GNN-based Multi-bit Flip-flop Clustering and Post-clustering Design Optimization for Energy-efficient 3D ICs

PRUEK VANNA-IAMPIKUL, YI-CHEN LU, DA EUN SHIM, and SUNG KYU LIM, Georgia Institute of Technology, USA

In high-performance three-dimensional Integrated Circuits (3D ICs), clock networks consume a large portion of the full-chip power. However, no previous 3D IC work has ever optimized 3D clock networks for both power and performance simultaneously, which results in sub-optimal 3D designs. To overcome this issue, in this article, we propose a GNN-based flip-flop clustering algorithm that merges single-bit flip-flops into multi-bit flip-flops in an unsupervised manner, which jointly optimizes the power and performance metrics of clock networks. Moreover, we integrate our algorithm into the state-of-the-art 3D physical design flow and verify the integration, which leads to a better 3D full-chip design. Experimental results on eight industrial benchmarks demonstrate that the algorithm achieves improvements up to 18% in total power and 8.2% in performance over the state-of-the-art 3D flow.

CCS Concepts: • **Hardware → Clock-network synthesis**; **Physical synthesis**; **3D integrated circuits;**

Additional Key Words and Phrases: Flip-flop clustering, graph neural network, Random forest, Pseudo-3D design flow

ACM Reference format:
Pruek Vanna-iampikul, Yi-Chen Lu, Da Eun Shim, and Sung Kyu Lim. 2023. GNN-based Multi-bit Flip-flop Clustering and Post-clustering Design Optimization for Energy-efficient 3D ICs. *ACM Trans. Des. Autom. Electron. Syst.* 28, 5, Article 76 (September 2023), 26 pages.
https://doi.org/10.1145/3588570

1 INTRODUCTION

Due to the lack of commercial three-dimensional (3D) physical design tools, existing 3D **Integrated Circuits (IC)** implementation flows leverage pseudo-3D approaches [1] to build commercial-quality 3D ICs from 2D commercial tools. Mainly, these 3D flows rely on 2D tools to perform pseudo-3D placement and routing on projected 2D layouts. To improve the power consumption of the final full-chip design, existing 3D flows merely focus on improving the switching power, given that it can be straightforwardly achieved through wirelength reduction. However, the clock and the internal cell power, which constitute a significant portion of the total power [2], need to be optimized directly in the existing 3D design flows. In addition, due to the inferiority of the tier partitioning algorithm named bin-based min-cut algorithm, which most of the previous

Authors' address: P. Vanna-iampikul, Y.-C. Lu, D. E. Shim, and S. K. Lim, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, Georgia, USA, 30332; emails: v.pruek@gatech.edu, yclu@gatech.edu, daeun@gatech.edu, limsk@ece.gatech.edu.

1084-4309/2023/09-ART76 $15.00
https://doi.org/10.1145/3588570

works [3] adopt, severe timing degradation often appears in the final 3D full-chip designs due to the ill-decided locations of registers. Therefore, a methodology that jointly improves the power and performance of 3D ICs is urgently needed in this work.

Generally speaking, 3D design flows can be categorized into two categories: partitioning-first and partitioning-last. Previous work [4], a partitioning-last design flow [1], has proposed a partition mitigation strategy to improve the power and performance of 3D ICs by partitioning clock buffers and **flip-flops (FFs)** based on a clock tree hierarchy while moving cells on critical paths within clusters to prevent skew degradation. However, this heuristic algorithm requires parameter tuning for each design benchmark and only shows marginal improvements after long tuning iterations. In partitioning-first design flow, previous work [5] attempted to resolve the timing degradation by enhancing placement constraints. Despite the performance improvement, the clock and sequential power still need to be fully optimized to have more power saving than the 2D ICs counterparts.

The clock and sequential power have lately become essential factors in the total power due to the high clock frequency in the gigahertz range. A clock delivery network consists of a source and multiple sinks (=FFs) [6]. The source is distributed from the center of the footprint with buffers and inverters in the tree structure and, last, to the leaf nodes. Different types of FFs are designed to support many functionalities, and their power estimations differ [7]. Therefore, the algorithm to co-optimize the clock delivery network for timing and power is mandatory in 2D and 3D design.

In this article, we propose joint power and timing optimization for 3D ICs through FF clustering. Merging single-bit FFs to **multi-bit flip-flops (MBFF)** is well known to help optimize clock power and timing in 2D designs [8], which has become a must-use technique in industrial design flows. MBFF provides power saving by sharing the same clock buffers with multiple sets of master–slave latches. However, there exist a few drawbacks in existing 2D MBFF algorithms. By focusing only on an arrival time constraint, another FF clustering approach [9] becomes applicable to industrial designs, because the arrival time is not coupled between FFs. However, this clustering approach only optimizes the clock network locally, because it only clusters neighboring flops whose arrival constraints are not violated, severely limiting the total number of FFs available to be clustered. Nonetheless, the most severe drawback of previous works [8, 9] is that after flop clustering, the underlying placement is not accordingly improved with the new locations of flops. They rely on the subsequent routing stages and optimizations to resolve the violations.

In this work, we advance the traditional MBFF algorithms through machine learning. Furthermore, we develop a novel learning-based flop clustering framework that improves both 2D and 3D ICs by jointly optimizing the performance and power metrics. Specifically, we propose a novel clock network optimization technique called **Multi-bit FF clustering (MBFC)** that can be integrated with any design implementation flow. Our approach is elegantly based on **graph neural networks (GNNs)** and essential features that learn to find good clustering results by understanding netlist characteristics. The main contributions of this work are listed as follows:

- We extend the state-of-the-art 3D design flows with the ability to perform effective FF clustering to improve 3D QoR metrics.
- We propose a novel GNN-based approach with essential features to cluster the clock cells from similarities instead of the traditional clustering algorithm.
- We further analyzed and quantified the impact of each feature on the final clustering results.
- The GNN-based flip-flop clustering algorithm reduces parameter tuning efforts in a practical runtime.
- Our framework improves the final 3D full-chip design using an efficient multi-bit flip-flop clustering algorithm. Additionally, the framework also supports the 2D IC design flow.

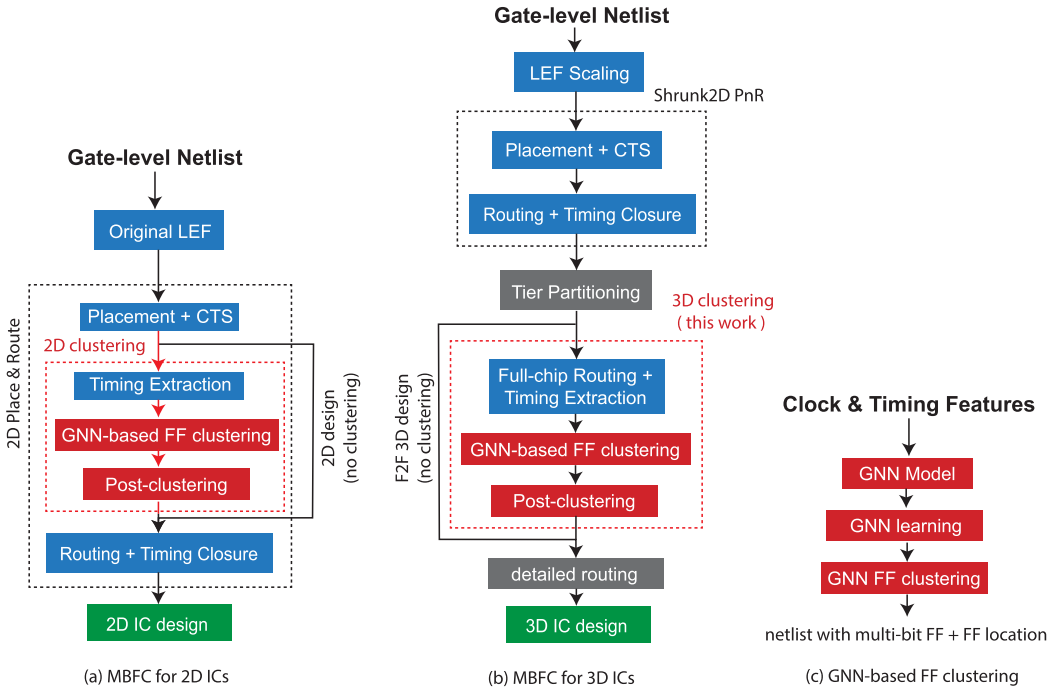


Fig. 1. Our MBFC, which supports both 2D and 3D ICs.

- Experimental results demonstrate that our proposed algorithm outperforms the existing 3D state-of-the-art flows with up to *18% and 8.2% improvements in the total power and performance*, respectively.

2 RELATED WORKS

In the state-of-the-art **Shrunk2D (S2D)** [3] flow, there are four main steps. First, for a given technology node, cells and wires are scaled down by a $1/\sqrt{2}$ along both the x and y dimensions. Second, the design is synthesized with this scaled technology using a 2D Back-End-Of-Line. Third, the design is partitioned into two dies. Last, **Face-to-Face (F2F)** pads are inserted, and tier-by-tier routing is performed to finalize the 3D design.

The study of MBFF is presented in the literature [10]. In References [11, 12], the FF clustering is performed by matching feasible regions of FFs from dynamic implied skew constraints. However, due to the high time complexity of such methods, the authors in Reference [13] proposed the FF clustering approach using clock arrival time from **linear programming (LP)** to cluster the FF. Reference [14] proposed the approach to design a low-power FIR filter using the MBFF technique. These studies are limited to a 2D IC design, and there are no existing works on applying multi-bit flip-flops to optimize the full-chip 3D design.

3 OUR METHODOLOGIES

3.1 Overview

In this section, we present our MBFC flow as illustrated in Figure 1. The MBFC supports both 2D and 3D ICs. For 3D IC design, we extend the S2D flow [3] to optimize the clock network power and design performance.

Table 1. Features Used to Construct Skew Constraint Graphs

features	descriptions
x	x location of flip-flop in design layout
y	y location of flip-flop in design layout
worst_slack_launch	worst slack of flip-flop as launch
worst_slack_capture	worst slack of flip-flop as capture
worst_slack_through	worst slack through the flip-flop
clock skew group	clock latency of sub tree

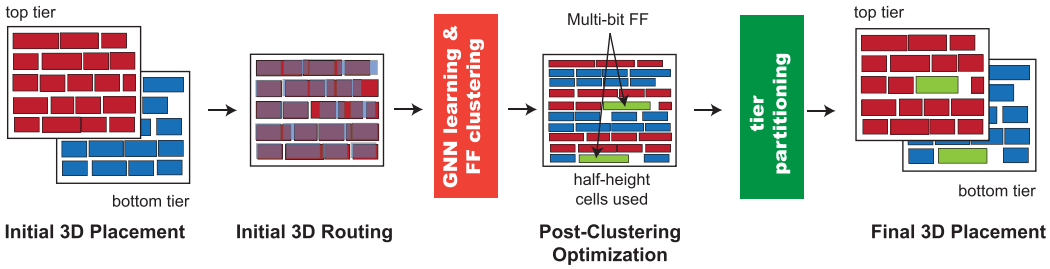


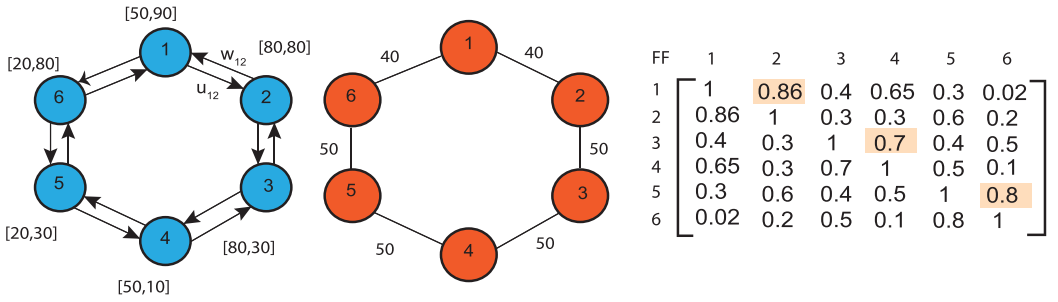
Fig. 2. Our GNN-based multi-bit FF clustering steps for 3D ICs. We use half-height cells in our post-clustering optimization step as further illustrated in Figure 6.

In the 2D design flow, we perform FF clustering after the **clock tree synthesis (CTS)** stage when the standard cells are placed and a clock tree has been constructed. The main reason to perform the FF Clustering after the clock tree synthesis is to obtain accurate clock information such as clock slack information and clock skew group defined as features, which are used in our GNN-based FF Clustering in Table 1. In the 3D design flow, we perform the FF Clustering after the tier partitioning stage. Before performing the FF Clustering in the 3D design flow, we implemented additional steps outlined in Section 3.2 to obtain accurate timing and clock information for the 3D design.

Since this is the first work to perform FF Clustering in 3D IC, we integrate the LP-based clustering approach [13] into our full chip design flow, as baseline design. This integration allows us to compare LP vs. GNN in a full-chip design environment instead of evaluating clock trees in isolation as done in Reference [13]. Thus, our MBFC flow has three main steps: timing extraction, GNN-based FF clustering, and post-clustering steps.

3.2 Initial Timing Information Extraction

In the first step, we extract the timing and clock information from the design. The required information differs between LP-based and GNN-based FF clustering based on their inputs. In 2D design, we extract the necessary information directly from the design after the CTS stage, with the estimated routing information obtained from the global routing. In 3D, the clock tree has been built in the CTS stage of S2D's Pseudo-3D [3]. However, we only obtain 3D placement after the tier partitioning stage, which only contains cell information such as FFs, clock buffers, and combinational logic cells but not routing. Therefore, we perform full-chip routing to extract the clock tree and timing information. The full-chip routing combines the top and bottom dies into a single design using a 3D Back-End-Of-Line with a double metal stack, as in the "Initial 3D Placement" and "Initial 3D routing" stages in Figure 2. As a result, both top and bottom cells are overlapped, but their pins are of different metal layers. The top cells have pins at the top metal layer, while the bottom cells



(a) SCG Graph (b) Graph Neural Network (c) Similarity Matrix

Fig. 3. GNN-based FF clustering illustration. (a) SCG, (b) graph neural network (GNN) for training, and (c) similarity matrix.

have pins at the bottom. After that, we perform the global and detailed routing, including both the clock and signal nets, with a fixed cell placement. Once the routing is completed, we extract the necessary information from the fully routed 3D design using the Cadence Innovus tool.

3.3 GNN-based FF Clustering

GNN-based FF clustering consists of three main steps: GNN Model, GNN learning, and GNN FF clustering. The goal of GNN-based FF clustering is to find the similarity between every pair of FFs under a GNN-transformed high-dimensional space. Given comprehensive features, including clock metrics and timing path information, we perform the feature ranking to select only essential features as initial features. Once we obtain the initial features for each design instance (=node), we leverage GNN to perform node representation learning to transform the initial features into high-dimensional representations by aggregating and distilling the netlist information through a message-passing process. And, last, we cluster similar pairs of FF from the similarity matrix derived from high-dimensional representations in GNN.

The key idea of using GNN to perform FF clustering is that if there exists a pair of FFs that should better be merged into a two-bit FF, then the representations between these two FFs should be more similar in the high-dimensional space (GNN-transformed) than in the low-dimensional space (manually defined).

3.4 GNN Model

In the first step of GNN-based FF clustering, we construct the **Skew Constraint Graph (SCG)** [15] from the netlist of the design benchmark. The SCG contains a node as a FF, and we add an edge if a timing path exists between two FFs. Therefore, we only extract the FFs from the netlist and omit other logical cells. An example of the SCG graph is illustrated in Figure 3(a). We assign the weight of each edge in the graph as the Manhattan distance between nodes. Therefore, the message passing process is scaled based on the different weights of the edges, which are determined by their distance. Next, we extend Reference [16] to handle the SCG graph as a graph network structure. In the conventional netlist, there is no separated subgraph. However, the SCG graph contains a subgraph where the FFs form local connections only among the small group of flip-flops. As a result, we select the largest connected subgraph as a training network.

3.4.1 Feature Selection. Initial features are essential for the best result of GNN. Therefore, we consider 13 relevant features for each FF, including location in X, Y, and Z (= tier location), clock information such as clock latency for both launch and capture, clock skew for all three

Table 2. Candidate Features for Evaluation in Our GNN Learning

no	features	descriptions
1	die loc	die location (0 = top, 1 = bot)
2	x	x location of flip-flop in design layout
3	y	y location of flip-flop in design layout
4	slack launch	worst slack of flip-flop as launch
5	slack capture	worst slack of flip-flop as capture
6	slack through	worst slack through the flip-flop
7	clock capture latency	capture clock latency
8	skew launch	worst skew of flip-flop as launch
9	skew capture	worst skew of flip-flop as capture
10	skew through	worst skew through the flip-flop
11	clock launch latency	launch clock latency
12	clock level	level of flip-flop in clock tree hierarchy
13	clock skew group	clock latency of sub tree

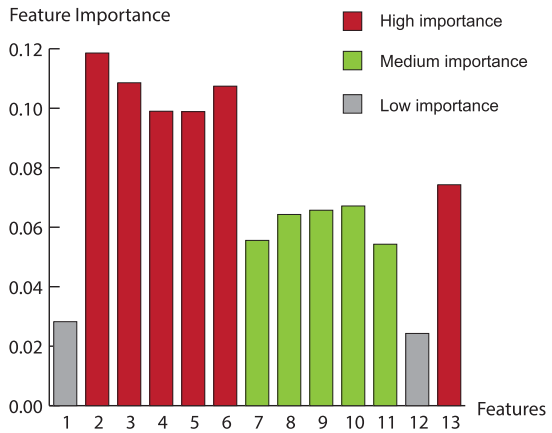


Fig. 4. Feature importance of the features shown in Table 2 using Random Forest classifier.

conditions: launch, capture, and through the FF, clock level, and clock skew group. The details of each comprehensive feature are listed in Table 2. Next, we utilize the Random forest [17] to effectively rank the feature importance of candidate features and select the best subset for GNN learning.

Specifically, we use the target variable as a pair number of FFs. A pair of merging FFs has the same pair number, starting from 1 to the maximum number of pairs possible. We obtain the pair number from the SCG graph training with all comprehensive features and perform the FF clustering. Therefore, we have a clustering pair for each FF. We train the Random forest classifier with comprehensive input features and pair numbers and then extract the feature importance as illustrated in Figure 4. The graph shows the average feature importance from all benchmarks for 13 features.

3.4.2 Initial Node Features. With feature importance in the previous section, we select particular features with feature importance more than the average. Therefore, we define the initial feature of each node (FF) in the GNN graph, as shown in Table 1. The features contain the location of FFs

(x,y) and slacks for all timing paths. All six features have high feature importance compared to other features. The first two features are the location of the FF, which contributes to the wirelength. The following three features, slack information of FFs, is timing path information that contains three values: worst timing slack with FF as launch FF, capture FF, and through FF. This detailed slack information allows us to group the FFs with similar slacks, ensuring that the additional useful clock skew required to close the timing is minimal and that the additional skew will not cause further hold or setup violations on related paths. The clock skew group's last feature is the clock latency of the subtree. A subtree in the context of a clock delivery network refers to a group of clock sinks that are connected to a common parent node in the clock tree. Sinks within the same subtree have the same clock skew group.

3.5 GNN Learning

Once we construct the GNN model with node and edge representation with initial features, we proceed into the GNN learning step, where we aggregate the neighbor information from low-dimensional space into high-dimensional space. Below, we illustrate the transformation process in detail.

3.5.1 Neighborhood Encoding. With the initial node features presented above, we perform node representation learning using GNNs. Based on the skew constraint graph shown in Figure 3(a) (more on the SCG generation is discussed in Section 3.9.1), we leverage GNNs to transform the initial features we define for each node (i.e., FF) into a high-dimensional representation by capturing neighboring information.

This aggregation process is performed iteratively. Our GNN model will act as a "graph filter" that visits every node in the design to aggregate the information of its local neighborhood. In general, GNN consists of a set of neron layers, where each layer is dedicated to performing the aggregation of a specific hop of the neighborhood. For a node $v \in V$, the representations at level k are obtained as follows:

$$h_v^k = \sigma\left(h_v^{k-1} + \theta_k \cdot \frac{1}{s_k} \sum_{u \in N_k(v)} h_u^{k-1}\right), \tag{1}$$

where σ is the sigmoid function, h_v^k denotes the representation vector of node v at level k, $N_k(v)$ denotes the neighbors sampled at k-hop that is subject to the sampling size s_k, and θ_k denotes the trainable parameters of neighborhood feature aggregation at level k, which is realized by a neural layer. Note that the concept of "level" corresponds to the concept of "hop," where h_v^0 is the initial features of node v, and $h_v^{k=K}$ is the final representation after aggregating the information within the K-hop neighborhood of v.

3.5.2 Unsupervised Loss Function. In this article, we construct a loss function that will encourage nodes connected on the constrained graph to have similar representations in the high-dimensional space while minimizing the similarity between nodes not connected on the graph. Following from Reference [18], the loss function is constructed as follows:

$$\mathcal{L}(h_v) = -\sum_{u \in N(v)} \log(\sigma(h_v^\top h_u)) - \sum_{i=1}^{M} \mathbb{E}_{n_i \sim Neg(v)} \log(\sigma(-h_v^\top h_{n_i})), \tag{2}$$

where $N(v)$ denotes the direct one-hop neighborhood of the target node v, $Neg(v)$ represents the negative sampled nodes in the perspective node v, and M represents the negative sampling size. Note that the negative sampling is achieved by performing random sampling on the nodes that are beyond the local neighborhood of the target node v, and we aim to minimize the similarity

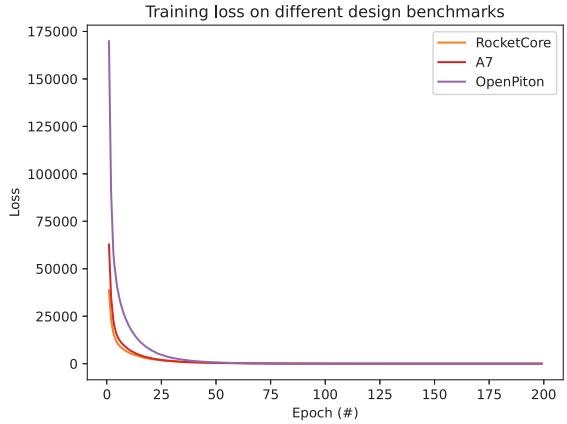


Fig. 5. Training loss on different design benchmark: RocketCore, A7, and OpenPiton.

between the negatively sampled nodes and the target node. Finally, note that Equation (2) is an unsupervised loss function, which means the proposed clustering methodology does not require a huge dataset to be pre-generated and is generalizable to any netlist as it does not assume any pre-defined netlist structure.

3.5.3 Training Parameters. We specify the number of epochs, where data of each node(=FF) aggregates to K neighbors for one time, of all design benchmarks to 200. The k value in Equation (1) is chosen to be two based on the empirical experiments [19]. The high number of k will degrade the quality of most GNNs. Therefore, the batch size is set to 256. Next, we run the gradient decent algorithm to reduce the loss on a built network in Figure 3(b) with the above setting. And, we observe that the loss value converges quickly for all designs despite the size of the SCG graph. As a result, the runtime of GNN training is short and more practical for large industrial designs, as shown in Figure 5.

3.6 GNN FF Clustering

In the last step of GNN-based FF clustering algorithm, we obtain the GNN model with high-dimensional representation. Each node contains 32 high-dimension features from feature encoding, which better represents the relationship of FF than the initial features. The last stage is the clustering stage, where a pair of FFs is merged based on the similarity of their GNN features. We utilize cosine similarity, which is given by

$$Cosine\ similarity = \frac{A \cdot B}{|A| \cdot |B|} \in [-1.0, 1.0] \quad \forall A, B \in \mathbb{R}^n. \tag{3}$$

With the Cosine similarity matrix, as shown in Figure 3(c), we sort the maximum entries and ignore the entries along the leading diagonal as they correspond to self-correlation. Next, we iterate the sorted entries and check the FF merging constraints.

We calculate the pairwise similarity matrix using the 32 embedded features derived from GNN. The entry in row i and column j of the similarity matrix indicate the similarity between node i and node j, with a value $\in (-1.0, 1.0)$. We ignore the diagonal entries, since they indicate similarity to themselves and are always equal 1.

Using the similarity matrix, we perform the GNN FF Clustering algorithm as outlined in Algorithm 1. At the start, the merging list contains all the FFs in the design. In Algorithm 1, we first pick an FF (*rand_FF*) from the merging list and find another FF (*comp_FF*) to merge with,

ALGORITHM 1: GNN FF Clustering Algorithm.

```
Result: Merge_pair : Pairs of FF to be merged
FF_info[name] ← [location, nets, type];
FF_list ← FlipFlops;
Sim_Mat ← SimilarityMatrix;
Merge_pair = ∅;
while FF_list not empty do
    rand_FF ← Random pick FFs;
    FF_list.pop(rand_FF);
    min_dist ← inf;
    max_sim ← − inf;
    merge_FF ← −1;
    for comp_FF in FF_list do
        D ← Dist(rand_FF, comp_FF);
        T_flag ← validate(rand_FF, comp_FF);
        i = name2index(rand_FF)
        j = name2index(comp_FF)
        if Sim_mat[i][j] > max_sim & T_flag & D ≤ Max_range then
            merge_FF ← comp_FF;
            max_sim ← Sim_mat[i][j];
        end
    end
    if merge_FF ≠ −1 then
        FF_pair = (rand_FF, merge_FF);
        Merge_pair.push(FF_pair);
        FF_list.pop(merge_FF);
    end
end
```

subject to the following conditions. Next, we calculate the Euclidean distance between *rand_FF* and *comp_FF*. We then verify that the two FFs are of the same type and have the same clock nets to maintain the correctness of the netlist. The FF with the highest similarity to *rand_FF* and within the maximum search range is selected to merge with *rand_FF* and is removed from the merging list. If there is no pair of FFs to merge with *rand_FF*, then we drop it and pick another FF. This process is repeated until all FFs have been checked. Once we complete the GNN-based flip-flop clustering in Algorithm 1, we merge pairs of FFs into multi-bit FFs and modify both the netlist and placement accordingly.

We use two-bit FF of four different types: D-Q FF, D-QN FF, D-Q FF with reset pin, and D-QN FF with reset pin, shown in Table 3. These provide hard constraints for merging two FFs and apply to both LP and GNN-based FF clustering to ensure correctness. In addition, we maintain the correctness of the netlist by validating if the clock and reset nets match for merging the FF pair candidates.

3.7 Post-clustering Optimization

This section presents the optimization stage for fixing the timing paths of the modified netlist with multi-bit FFs. In 3D design, we extend the approach presented in Reference [5] to perform placement legalization and CTS with both dies simultaneously in a single design. The first step is to scale cells into half their height, as shown in Figure 6(a). The order of row alignment differs

Table 3. Area and Power Comparison between Single-bit FF and Multi-bit (= 2-bit) FF of the Commercial 28-nm Technology We Use in Our Experiment

Pins		Single-bit FF		Two-bit FF	
R	QN	Area (μm^2)	Power (nW)	Per-bit Area (μm^2)	Per-bit Power (nW)
N	N	2.73	2.30	2.35	1.82
N	Y	2.75	2.01	2.35	1.99
Y	N	3.40	2.63	3.00	1.99
Y	Y	3.24	2.98	2.84	2.27

"R" denotes if flip-flop has reset pin and "QN" if output is inverted.

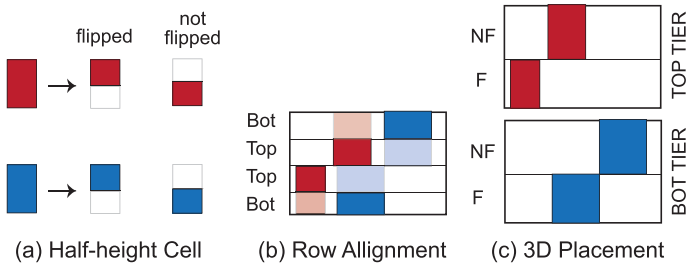


Fig. 6. Cell height management in our flow. (a) Half-height cells; (b) row alignment, timing closure, and legalization; and (c) height restoration after tier partitioning for the final 3D IC design. The reason for half-height is to use a commercial 2D IC timing closure tool for 3D IC designs.

from Reference [5] as the row orientation for both dies is identical as in Figure 6(b). Finally, the cells are shifted to match row alignments, as shown in Figure 6(b). With the above settings, we perform the incremental placement to correct the multi-bit FF location, as illustrated in Figure 2. Then, the remaining steps of the physical design flow (CTS, Signal Routing, and Timing Closure) were performed to finalize the design. To convert this mixed-die design into a 3D design, we split the dies and shifted the even-order rows before restoring the cells to their original size. Finally, the 3D placement is obtained as in Figure 2, as the cell height is restored in Figure 6(c). In 2D design, we perform post-clustering steps as mentioned above, with original technology nodes after the clock tree synthesis stage.

3.8 Handling Multiple-bit Flip-flop Clustering

In Section 3.6, we present GNN-based FF clustering of a pair of FFs into a two-bit FF. However, multiple-bit FFs exist, such as 4, 8, and 16 bits. Therefore, we provide the additional steps for our GNN-based algorithm to support multiple-bit FFs. From the similarity matrix in Figure 7(a), there are six FFs in the SCG graph, where each FF has pairwise similarity to other FFs. We highlight the pairwise element where the value exceeds the threshold (0.5). The maximum bit size of the six FFs is 4. This is because multi-bit FFs are available in sizes that are powers of two, such as 2, 4, 8, and 16. We define k as the number of bits in the multiple-bit FF. Next, we check each FF_i from the ith row and check if there are $k-1$ entries in the similarity matrix with a value greater than the threshold (0.5), where k is the maximum number of bit size of multi-bit FF that FF_i will be grouped with. After we obtain the set of FFs for each row, we generate the pairwise combinations of the given set and check if all elements in the combination meet the requirement ($\geq$0.5) in terms of their pairwise distance. If any pairwise element in the pairwise combination for each row does not meet the requirement, then we remove them from the candidate list for k-bit clustering. Moreover, if

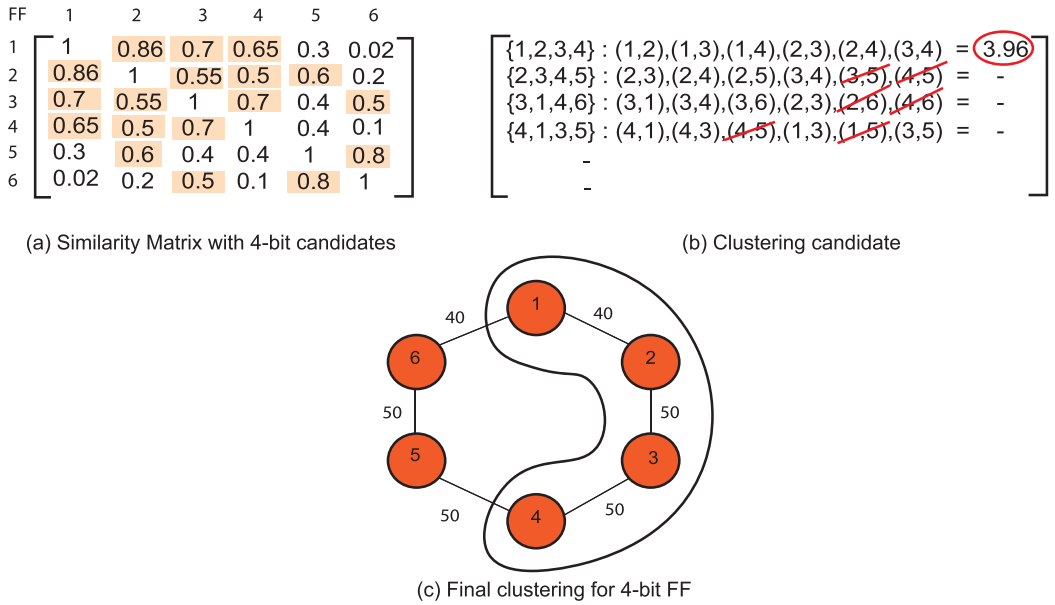


Fig. 7. Multiple-bit flip-flop clustering steps for GNN-based algorithm. (a) Cosine silimarity matrix. (b) Handling overlaping pairs. (c) Final cluster for 4-bit FF.

there exist more than two rows where the requirement is met, and they contain the same subset of FFs, then we calculate the sum pairwise to select the highest similarity sum as a cluster. As a result, our GNN-based FF clustering supports more than a 2-bit FF. However, in this article, due to the limitations of commercial **process design kit (PDK)**, we could not perform the clustering with more than 2 bits. Nevertheless, our GNN approach could perform better clustering choices from the better representation of FFs in the higher-dimensional space than the initial low-dimension feature.

3.9 LP-based FF Clustering

Since Reference [13] only performs the clock tree synthesis, we integrate and extend the FF clustering algorithm into our 3D design flow with the followings. However, we also perform the comparison with the original [13] at the CTS stage in the experimental result. The LP-based FF Clustering consists of three main steps: (1) We extract all timing path constraints from the design layout and generate the SCG. (2) We use LP formulation with constraints to solve for a clock arrival time of a FF. (3) We perform the FF clustering based on overlapped (=matching) arrival time and additional merging criteria.

3.9.1 Timing Path Constraint. Given a design layout, we extract the RC parasitic to generate timing path constraints for both setup and hold. For each path constraint, two FFs (launching and capturing FFs) are connected through a chain of combinational logic cells. With setup and hold constraints, we reformulate the differences in clock latency between launching and capturing FF in terms of skew as follows:

$$-l_{launch\_cap} \leq t_{launch} - t_{capture} \leq u_{cap\_launch}, \tag{4}$$

$$l_{launch\_cap} = t^{min}_{comp} + t^{CQ}_{launch} - t^{hold}, \tag{5}$$

$$u_{cap\_launch} = T - t^{max}_{comp} - t^{CQ}_{launch} - t^{setup}, \tag{6}$$

where u_{cap_launch} and l_{launch_cap} refer to the upper and lower bounds of the skew constraint of each path, respectively. t_{comp}^{min} and t_{comp}^{max} refer to the combinational delay of each path for hold and setup constraints, respectively. t_{launch}^{CQ} denotes the clock-to-Q delay for the launching FF, and t^{setup} and t^{hold} are the setup and hold times, respectively. With Equation (4), we calculate upper and lower bound skew constraints in the design to generate the SCG, as shown in Figure 3(a). Each node represents a FF, while the edges represent the skew constraint of a FF pair.

3.9.2 LP Formulation. Next, we utilize the LP formulation to solve the optimal arrival time of each flip-flop using the objective function in Reference [13]. The following equations show the objective function and constraints of the LP problem:

$$min \sum_{i \in V} f(x_i^{lb})^{lb} + f(x_i^{ub})^{ub}, \tag{7}$$

$$x_i^{lb} \leq x_i^{ub}, \forall i \in V, \tag{8}$$

$$-l_{ij} \leq x_i^{ub} - x_j^{lb} \leq u_{ji}, \forall (i,j) \in E, \tag{9}$$

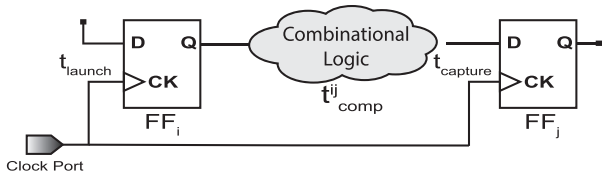
where $f(x_i^{lb})^{lb}$ and $f(x_i^{ub})^{ub}$ are two piecewise functions to maximize the range of arrival time. $[x_i^{lb}, x_i^{ub}]$ denotes a range of clock arrival time with lower bound and upper bound. l_{ij} and u_{ji} denote the lower bound and upper bound of skew constraint for path i to j, as defined in Equations (5) and (6), respectively. Equation (8) is to ensure that the lower bound is less than the upper bound. Equation (9) is the skew constraint defined at an edge of the SCG graph (Equation (4)). This objective function maximizes the bound of the arrival time range of all FFs using wrapper piecewise functions. Thus, this increases the number of pairs to be merged in the clustering stage. From the SCG graph, we need to satisfy each edge representing the arrival time constraint between a pair of FF as defined in Equation (4).

We consider the hold constraint in Equation (9) in addition to Reference [13] to avoid additional buffer insertion. Moreover, we performed an analysis to fine-tune the piecewise linear model for the maximum range of arrival time and set the number of slopes to three at 1, 2, and 3 ns with the slope equation as in Reference [9].

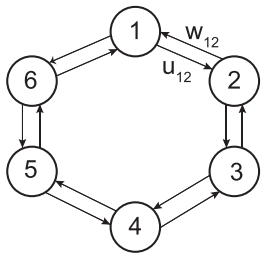
3.9.3 Clustering Constraints. Since our MBFC design flow performs full-chip routing for both clock and signal nets in a 3D metal stack, we modified the merging criteria from Reference [13] by adjusting the maximum search bound D_{max}. This allows each flip-flop to search for matching candidates, as shown in Figure 9(a). In Reference [13], the matching criteria use the overlapped arrival time range of each flip-flop in Figure 8(c), which is solved from the LP formation. Utilizing the maximum search bound enables more FFs to be merged in some designs where the timing constraints are strict, and no available pairs can be merged without violating the timing. We fine-tune the maximum search distance for the best result for a fair comparison. Other merging criteria are defined in Table 3.

3.10 Clustering before Tier Partitioning: An Alternative

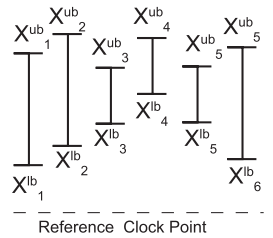
Another option to conduct FF clustering within the S2D [3] framework is to perform FF clustering before the tier partitioning stage. This overall approach is illustrated in Figure 10. This algorithm begins right after the post-placement stage, where placement and timing are optimized before constructing a clock tree (CTS). During this stage, the timing analysis is estimated without actual routing. This framework supports any post-placement FF clustering algorithm into 2D and 3D designs. In this work, we utilize the LP-based FF clustering in Section 3.9.



(a) Timing path constraint



(b) SCG Graph



(c) Clk. Arr. time bound of each FFs.

Fig. 8. First three steps of Multi-bit flip-flop Clustering. (a) Extract timing path constraints from layout. (b) Generate SCG graph. (c) Arrival Time Result of each FFs.

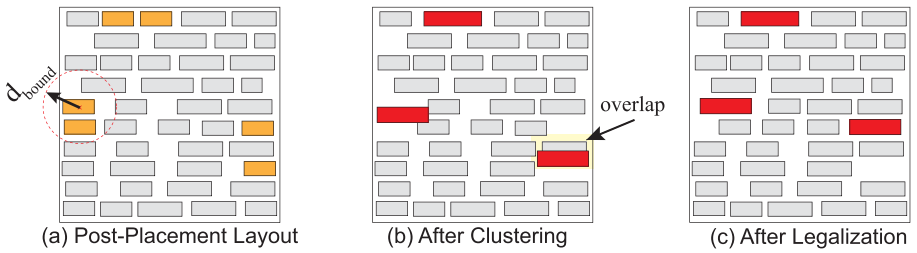


Fig. 9. Last two steps of Multi-bit flip-flop Clustering. (a) Input layout from Post-Placement optimization stage. (b) Layout after clustering step. (c) Layout after placement legalization.

The algorithm analyzes the timing and merges single-bit FFs into MBFFs. The output from the algorithm provides legalized placement with an updated netlist containing MBFFs. Next, we perform clock tree synthesis, signal routing, and timing closure stages with the default S2D flow. The cells are then resized to their original size and partitioned into two dies. Last, the inter-tier vias are inserted, and tier-by-tier routing is performed.

In **Multi-bit FF Clustering before Tier partitioning flow (MBFC-BT)**, the tier location is not assigned. Thus, the clustering only considers the design as a 2D placement in Figure 9. From Figure 9(a), the original placement of post-placement optimization is a starting point with the orange and gray rectangles as FFs and combinational cells, respectively. First, we randomly pick the FF in the layout and find the nearest-neighbor FFs to merge within a bound distance (*d_bound*) if the clock and reset net match. Suppose Close-by FFs have intersect-bound clock arrival time. They are merged and displaced at the mid-point of the pair, as illustrated in Figure 9(b). Finally, we performed incremental placement to legalize newly merged MBFF placement and minimize data-path wirelength, and the outcome is shown in Figure 9(c).

3.10.1 Iterative Clustering Consideration. We present the baseline approach of our MBFC-BT. The purpose is to consider the impact of previous flip-flop Clustering iterations. For each iteration, the timing path extraction, SCG generation, arrival time solver, FF clustering, and

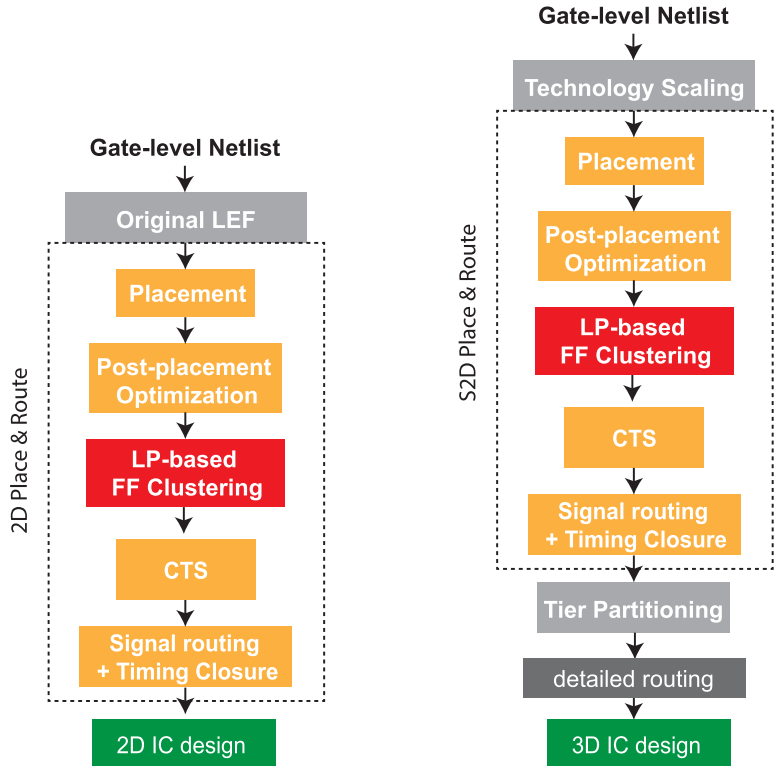


Fig. 10. Our MBFC–BT flow.

legalization are performed with additional criteria in the clustering stage. Instead of choosing all possible clustering pairs to be merged, only the best candidate pairs (N% of all possible pairs) are selected based on the amount of overlap bound of clock arrival time. In other words, all possible pairs are sorted in descending order of overlap, and only N% of pairs are selected, where N is specified by the user. The stop condition of this algorithm is when no clustering pairs are possible, and the remaining steps in Shrunk-2D [3] are performed. The experiment result is in Section 5.3.

4 IMPACT OF GNN-BASED FF CLUSTERING

4.1 Experimental Setup

In our experiments, we compare the impact of our MBFC algorithm on **Power-Performance-Area (PPA)**, clock Metrics, and full-chip timing. We utilize a commercial 28-nm PDK with six metal layers for 2D IC design and with 12 metal layers (six metals for each die) for 3D IC design (3D). The 3D design contains two dies with six metal layers, and uses F2F vias as inter-tier connections. The F2F via size, pitch, resistance, and capacitance are set to 0.5 μm, 1 μm, 1.0 Ω, and 0.05 fF, respectively. We use eight circuits (pure-logic and processor RTLs) as benchmarks to evaluate the performance of our algorithm. The details of these eight benchmarks are given in Table 4. We set the clock frequency of each benchmark circuit to the maximum clock frequency of 2D design with the **Worst Negative Slack (WNS)** within 10% of the clock period. All the 3D footprints of a specific RTL are identical in terms of footprint and placement of memory macros. The results of the ARM Cortex-A7 and Cortex-A53 designs are normalized with respect to the 2D design to

Table 4. Benchmark Statistics

Design	# Cells	# FF	purpose
VGA	34,104	17,051 (50.0%)	training
TATE	212,996	31,409 (14.7%)	cell dominated
NOVA	138,383	29,078 (21.0%)	FF dominated
JPEG	240,155	37,540 (15.6%)	large circuits
RocketCore	120,565	16,429 (13.6%)	small processor
A7	208,491	22,366 (10.7%)	industrial processor
OpenPiton [20]	186,485	53,358 (28.6%)	RISC-V processor
A53	528,488	47,527 (8.9%)	industrial processor

Table 5. The Metrics Definition for Both PPA and Clock Metrics

PPA Metrics	Description	Clock Metrics	Description
WL	Total wirelength in the design	**Clk. WL**	Total clock wirelength in the design
Pwr.	Total power consumed in the design	**FF #**	Number of total flip-flops in the design
WNS.	Worst negative slack	**Clk. Pwr.**	Total clock power (Clock Net & Clock Buffers)
TNS.	Total negative slack	**Clk. Lat**	Worst clock Latency
PDP.	Power delay product (mW*ns)	**Clk. Skew.**	Worst clock skew
EDP.	Energy delay product (mW*ns2)	**FF Pwr.**	Total sequential power

protect sensitive information due to non-disclosure agreements. We define the metric definition for PPA and Clock metric comparison in Table 5.

The two-bit FFs cells used in this article are provided from the PDK. Due to the limited access to the back-end library, we could not generate higher multi-bit FFs such as 4 or 8. Therefore, we only leverage commercial-quality two-bit FFs with a different type to experiment.

The 2D and 3D design options in this experiment are listed as follows.

- **2D**: Monolithic 2D IC with six metal layers
- **3D**: The state-of-the-art 3D design [3]
- **LP**: Two-dimensional design using the state-of-the-art clock clustering algorithm using clock arrival time [13].
- **LP_opt**: Since no previous works have implemented a final full-chip design with MBFFs, we implement the multi-bit flip-flop clustering 3D design using the linear programming (MBFC_LP) clustering approach outlined in Section 3.9.
- **2D_LP_opt**: Two-dimensional design using our linear programming (MBFC_LP) clustering approach outlined in Section 3.9.
- **2D_GNN**: Two-dimensional design using our GNN-based FF clustering algorithm (MBFC_GNN)
- **GNN (This work)**: Three-dimensional design using our GNN-based FF clustering algorithm (MBFC_GNN).

4.2 GNN-related Results

In this section, we evaluate graph learning by constructing **t-distributed stochastic neighboring embedding (t-SNE)** to visualize the FF representations from 32 dimensions to 2 dimensions. For example, in Figure 11, we demonstrate the node representation of FFs in the RocketCore design with 10 clusters based on the similarity representation from embedded dimensions. Furthermore, we observe that similar FFs form groups of clusters, each denoted by a different color. The clear separation of each cluster reflects the effectiveness of our GNN framework in translating initial features to meaningful embedded features that serve as a basis for selecting the pairs to merge into MBFFs.

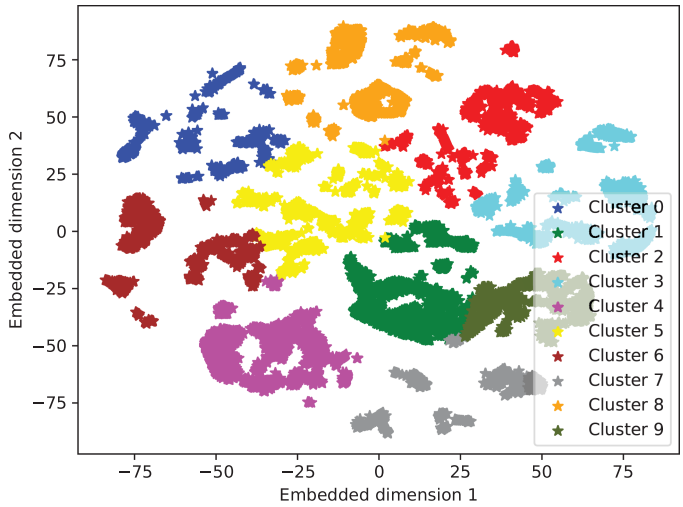


Fig. 11. t-SNE visualizations of the learned node representations from GNN for RocketCore design. Each dot represents a FF. FF color is assigned based on the similarity of its representation.

Table 6. 2D IC Clock Metrics Comparison

Pure-logic Design										
Metrics		TATE (1.8 GHz)			NOVA (500 MHz)			JPEG (1.55 GHz)		
		2D	LP	2D_GNN	2D	LP	2D_GNN	2D	LP	2D_GNN
Clk. WL.	(mm)	118	86	86	104	83	95	139	97	99
FF	#	31K	17K	17K	29K	19K	25K	37K	20K	20K
Clk. Pwr.	(mW)	46.7	33.7	33.6	8.5	6.5	7.4	49.2	34.7	35.4
FF Pwr.	(mW)	192	160	161	35.1	30.5	32.9	205.1	173	174
Clk. Lat.	(ps)	334	352	309	285	350	326	420	438	405
Clk. Skew	(ps)	44	31	25	191	173	191	118	119	87
WNS.	(ps)	8	8	8	0	0	10	44	32	46
Processor Design										
Options		RocketCore (1 GHz)			A7*			OpenPiton (450 MHz)		
		2D	LP	2D_GNN	2D	LP	2D_GNN	2D	LP	2D_GNN
Clk. WL	(mm)	66	49	49	1	0.76	0.76	390	367	372
FF	#	16K	9K	9K	1	0.55	0.56	53K	45K	48K
Clk. Pwr.	(mW)	15.6	11.7	11.9	1	0.78	0.76	7.4	7	7.3
FF Pwr.	(mW)	56.8	48.3	48.3	1	0.81	0.82	26.9	24.6	25.8
Clk. Lat.	(ps)	322	346	338	1	0.99	1.00	469	572	467
Clk. Skew	(ps)	84	125	104	1	1.22	1.33	374	369	331
WNS.	(ps)	81	94	84	1	0.88	0.98	171	276	159

(1) 2D: Cadence Innovus using 1-bit FFs; (2) LP: State-of-the-art LP-based FF clustering [13]; and (3) GNN: our GNN-based FF clustering. The highlighted entries indicate the best result among all designs.

4.3 Comparison with Existing Work

We validate our GNN-based FF clustering approach with state-of-the-art FF clustering (LP) [13] on six design benchmarks. Furthermore, since the state-of-the-art design evaluates the result after the CTS stage, we compare the clock metrics such as clock wirelength, clock-related power, and

Table 7. Clock Metrics Comparison

Pure-logic Design													
Metrics		TATE (1.8 GHz)				NOVA (500 MHz)				JPEG (1.55 GHz)			
		2D	3D	LP_opt	GNN	2D	3D	LP_opt	GNN	2D	3D	LP_opt	GNN
Clk. WL.	(mm)	118	104	85	78	103	90	76	69	141	120	96	86
FF	#	31K	31K	18K	16K	29K	29K	17K	16K	37K	37K	21K	19K
Clk Pwr.	(mW)	47.07	44.23	34.45	31.87	8.68	7.96	6.33	5.93	50.54	47.96	38.39	33.56
FF Pwr.	(mW)	192.8	190	161	157	35.33	35.03	29.9	29.19	206	203	175.4	171.4
Clk. Lat.	(ps)	331	279	303	287	286	267	308	240	414	369	412	385
Clk. Skew	(ps)	48	47	79	65	193	173	195	188	129	127	187	123
WNS.	(ps)	22	60	46	39	27	86	15	23	48	89	70	51
Processor Design													
Metrics		RocketCore (1 GHz)				A7*				OpenPiton (450 MHz)			
Clk. WL	(mm)	66	56	48	44	1	0.82	0.73	0.68	390	434	362	344
FF	#	16K	16K	10K	9K	1	1.00	0.58	0.55	53K	53K	49K	47K
Clk. Pwr	(mW)	16.17	14.97	12.39	11.29	1	0.87	0.76	0.75	7.7	7.4	6.95	6.94
FF Pwr.	(mW)	57.21	56.32	48.06	47.53	1	0.98	0.83	0.82	27.26	26.04	24.66	24.31
Clk. Lat.	(ps)	350	290	314	306	1	0.90	0.85	0.80	454	509	449	447
Clk. Skew	(ps)	116	115	167	126	1	1.66	1.10	1.29	373	225	258	247
WNS.	(ps)	78	92	89	84	1	1.08	0.50	0.41	473	88	17	19

(1) 2D: commercial 2D IC with 1-bit FF (Cadence Innovus); (2) 3D: 3D IC with 1-bit FF [3]; (3) LP_opt: 3D IC with 2-bit FF using LP approach in Section 3.9; and (4) GNN: 3D IC with 2-bit FF using GNN (this work). The highlighted entries indicate the best result among 3D designs.

clock delay between LP [13] and our GNN-based FF clustering approach. From Table 6, it can be seen that both LP- and GNN-based approaches improve clock wirelength, clock net power, and FF power over the single-bit FF design (2D). Second, our GNN achieves comparable clock wirelength, clock power, and FF power to LP with an average saving of 18.2%, 17.3%, and 13.8% to 2D design, respectively.

The benefit of GNN-based FF clustering is that it provides better clock latency and skew in most designs except A7 and NOVA designs. The main reason is that the GNN approach groups a pair of FF based on similarity, considering more design features such as slack parameters and other features such as location and clock latency. As a result, GNN provides better clock tree performance with comparable power to the state-of-the-art approach.

4.4 Clock Metrics Comparison

In this experiment, we validate our GNN on all six design benchmarks by comparing clock metrics on all design options. The LP_opt implementation is based on Section 3.9. From Table 7, we first observe that multi-bit FFs improve clock wirelength, clock net power, and FF power for both LP_opt and GNN-based approaches.

4.4.1 Clock Wirelength and FF Cells. We observe that the 3D design obtains slightly better wirelength than the 2D design from a smaller footprint and shorter interconnect. Furthermore, LP_opt and GNN further improve the clock wirelength by merging single-bit FFs, which results in fewer clock pins in the clock tree network. However, the GNN obtained the best clock wirelength in all six design benchmarks due to more clustering pairs. This is because the GNN FF clustering algorithm is based on similarity, which clusters FFs that are farther apart, not just nearest-neighboring FFs.

4.4.2 Clock Power. The clock and FF power in 3D are slightly improved from the 2D design due to the shorter clock wirelength. However, the average clock power and FF power saving from 2D design are 7.31% and 2%, respectively. The MBFF design for both LP_opt and GNN further improves clock power and FF power from shorter clock wirelength and FF power saving from

Table 8. Full-chip PPA Comparison

Metrics	TATE (1.8 GHz)				NOVA (500 MHz)				JPEG (1.55 GHz)			
Pure-Logic Design	**2D**	**3D**	**LP_opt**	**GNN**	**2D**	**3D**	**LP_opt**	**GNN**	**2D**	**3D**	**LP_opt**	**GNN**
eff. freq (MHz)	1737	1611	1662	1682	493	479	496	494	1443	1362	1398	1436
# cells	211K	210K	197K	195K	140K	139K	129K	127K	258K	252K	245K	242K
WL. (m)	2.24	1.82	2.07	2.04	2.26	1.76	1.99	1.98	2.63	2.23	2.46	2.47
Pwr. (mW)	349.1	337.0	307.0	296.0	60.5	57.9	52.0	50.6	541.0	497.0	506.0	483.0
WNS. (ps)	20	65	46	39	27	86	15	23	48	89	70	51
TNS. (ns)	0.42	58	9.97	5.6	0.4	33	0.08	1	27	108	53	40
PDP. (mW*ns)	200	209	184	175	122	120	104	102	375	364	361	336
Processor Design	**RocketCore (1 GHz)**				**A7***				**OpenPiton (450 MHz)**			
eff. freq (MHz)	928	916	918	923	1.00	0.99	1.06	1.07	371	433	447	446
# cells	122K	122K	116K	114K	1.00	1.00	0.96	0.96	195K	194K	190K	188K
WL. (m)	2.04	1.46	1.7	1.68	1.00	0.71	0.77	0.77	6.71	5.2	5..47	5.42
Pwr. (mW)	155.3	143.0	138.2	134.2	1.0	0.89	0.86	0.9	97.3	87.3	84.4	84.1
WNS. (ps)	78	92	89	84	1.00	1.08	0.50	0.41	473	88	17	19
TNS. (ns)	19	50.7	22	22	1.00	9.44	0.60	0.25	299	0.66	0.22	0.29
PDP. (mW*ns)	167	156	150	145	1	0.9	0.8	0.8	262	201	189	188

(1) 2D: commercial 2D IC with 1-bit FF (Cadence Innovus); (2) 3D: 3D IC with 1-bit FF [3]; (3) LP_opt: 3D IC with 2-bit FF using LP in Section 3.9; and (4) GNN: 3D IC with 2-bit FF using GNN (this work). The highlighted entries indicate the best result among 3D designs.

smaller internal power of MBFF. The GNN obtain 10.10% better clock net power and 5.20% lower FF power than LP_opt from more clustering pairs.

4.4.3 Clock Performance. The performance parameters contain clock latency, skew, and the worst negative slack. We first observe that the 3D design obtains better clock latency and skew than the 2D design. The clock latency in 3D design is smaller than in 2D, because the clock wirelength is shorter than in 2D design. However, the smaller skew in 3D design is due to ignoring the 3D overhead, which results in worse WNS than the 2D design. In MBFF designs, we observe that the clock latency and clock skew increases from 3D design. The main reason is that the MBFF design considers 3D overhead by performing placement and routing with a 3D metal stack by extending the Snap-3D flow [5]. The main difference between the LP_opt and GNN approach is that our GNN achieves, on average, 7.2% improvement in clock latency and 6.3% improvement in clock skew compared to the LP_opt approach, while maintaining comparable worst negative slack. In conclusion, GNN provides better clock tree performance and power efficiency compared to the LP_opt approach.

4.5 Full-chip PPA Comparisons

From Table 8, we analyze the impact of our GNN algorithm on PPA.

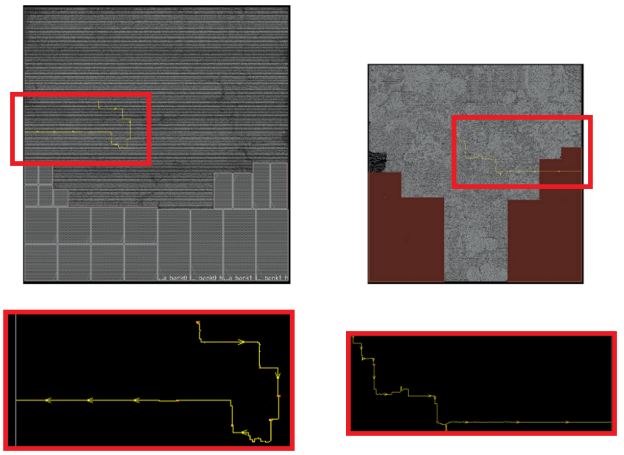
4.5.1 Total Wirelength. We first observe that 3D design has better wirelength than 2D design due to its small footprint. However, the MBFF designs obtain longer wirelength than the 3D design. The main reason is that once a pair of FFs is merged into a two-bit FF, the data path wirelength becomes longer due to less flexibility in placement. As a result, the GNN obtains 10.65% longer wirelength than 3D design but shorter than 2D design.

4.5.2 Total Power. We observe that MBFF designs obtain better power than 2D and 3D designs. The GNN obtains 7% better power than the 3D design. Additionally, the GNN achieves 2.4% better power than LP_opt.

Table 9. Full-chip Timing Comparison of A7 between Commercial 2D IC (2D) and 3D IC with 2-bit FF using GNN

Parameter		2D	GNN	Δ(%)
Path type		Ext Delay		—
Status		Violated	Met	—
External Delay	no unit	1		—
Launch Latency	ns	1	0.65	34.7%
Cell delay	ns	1	0.94	6.5%
Wire delay	ns	1	0.85	14.5%
Total Delay	ns	1	0.91	8.8%
Slack	ns	1	−0.02	102.1%

The highlighted entries indicate the best result among all designs.



(a) 2D Critical path (b) 3D_GNN Critical path

Fig. 12. Critical path layout comparison of A7 design. (a) 2D design. (b) 3D design with our GNN-based FF clustering algorithm.

4.5.3 Performance. The performance metrics include the WNS, **total negative slack (TNS)**, and effective frequency. The 3D design performance degrades from 2D design with larger WNS and TNS, because it ignores the 3D overhead and worsens timing. However, the GNN design obtains a 4.18% higher effective frequency than the 3D design. The GNN has comparable WNS with LP_opt but obtains better total negative slack.

4.6 Full-chip Timing Comparison

We compare the full-chip critical path between 2D and our GNN design in Table 9 for the A7 design. We select the worst critical path in 2D design as the timing path. The path type is an external delay output, where the path starts from the FF to the combinational delay and output port. The timing constraint for this path is specified by the external delay of the output port. In 2D, the timing is not met with negative slack, because the launch latency in 2D is 34.7% more than that in GNN, which significantly contributes to the timing path. Moreover, the total delay, consisting of wire delay and cell delay, is 8% higher than that of GNN. GNN provides a better timing path with smaller clock latency and total delay. From the Figure 12, we observe that the wirelength of the critical path for

Table 10. Runtime Breakdown of MBFC_GNN in Minutes

Stage (min)	NOVA	RocketCore	A7	OpenPiton
Extraction	9	9	4	10
SCG	5	6	1	1
GNN Training	4	2	3	3
Clustering	1	1	1	5
Legalization	5	4	7	15
Total	24	22	16	34

Table 11. Full-chip PPA Comparison between 2D and 2D_GNN

Pure-logic Design						
Metrics	**TATE (1.8 GHz)**			**JPEG (1.55 GHz)**		
	2D	**2D_LP_opt**	**2D_GNN**	**2D**	**2D_LP_opt**	**2D_GNN**
eff. freq (MHz)	1,739	1,769	1,779	1,439	1,459	1,459
Cells #	211K	195K	195K	258K	242K	242K
WL. (m)	2.24	2.34	2.34	2.63	2.57	2.56
Pwr. (mW)	349	307	308	541	495	495
WNS. (ps)	20	10	7	50	40	40
PDP. (mW*ns)	201	173	173	376	340	340
Processor Design						
Metrics	**RocketCore (1 GHz)**			**A7***		
eff. freq (MHz)	928	925	929	1.00	1.01	1.05
# cells	122K	115K	114K	1.00	0.95	0.97
WL. (m)	2.04	2.01	2.01	1.00	0.96	0.96
Pwr. (mW)	155	142	142.3	1.00	0.91	0.92
WNS. (ps)	78	80	77	1.00	0.86	0.50
PDP. (mW*ns)	167	153	153	1.00	0.90	0.86

The highlighted entries indicate the best result among all designs.

both 2D and GNN are comparable. Therefore, the most important metric that affects the timing is the clock latency.

4.7 Runtime Results

From Table 10, our GNN-based FF clustering algorithm only adds 16 – 34 min, which is insignificant compared to the overall runtime, which can be more than 12 h for commercial designs such as A7. The GNN training step utilizes a small amount of the overall runtime of the MBFC flow. Moreover, the training time does not differ from small to large designs. The extraction and legalization steps differ based on design size. As a result, the MBFC design flow reduces the effort required for parameter tuning while achieving substantial improvements in results.

4.8 Multi-bit FF Comparison on 2D

In this section, we analyze the impact of our MBFC with the GNN approach on 2D design. The Table 11 compares the PPA metrics of the 2D_GNN design with those of the 2D design and the 2D_LP_opt design. We observe that the effective frequency of the 2D_GNN is slightly higher or considered comparable to the 2D_LP_opt and 2D design. The number of cell counts in the 2D_GNN is comparable to the 2D_LP_opt, but is fewer than the 2D design from a MBFF. The wirelength in

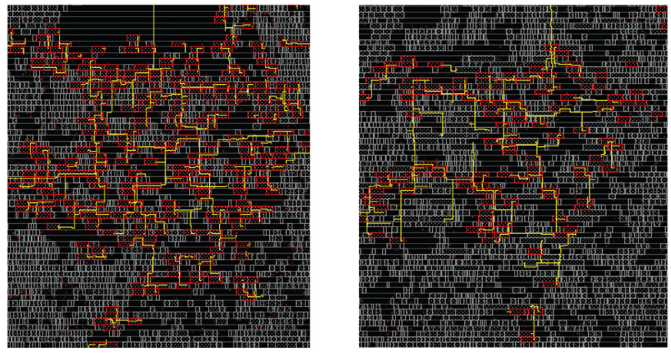


Fig. 13. Clock tree comparison of TATE circuit (zoom-in of the top tier shown). (a) S2D [3] clock tree using 1-bit FFs. (b) Our 2-bit FF-based clock tree. The red rectangle represents FF and the yellow line represents clock nets. Clock wires are shorter in the 2-bit case.

all designs is considered comparable, except in A7. The power in the 2D_GNN and 2D_LP_opt is considered comparable. However, the 2D_GNN obtains better performance than the 2D_LP_opt due to better WNS and effective frequency. Figure 13 illustrates that the MBFC design provides a shorter wirelength compared with the standard single-bit FF 2D design. As a result, our 2D_GNN provides better performance than 2D_LP_opt in addition to power saving from the MBFC over the 2D design.

5 IMPACT OF CLUSTERING BEFORE TIER PARTITIONING STAGE

This section analyzes the impact of our MBFF clustering before tier partitioning flow by comparing essential metrics with the state-of-the-art 3D design (S2D) and MBFF clustering after tier partitioning flow.

In this experiment, we compare the impact of our MBFC with the state-of-the-art 3D design (Shrunk-2D [3]). The technology spec used in this experiment is mentioned in Section 4.1. Our MBFC algorithm has the following two versions:

- **MBFC-BT:** multi-bit flip-flop clustering before tier partitioning using LP-based FF clustering in Section 3.10.
- **MBFC-AT:** multi-bit flip-flop clustering after tier partitioning using GNN-based FF clustering.

Using the LP approach, we perform the FF clustering before the tier partitioning stage. Since the LP-based FF clustering does not require accurate clock information as GNN does, we can perform the FF clustering after the placement stage before the clock tree synthesis, as in Figure 10 in S2D stage.

5.1 Full-chip PPA Comparison between before and after Tier Partitioning

In pure-logic circuits from Table 12, we observe that MBFC-BT achieves a slightly degraded and comparable wirelength compared with the S2D design. However, the MBFC-AT fixes the wirelength degradation issue and provides up to 2.78% shorter wirelength than the S2D design. Also, it reduces the total power up to 19.98% compared to the S2D design. In addition, the total power in both MBFC-BT and MBFC-AT is comparable, representing a similar number of clustering pairs among the two designs. Likewise, the MBFC designs achieved better performance, up to 75.17% in the worst negative slack. The MBFC-AT provides better performance over MBFC-BT due to no cell

Table 12. Full-Chip PPA Metrics Comparison between S2D [3] and Our Multi-flip-flop Clustering Algorithm

Pure-logic Design												
Design	**VGA @ 1.5 GHz**			**TATE (1.8 GHz)**			**NOVA (500 MHz)**			**JPEG (1.55 GHz)**		
Parameter	**3D**	**BT**	**AT**	**3D**	**BT**	**AT**	**3D**	**BT**	**AT**	**3D**	**BT**	**AT**
WL. (m)	0.91	0.89	0.912	1.89	1.885	1.865	1.85	1.86	1.838	2.38	2.34	2.314
Pwr. (mW)	30.98	24.99	24.79	337.4	293.6	292.5	58.77	50.79	50.09	499.8	465.96	471.77
WNS. (ps)	44	54	40	81	61	47	200	80	50	85	79	75
TNS. (ns)	1.05	3.58	0.838	66.62	57.98	12.92	184.9	38.75	5.334	137.7	109.64	100.47
PDP. (mW*ns)	22.02	18.01	17.53	214.7	180.9	176.2	129.3	105.6	102.7	365.1	337.38	339.66
EDP. (mW*ns2)	15.65	12.98	12.39	136.6	111.4	106.1	284.4	219.7	210.4	266.7	244.27	244.54

Processor Design												
Design	**RocketCore (1 GHz)**				**A7***				**A53***			
Parameter	**3D**	**BT**	**AT**	**Δ**	**3D**	**BT**	**AT**	**Δ**	**3D**	**BT**	**AT**	**Δ**
WL. (m)	1.53	1.57	1.53	2.7	1.00	1.03	0.98	7.34	1.00	1.55	1.11	28.27
Pwr. (mW)	143.8	131.5	130.5	0.8	1.00	0.94	0.91	−2.67	1.00	1.17	0.95	18.59
WNS. (ps)	116	100	96	14.8	1.00	0.67	0.82	−80.70	1.00	3.61	2.00	44.66
TNS. (ns)	105.1	70.8	39.3	44.4	1.00	0.54	0.22	23.91	1.00	2.65	0.39	85.22
PDP. (mW*ns)	160.5	145.4	142.2	2.2	1.00	0.88	0.88	−13.48	1.00	1.62	1.09	32.67
EDP. (mW*ns2)	179.2	160.8	155.1	3.6	1.00	0.82	0.85	−25.43	1.00	2.25	1.25	44.32

Note that BT and AT represents MBFC-BT and MBFC-AT, respectively. The Δ is the percentage difference between MBFC-AT and MBFC-BT. The highlighted entries indicate the best result among all designs.

displacement, a more accurate 3D clock tree, and a better clustering choice. As a result, The MBFC designs give 20.59% improvement in **Power-Delay Product (PDP)** and 26.02% improvement in **Energy-Delay Product (EDP)** compared to the baseline S2D design. In the processor designs, we observe degraded wirelength in MBFC-BT over the S2D design due to cell displacement from memory macro placement blockage. However, the MBFC-AT provides better wirelength in Rocketcore and A7 but reduces massive wirelength degradation in A53 within 11% compared to the S2D design. As a result, the MBFC-BT design improves the total power from the S2D design except in A53, which degraded significantly due to longer wirelength. Nevertheless, the MBFC-AT design improves total power over S2D design up to 9.23%. In addition, the MBFC-AT achieves better worst negative slack in all processor designs except A53. However, with better power saving and overall performance, the MBFC-AT design achieves up to 11.97% in PDP and 15.28% in EDP.

5.2 Clock Metrics Comparisons between before and after Tier Partitioning

Here we present the impact of MBFC in clock metrics for both MBFC-BT and MBFC-AT by comparing them with S2D [3].

5.2.1 Clock Wirelength Comparison. In the Pure-logic designs from Table 13, we observed that both our MBFC-BT and MBFC-AT achieve better clock wirelength than the S2D design for all design benchmarks. The MBFC-BT improves the clock wirelength from S2D up to 25.08%. The MBFC-AT further improves the clock wirelength from MBFC-BT up to 20.50%. The shorter clock wirelength in MBFC-BT is by merging single-bit FFs such that the number of FFs is reduced, as illustrated in Figure 13. Moreover, with fewer clock pins to access, the clock wirelength becomes shorter. MBFC-AT further improves clock wirelength due to 3D placement legalization and no displacement.

In processor design from Table 13, we also observe the shorter clock wirelength in MBFC-BT design compared to S2D design with up to 64.54% improvement. However, the MBFC-AT provides a comparable clock wirelength compared with the MBFC-BT design except for the Rocketcore design, where the clock wirelength is better by 8.93%.

Table 13. Clock Metrics Comparison between 3D [3] and Our Multi-flip-flop Clustering Algorithm

Pure-logic Design													
Options		VGA (1.5 GHz)			TATE (1.8 GHz)			NOVA (500 MHz)			JPEG (1.55 GHz)		
		3D	BT	AT	3D	BT	AT	3D	BT	AT	3D	BT	AT
Clk. WL	(mm)	116	97	86	109	87	69	97	78	71	135	101	92
Clk. Pwr.	(mW)	7.9	7.6	7.8	44.5	32.8	31.9	8.0	6.1	6.0	48.2	34.7	34.8
FF Pwr.	(mW)	18.9	13.0	12.7	190.3	157.6	158.7	35.8	29.5	29.3	203.9	171.2	172.8
Clk. Lat.	(ps)	266.6	227.9	220.3	279.7	255.5	233.2	268.6	258.8	252.8	369.6	351.1	327
Clk. Skew	(ps)	54.2	42.9	79.1	60.5	50.7	51.7	175.8	151.7	171.3	127.8	113.2	112.4

Processor Design													
Options		RocketCore (1 GHz)				A7*				A53*			
		3D	BT	AT	Δ	3D	BT	AT	Δ	3D	BT	AT	Δ
Clk. WL	(mm)	62	48	43	8.9%	1	0.71	0.73	−2.8%	1	0.35	0.69	-93.4%
Clk. Pwr.	(mW)	14.9	11.21	11.22	−0.1%	1	0.82	0.82	0.1%	1	0.90	0.70	22.7%
FF Pwr.	(mW)	56.3	46.7	47.3	−1.2%	1	0.83	0.85	−2.1%	1	0.92	0.37	59.5%
Clk. Lat.	(ps)	303.1	308.7	275.6	10.7%	1	1.04	0.89	14.8%	1	1.10	1.06	4.2%
Clk. Skew	(ps)	137.3	136	108.6	20.1%	1	0.97	0.73	25.3%	1	0.89	0.65	26.7%

Note that BT and AT represents MBFC-BT and MBFC-AT, respectively. The Δ is the percentage difference between MBFC-AT and MBFC-BT. Note that the highlighted entries indicate the best result among all designs.

5.2.2 Clock Power Comparison. In this section, we analyze the impact of both MBFC-BT and MBFC-AT on the clock power by comparing them with S2D [3]. We analyze two metrics: clock power and FF power. The clock power determines the clock delivery network, including clock buffer power and net power. Another metric indicates the power of all FFs in the design, including internal power, switching power, and leakage power. In the pure-logic designs from Table 13, we observe better FF power saving in both MBFC-BT and MBFC-AT over S2D design up to 32.80% due to per-bit power saving in MBFF. The FF power between MBFC-BT and MBFC-AT is comparable, representing a similar amount of MBFFs clustered in the design. In cell-dominated circuits such as AES_128, TATE, and JPEG, both MBFC-BT and MBFC-AT achieve up to 28.36% clock power savings. Moreover, the savings are comparable between MBFC-BT and MBFC-AT. In wire-dominated circuits such as VGA, the clock power saving in MBFC designs are comparable to S2D design. In the processor designs from Table 13, we observe similar savings in clock power and FF power as in pure-logic design. However, the FF power in the MBFC design improves significantly over the S2D design up to 62.92% in large processor A53, while the clock power reduces up to 30.47%.

5.2.3 Clock Performance Comparison. This section analyzes the two clock performance metrics: clock latency and clock skew. In the pure-logic designs from Table 13, we observe an improvement in clock latency in MBFC-BT over S2D in most designs up to 17.37%. Similarly, the clock skew is reduced to 20.85% better than that of S2D in most designs. The main reason is that there is a fewer displacement impact in the Tier partitioning stage from reduced clock MIV and merged FFs. In MBFC-AT, the clock metrics are further improved from 3D clock tree synthesis and no displacement. In the processor designs, we observe a similar saving with up to 11.48% improvement in clock latency and 34.55% in clock skew. Additionally, we observe degraded clock latency in MBFC-BT over S2D design in larger processor designs such as A7 and A53 due to large displacement from the memory macro area. However, the MBFC-AT resolves that issue and provides better clock performance metrics.

5.3 Impact of Number of Iterations

We evaluate the performance of MBFC-BT by comparing the essential metrics with the number of updates on our LP-based FF Clustering algorithm. In the original MBFC-BT flow, we cluster all possible FFs at once. However, merging pairs of the FF can affect the arrival time of other FFs that are yet to be merged. Therefore, we update the arrival times after clustering N pairs of

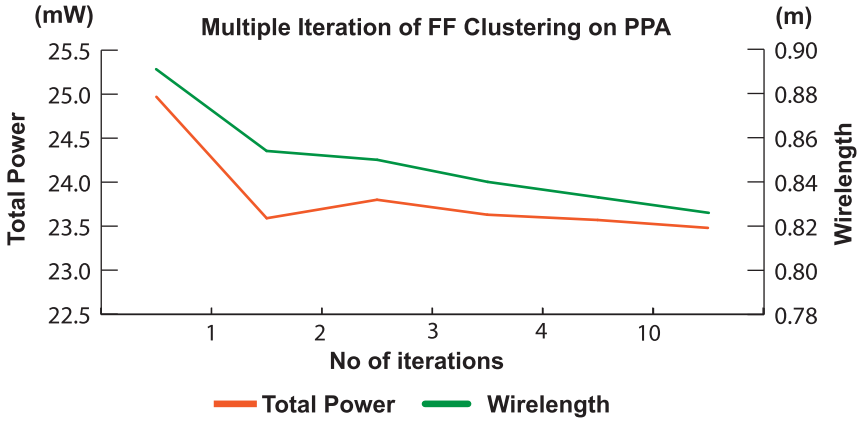


Fig. 14. Total Power and Wire length under various number of iterations on our algorithms for VGA Benchmark. The remaining designs are omitted due to space limits.

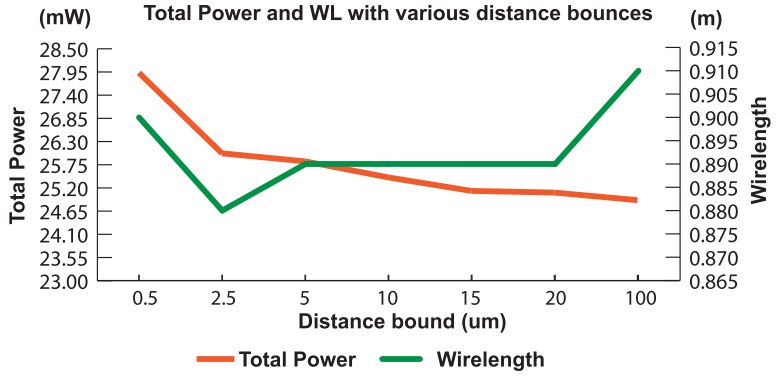


Fig. 15. PPA of VGA circuit under various distance bounds.

FFs by merging the FFs. Next, we perform full-chip routing to extract new timing information before clustering other N pairs of FFs. We repeat these steps until no possible pair of the FF can be merged. We set the maximum clustering pair (N_Max) to control the number of updates. Each update's candidate clustering pairs are sorted based on their Euclidean distance. Up to N_Max, pairs with minimum distance are chosen while the rest are ignored. After each iteration, the timing constraint will be updated with newly clustered MBFFs. We chose the VGA circuit to perform the comparison, since the proportion of net power is the most significant compared with other benchmarks to highlight the impact of iterative FF clustering. In Figure 14, we perform our MBFC-BT with various numbers of update iterations from a single iteration to ten iterations to observe the impact on PPA. We compare the final design's main metrics (the total power and wirelength). We observe that the higher the iteration, the better the wirelength, contributing to better power saving. However, the differences in PPA are minor and become saturated when the number of iterations increase. As a result, we utilize a single iteration of our MBFC-BT design.

5.4 Impact of Distance Bound

We analyzed the impact of distance bound on the FF clustering stage. From Figure 15, we observe that as the distance bound increases, the number of FF merging pairs increases and starts

converging to the maximum number of pairs. To choose the optimum distance bound, we measure each clustering result by calculating the expected value of overlapped bound. Any clustering result with the highest expected value reflects good clustering quality, since a high amount of overlap bound provides more margins for skew utilization during clock tree synthesis.

6 CONCLUSION

In this work, we proposed a GNN-based flip-flop clustering algorithm to optimize the clock power and performance in 3D ICs. We leveraged the graph neural networks on skew-constraint-graph to capture a better representation of similarities in flip-flops. The initial features, representing essential flip-flop information, are selected from all candidate features. Furthermore, by mapping embedded dimensions into a similarity matrix, we effectively found similar pairs of single-bit flip-flops and merged them into two-bit flip-flops. We extend the state-of-the-art 3D design flow with the ability to perform flip-flop clustering to improve QoR metrics. Moreover, our framework supports both 2D and 3D design flow with practical runtime for large industrial designs. The multi-bit flip-flop design for both 2D and 3D outperforms the original 2D and 3D designs with lower power and better performance.

ACKNOWLEDGEMENT

This research is funded by the Packaging Research Center at Georgia Institute of Technology and Meta.

REFERENCES

[1] Heechun Park et al. 2020. Pseudo-3D approaches for commercial-grade RTL-to-GDS tool flow targeting monolithic 3D ICs. In *Proceedings of the International Symposium on Physical Design*. 47–54.

[2] D. Duarte, V. Narayanan, and M. J. Irwin. 2002. Impact of technology scaling in the clock system power. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Paradigms for VLSI Systems Design (ISVLSI'02)*. 59–64. DOI: http://dx.doi.org/10.1109/ISVLSI.2002.1016875

[3] S. Panth et al. 2017. Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36, 10 (2017), 1716–1724. DOI: http://dx.doi.org/10.1109/TCAD.2017.2648839

[4] D. E. Shim et al. 2019. Tier partitioning and flip-flop relocation methods for clock trees in monolithic 3D ICs. In *Proceedings of the IEEE International Symposium on Low Power Electronics and Design (ISLPED'19)*, Lausanne, 1–6 DOI: http://dx.doi.org/10.1109/ISLPED.2019.8824801

[5] Pruek Vanna-Iampikul et al. 2021. Snap-3D: A constrained placement-driven physical design methodology for face-to-face-bonded 3D ICs. In *Proceedings of the International Symposium on Physical Design*. 39–46. DOI: https://doi.org/10.1145/3439706.3447049

[6] Vojin G. Oklobdzija, Vladimir M. Stojanovic, Dejan M. Markovic, and Nikola M. Nedovic. 2003. *Digital System Clocking: High-Performance and Low-Power Aspects* (1st ed.). Wiley-IEEE Press, New York, NY.

[7] V. Stojanovic and V. G. Oklobdzija. 1999. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE J. Solid-State Circ.* 34, 4 (April 1999), 536–548. DOI: http://dx.doi.org/10.1109/4.753687

[8] Mark Po-Hung Lin et al. 2011. Recent research in clock power saving with multi-bit flip-flops. In *Proceedings of the IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS'11)*. IEEE, 1–4. DOI: http://dx.doi.org/10.1109/MWSCAS.2011.6026538

[9] Rickard Ewetz et al. 2017. Clock tree construction based on arrival time constraints. In *Proceedings of the ACM on International Symposium on Physical Design*. 67–74. DOI: https://doi.org/10.1145/3036669.3036671

[10] S. Rekha, K. R. Nataraj, K. R. Rekha, and S. Mallikarjunaswamy. 2021. Comprehensive review of optimal utilization of clock and power resources in multi bit flip flop techniques. *Ind. J. Sci. Technol.* 14, 44 (2021), 3270–3279. DOI: https://doi.org/10.17485/IJST/v14i44.1790

[11] Iris Hui-Ru Jiang et al. 2012. INTEGRA: Fast multibit flip-flop clustering for clock power saving. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 31, 2 (2012), 192–204. DOI: http://dx.doi.org/10.1109/TCAD.2011.2177459

[12] Seitanidis et al. 2019. Timing-driven and placement-aware multibit register composition. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 38, 8 (2019), 1501–1514. DOI: http://dx.doi.org/10.1109/TCAD.2018.2852740

[13] Chuan Yean Tan et al. 2018. Clustering of flip-flops for useful-skew clock tree synthesis. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC'18)*. IEEE, 507–512.

[14] Lamjed Touil, Abdelaziz Hamdi, Ismail Gassoumi, and Abdellatif Mtibaa. 2020. Design of low-power structural FIR filter using data-driven clock gating and multibit flip-flops. *J. Electr. Comput. Eng.* (2020).

[15] Chung-Wen Albert Tsao and Cheng-Kok Koh. 2002. UST/DME: A clock tree router for general skew constraints. *ACM Trans. Des. Autom. Electr. Syst.* 7, 3 (2002), 359–379. DOI: https://doi.org/10.1145/567270.567271

[16] Yi-Chen Lu et al. 2020. TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC'20)*. IEEE, 1–6.

[17] Jeremy Rogers and Steve Gunn. 2005. Identifying feature relevance using a random forest. In *International Statistical and Optimization Perspectives Workshop "Subspace, Latent Structure and Feature Selection."* Springer, 173–184.

[18] Yi-Chen Lu, Sai Pentapati, and Sung Kyu Lim. 2021. The law of attraction: Affinity-aware placement optimization using graph neural networks. In *Proceedings of the International Symposium on Physical Design*. 7–14.

[19] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2021), 4–24. DOI: http://dx.doi.org/10.1109/TNNLS.2020.2978386

[20] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, et al. 2018. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *Proceedings of the ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA'18)*. IEEE, 29–42.

Received 18 August 2022; revised 30 January 2023; accepted 21 February 2023