers to total HPWL of `macro cluster-macro cluster` connections, WL_{m-c} means HPWL of `macro cluster-cell cluster` connections, WL_{m-c-c} represents HPWL of the second-hop connection from `macro cluster-cell cluster-cell cluster`.

IV. Evaluation Results

A. Experiment Setup

The proposed method is implemented in C++ and the database is compatible with what has been used in the latest released OpenROAD [17], [18] flow. The flow runs on Intel Core i7 11700 CPU with 128GB of memory allocated. The netlist is obtained by running synthesis with Yosys [19]. We conduct extensive experiments and select seven benchmark designs from published literature or newly released test suites

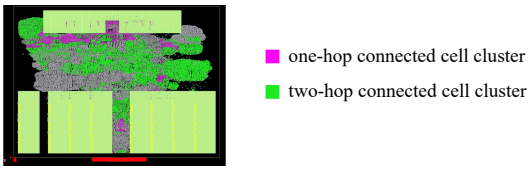


Fig. 6. Comparison between one-hop connected cell cluster (only macro cluster-cell cluster) and two-hop connected cell cluster (after considering macro cluster-cell cluster-cell cluster) in an example design `swerv_wrapper`.

for macro placement evaluation [18], [20]. We test the designs in NanGate45 [21] technology node. To measure the quality of the macro placement, we use half-perimeter wire length (HPWL) and congestion overflow as our main comparison metrics. Clustering is done with the method introduced in Section II, followed by the proposed extraction method to obtain all the dataflow connections and define the loss function. We leverage the existing Simulated Annealing (SA) method in [8] to further optimize the loss function and open-source tools for placement and routing (P&R).

The proposed method is compared against Triton Macro Placer (TMP) [22], a default macro placer within OpenROAD, and RTL-MP [8], a recently released dataflow-aware macro placer. To quantify the impact of different dataflow connections, we first measure the impact of `macro cluster-cell cluster` dataflow only and then the `macro cluster-cell cluster-cell cluster` dataflow. The full comparison is summarized in Table II, and the analysis will be detailed in the following sections.

B. Weight Analysis

Automatic macro placers always use preset or tuned weights and coefficients, which may be less portable across designs. In contrast, to set the weights for loss function and SA model, we incorporate the actual dataflow exchange bit width as part of the weight. In the `cell cluster-cell cluster` model, we consider the cluster size and convert it into weights. The main optimization target in the macro placement stage is wirelength with the consideration of other metrics such as area and legalization. The weights of wirelength targets are set to be larger than others. To make a fair comparison against other macro placers such as TMP [22] and RTL-MP [8], we keep the rest of weighting parameters the same.

C. Dataflow Connection Relationship Analysis

Table I lists the statistics of all benchmarks and the number of unique dataflow connections extracted based on the proposed method. Among them, `TinyRocket` is a tiny design with only two macros and very few standard cells, but the number of `cell cluster-cell cluster` connections is large. Though the number of macros for `bp_multi` is closer to `black parrot`, they exhibit very different dataflow behaviors. Specifically, `bp_multi` has significantly fewer `macro cluster-cell cluster`, `cell cluster-macro cluster`, and `cell cluster-cell cluster` connections compared to `black parrot`. `ariane133` represents the macro-heavy design with only one unique type. However, the number of unique connections in `ariane133` is sometimes less than other benchmarks. By conducting such analysis, the macro placement can be tuned to be more design-specific. This is

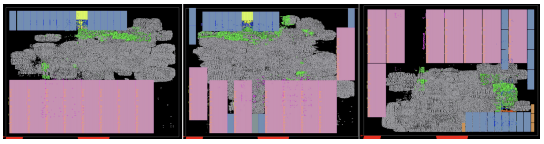
TABLE I
BENCHMARK INFORMATION AND EXTRACTED DETAILED CONNECTIONS FOR EACH DESIGN

Design Name	Std Cells Count	Macros Count	Macro Type [1]	# of IOs	Total Cluster Count	**Macro Cluster-Macro Cluster** *	**Macro Cluster-Cell Cluster** * [2]	**Cell Cluster-Macro Cluster** * [2]	**Cell Cluster-Cell Cluster** *	**Macro Cluster-Cell Cluster-Cell Cluster** *
TinyRocket	27217	2	1	269	65	0	2	35	829	67
bp_be	59882	10	3	3029	139	25	120	364	1815	251
bp_fe	29993	11	3	2511	82	24	132	260	406	196
black parrot	427501	24	5	1198	763	2	565	1095	14649	3375
bp_multi	209086	26	6	1453	432	2	351	828	4962	1427
swerv_wrapper	99750	28	3	1416	518	0	232	741	15300	594
ariane133	165953	133	1	495	314	44	380	1947	9963	1708

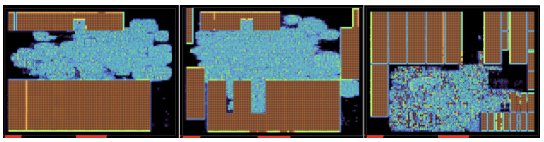
* # of unique connections.
[1] This includes macros with different sizes and functionalities.
[2] Bi-directional connections.

TABLE II
EXPERIMENTAL RESULTS

Design Name	TMP [17]		RTL-MP [8] (baseline)		**Proposed Method**					
	HPWL (m)	Congestion Overflow	HPWL (m)	Congestion Overflow	**Macro Cluster-Cell Cluster HPWL (m)**	**HPWL Improvement**	**Congestion Overflow**	**Macro Cluster-Cell Cluster -Cell Cluster HPWL (m)**	**HPWL Improvement**	**Congestion Overflow**
TinyRocket	830.03	73	803.80	74	758.98	5.5%	26	**752.80**	6.3%	**19**
bp_be	4151.71	66553	4445.60	40537	4218.92	5.1%	2224	**4113.33**	7.5%	**454**
bp_fe	2772.04	3373	2789.94	4445	2664.61	4.5%	3165	**2519.63**	9.7%	**3156**
black parrot	12966.52	142	12930.57	981	12600.53	2.5%	272	**11820.00**	8.6%	**130**
bp_multi	7496.04	2854	7252.76	3965	7235.15	0.2%	**1328**	**7146.48**	1.5%	1470
swerv_wrapper	5199.62	3384	4745.86	2533	4718.80	0.5%	2299	**4648.02**	2.1%	**887**
ariane133	8775.61	3421	8857.12	23876	8737.35	1.3%	**3842**	**8624.48**	2.6%	3919
Avg. Improvement	0.9% ↓	4.4% ↓	0%	0%	-	2.8% ↑	62.9% ↑	-	**5.5%** ↑	**73.4%** ↑

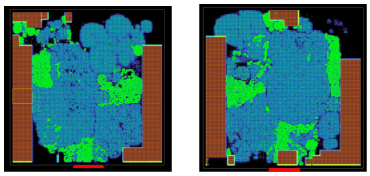


(a) Proposed, RTL-MP and TMP macro placement result



(b) Proposed, RTL-MP and TMP congestion map

Fig. 7. Layouts and congestion map of `swerv_wrapper` with different macro placement flows. Use the same color to highlight the directly connected cell instances and macros and one-hop connected cell clusters are colored green.



(a) RTL-MP (b) Proposed

Fig. 8. Layouts of `black parrot` with congestion map. The highlighted green cells is from the one-hop connected cell cluster.

especially useful when a new design is implemented and no prior experience has been provided yet.

D. HPWL Result Analysis

Table II summarizes HPWL results obtained by running TMP, RTL-MP, and the proposed two different methods, where the best data for each design has been highlighted in bold and the improvement (in terms of %) over RTL-MP (most of the time better than TMP) is listed alongside. The results are separated into two categories, which are denoted as `macro cluster-cell cluster` and `macro cluster-cell cluster-cell cluster`. The first category considers both direct and indirect `macro cluster-macro cluster` connections and the one-hop `macro cluster-cell cluster` connections. The second category considers two-hop `macro cluster-cell cluster-cell cluster` connections and all connections of the first one. Results demonstrate that the proposed method outperforms RTL-MP for all designs and leads to an average HPWL improvement of 2.8% and 5.5% in two categories respectively. A maximum of 9.7% HPWL improvement has been achieved for design `bp_fe`.

`swerv_wrapper` is a design that has a relatively large number of macro and cell clusters. Take it as an example, Table II shows that our approach achieves up to 2.1% better HPWL result than the other two methods, as well as excellent placement and congestion outcomes, as shown in Fig. 7(a) and Fig. 7(b). Highlighted cells are those have direct or indirect connections to macro clusters. It is clearly shown that our proposed method is able to place these cells closer to their associated macros. The extracted connections are able to guide the placement of both types of instances. The resulting macro placement is similar to manual efforts, where macros of the same hierarchy are clustered for improved legalization—a task not accomplished by the other two macro placers. Our method also results in better utilization and leaves more empty areas for further optimization.

Similar optimization has been achieved for `black parrot`, which has more types of macros and more connections among macro clusters and cell clusters compared to `swerv_wrapper`. The placement results are shown in Fig. 8(a) and Fig. 8(b). It is worth mentioning that the previous macro placer [8] failed to identify the cell cluster connection with certain macros closer to the pin access region and led to a longer wirelength. This has been fixed in our dataflow connection extraction method. Although blocking pin access seems contradictory to engineering practices, our experiments show that it can be beneficial in certain cases. This is also one area where our method can potentially outperform human efforts by allowing "non-conventional" guidance.

E. Congestion Overflow Result Analysis

By taking consideration of "hidden" connections, the congestion situation has also been improved significantly. This has been reflected in the reduced overflow in Table II. The proposed method achieves an average of 62.9% and 73.4% overflow reduction in two categories respectively. This is due to the fact that the macros and their associated cell clusters are placed closer, long and zigzag wires can be avoided. More routing resources can thus be allocated.

TABLE III
DATAFLOW EXTRACTION RUNTIME ANALYSIS

Design Name	RTL-MP [8]	Proposed Dataflow Extraction	
	Runtime (s) [1]	Macro Cluster-Cell Cluster Runtime (s) [2]	Macro Cluster-Cell Cluster-Cell Cluster Runtime (s) [3]
TinyRocket	1	7	10
bp_be	7	17	24
bp_fe	2	4	7
black parrot	323	417	512
bp_multi	56	104	139
swerv_wrapper	20	69	106
ariane133	54	374	608
Average Runtime (s)	66.14	141.71 (2.14×)	200.86 (3.04×)

[1] Extracting macro cluster-macro cluster connections only within RTL-MP.
[2] Includes extracting connections among macro clusters.
[3] Includes extracting previous two types of connections in 1 and 2.

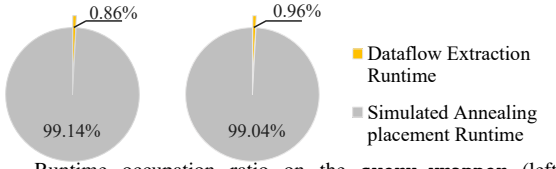


Fig. 9. Runtime occupation ratio on the `swerv_wrapper` (left) and `ariane133` (right) test cases.

F. Optimization Runtime Analysis

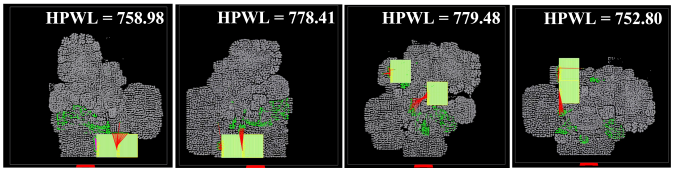
The runtime is compared against RTL-MP [8] and is summarized in Table III. It can be observed that extractions that include the `macro cluster-cell cluster` and `macro cluster-cell cluster-cell cluster` connections lead to an average $2.14\times$ and $3.04\times$ runtime respectively. Despite the increased runtime on the extraction part, the incurred overhead is still less than 1% of the total runtime of the entire macro placement process, as shown in Fig. 9. It is worth mentioning that the extraction runtime only increases sub-linearly with the number of connections, yielding a worthy design quality gain with marginal runtime penalty.

G. The Impact of "Push Boundary" Action

A common practice in floorplanning is to push macros closer to the boundary to create more spaces for standard cells. With the proposed method, we find out that sometimes it is totally acceptable to violate such "rule". Since stronger dataflow connections can lead to a significant increase in wirelength if macros are pushed to the edge far from standard cells. Therefore, such restrictions are relaxed in our experiment. An example is shown in Fig. 10. The design is `TinyRocket`, which is cell dominant and has a large number of `cell cluster-cell cluster` connections. The table summarizes differences in HPWL when adding or removing pushing-boundary constraints. It shows that when considering only the one-hop connection, "pushing boundary" has better HPWL because it involves fewer cell clusters. However, when analyzing the two-hop connections, "not pushing boundary" has a better performance on HPWL which achieves overall 6.3% improvement over RTL-MP. This indicates that the proposed method is able to identify those opportunities that are otherwise invalidated by human experiences.

V. CONCLUSIONS

In this paper, we identified the importance of dataflow analysis in macro placement and proposed a methodology that is able to extract detailed dataflow connections among macro clusters and between macro and cell clusters. These extracted connections were further incorporated into the loss function for guiding subsequent macro placement steps. We demonstrated with a set of diverse benchmarks that the proposed methodology outperforms the recently released academia dataflow-aware macro placer in terms of HPWL and congestion improvements. This work raises the awareness of detailed and comprehensive dataflow analysis in developing automatic macro placers. It provides an opportunity to co-optimize macro placement and standard cell placement.



	Macro Cluster-Cell Cluster HPWL (m)	Macro Cluster-Cell Cluster-Cell Cluster HPWL (m)
Push boundary	758.98	778.41
Not push boundary	779.48	**752.80**

Fig. 10. The proposed method placement result of `TinyRocket` with "push boundary" and "not push boundary" actions.

ACKNOWLEDGEMENT

Authors would like to acknowledge Dr. Hongzhong Wu and Dr. Yu Huang from Huawei HiSilicon for their helpful insights for this project.

REFERENCES

[1] A. Mirhoseini *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.
[2] Y. Liu *et al.*, "Floorplanning with graph attention," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1303–1308.
[3] A. Agnesina *et al.*, "Autodmp: Automated dreamplace-based macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, 2023, pp. 149–157.
[4] C.-K. Cheng *et al.*, "Assessment of reinforcement learning for macro placement," *arXiv preprint arXiv:2302.11014*, 2023.
[5] T. T. Ye *et al.*, "Data path placement with regularity," *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, vol. 2000-Janua, pp. 264–270, 2000.
[6] A. Vidal-Obiols *et al.*, "RTL-Aware Dataflow-Driven Macro Placement," *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, pp. 186–191, 2019.
[7] J.-M. Lin *et al.*, "Dataflow-aware macro placement based on simulated evolution algorithm for mixed-size designs," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 29, no. 5, pp. 973–984, 2021.
[8] A. B. Kahng *et al.*, "Rtl-mp: toward practical, human-quality chip planning and macro placement," in *Proceedings of the 2022 International Symposium on Physical Design*, 2022, pp. 3–11.
[9] A. Vidal-Obiols *et al.*, "Multilevel dataflow-driven macro placement guided by rtl structure and analytical methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 12, pp. 2542–2555, 2021.
[10] A. B. Kahng *et al.*, "Hier-rtlmp: A hierarchical automatic macro placer for large-scale complex ip blocks," *arXiv:2304.11761*, 2023.
[11] MAXEDA, "Maxplace," http://www.maxeda.tech/maxplace.html.
[12] Synopsys, "Freeform macro placement technology," https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html.
[13] Cadence, "Innovus mixed placer," https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/innovus-mixed-placer.
[14] J. Lu *et al.*, "eplace-ms: Electrostatics-based placement for mixed-size circuits," vol. 34, no. 5, pp. 685–698, 2015.
[15] S. Kirkpatrick *et al.*, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[16] H. Murata *et al.*, "Vlsi module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.
[17] A. B. Kahng *et al.*, "The openroad project: Unleashing hardware innovation," in *Proc. GOMAC*, 2021.
[18] "Openroad-flow-script," https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.
[19] C. Wolf, "Yosys open synthesis suite," https://github.com/YosysHQ/yosys, 2016.
[20] "Tilos macro placement benchmark," https://github.com/TILOS-AI-Institute/MacroPlacement/tree/main/Testcases.
[21] "Nangate 45nm library," http://www.nangate.com/.
[22] "Triton macro placer," https://github.com/The-OpenROAD-Project/TritonMacroPlace.