Flip-Flop Centric Incremental Placement for Simultaneous Timing and Clock Network Power Optimization

Cristhian Roman-Vicharra
Texas A&M University
cristhianroman@tamu.edu

Yiran Chen
Duke University
yiran.chen@duke.edu

Jiang Hu
Texas A&M University
jianghu@tamu.edu

Abstract

Flip-flops are simultaneously endpoints of combinational logic timing paths and leaf nodes of clock networks. Hence, flip-flop placement significantly affects both circuit timing and clock network power. Previous works on flip-flop clustering/placement are mostly based on geometric models, which have limited accuracy for timing and power estimation. Moreover, these methods cannot account for the impact of clock tree synthesis parameters. In order to overcome this drawback, we develop CNN-based circuit timing and clock network power models. The CNN models are further applied to guide flip-flop centric incremental placement for simultaneous timing and clock network power optimization. Experimental results on benchmark circuits show that our method outperforms state-of-the-art previous works on timing, power, and computation runtime.

CCS Concepts

• **Hardware → Electronic design automation**.

Keywords

Clock network power, timing performance, co-optimization, placement

1 Introduction

Apart from functional correctness, satisfactory timing performance and low power dissipation are perhaps the two most important objectives for almost all digital IC designs. Flip-flops play a critical role in achieving both objectives due to their structural locations at the boundary between combinational logic circuits and clock networks. They are endpoints of circuit timing paths, and therefore flip-flop placement directly affects path delay as well as timing slack. At the same time, flip-flops are leaf nodes of clock networks, which often constitute the largest power consumer in many chip designs. Since the number of clock buffers and clock network wirelength heavily depends on flip-flop locations, flip-flop placement is also an effective leverage for clock network power reduction.

Due to the importance of flip-flop placement, there have been numerous studies on flip-flop clustering [2, 3, 6–10, 14–16]. Many of these works aim to facilitate multi-bit flip-flops, which are significantly more power-efficient than unclustered flip-flops. Indeed, remarkable clock network power reduction has been achieved in these works. However, most of these works handle circuit timing in a very conservative manner. Specifically, they move only flip-flops while keeping all logic cells fixed. With this constraint, timing is addressed by limiting flip-flop moving ranges according to the timing slacks prior to clustering. Thus, timing is merely addressed according to the principle of "do-no-harm" instead of proactive optimization. There are two drawbacks to this approach. First, flip-flops associated with negative slacks cannot be moved although a movement may potentially improve timing. Second, fixed logic cells restrict the degree of freedom for flip-flops to be clustered. For example, in Figure 1(a), the flip-flop movement is restricted to be within the red dotted box due to limited timing slack and fixed logic cells. However, if the logic cells can move as in Figure 1(b), the flip-flop can be relocated out of the box without degrading timing. Hence, moving logic cells allows an increased chance for flip-flops to be clustered.

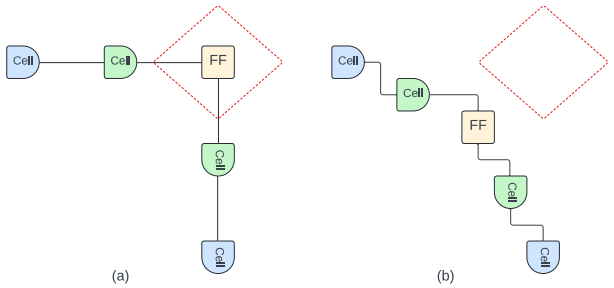

Figure 1: (a) Conventional techniques where logic cells are fixed and the movement of a flip-flop is bounded (in the red dotted box). (b) Our approach allows cells near flip-flops to be moved, enabling the flip-flop to move in a larger range for better clustering.

In [7], timing is explicitly optimized along with flip-flop clustering. However, the timing model used in [7] is overly simplified. Again, this work only allows flip-flops to be moved while keeping all logic cells fixed. Based on this simplification, the timing between two sequentially adjacently flip-flops is estimated by the Euclidean distance between them. The effectiveness of such estimation is restricted to designs where timing paths in layout are monotone. The example in Figure 2 illustrates a failure of this estimation. When the flip-flop locations are changed from Figure 2(a) to (b), the path wirelength as well as delay are slightly increased although the distance between the two flip-flops becomes smaller.

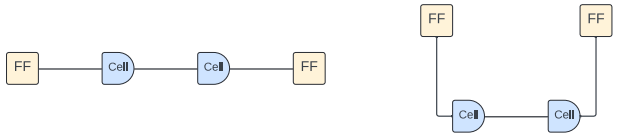

Figure 2: From (a) to (b), the two flip-flops are closer but the path delay is slightly increased.

It is observed that CTS (Clock Tree Synthesis) parameters may cause significant differences in timing and power even for the same placement solution. Figure 3 shows the total negative slack and clock network power for two circuits. Based on the same placement solution, CTS is performed on each circuit with three different parameter settings. However, this effect has been largely ignored in previous works on flip-flop clustering/placement.

In this work, we propose a new approach of Flip-flop Centric Incremental Placement (**FCIP**) in an effort to overcome the drawbacks of previous works. Given a global placement, our FCIP performs incremental placement for flip-flops and a small subset of logic cells structurally near flip-flops. Both circuit timing and clock network power are explicitly optimized in FCIP. Moreover, machine learning models that consider CTS parameters are constructed and employed in the placement optimization. Experimental results on benchmark circuits show that FCIP outperforms the state-of-the-art previous work on

MLCAD '24, September 9–11, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0699-8/24/09
https://doi.org/10.1145/3670474.3685949

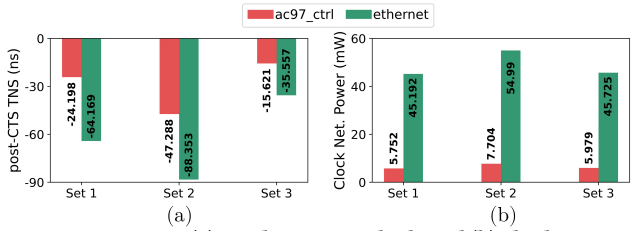

Figure 3: Post-CTS (a) total negative slack and (b) clock network power using the same placement solution with three different CTS parameter settings.

all of circuit timing, power and computation runtime. Although related, our FCIP is orthogonal to conventional flip-flop clustering and multi-bit flip-flop generation techniques. The contributions of this work are summarized as follows.

- This is the first incremental placement work optimizing both flip-flop and nearby logic cell locations for clock network power reduction, to the best of our knowledge.
- This is the first ML-based approach for simultaneous timing and power optimization in placement, to the best of our knowledge.
- Our incremental placement can capture the effect of CTS (Clock Tree Synthesis) parameters, which have not been considered in previous works.
- Experimental results on benchmark circuits show that our FCIP approach outperforms the state-of-the-art previous works in terms of timing, power, and computation runtime.

2 Previous Related Works

An early work on flip-flop placement is [11], where flip-flops are attracted to Manhattan circles during cell placement according to prescribed skew targets so that clock network size and power are reduced. However, its solution evaluation is based on a very simple power model. It also considers circuit timing in terms of Elmore delay model, which may be far away from static timing analysis. The work of [8] clusters flip-flops to form multi-bit flip-flops. It considers timing by enforcing wirelength bounds for fanin and fanout nets of flip-flops while logic cells are fixed. As such, timing is not explicitly optimized and it also causes 7.5% wirelength increase, which is significant. This work is extended in [16] to reduce the wirelength overhead. However, timing is still simply considered through wirelength bounds for nets incident to flip-flops without explicit optimization. In [15], flip-flops are clustered using weighted K-means algorithm while timing is addressed by merely minimizing disturbance to the initial placement. The work of [10] is also flip-flop clustering based on K-means algorithm and network flow, and does not explicitly optimize timing either. An ILP-based incremental flip-flop placement method is developed in [14], where timing is not explicitly optimized either. A heuristic [6] is proposed to merge flip-flops to form multi-bit flip-flops and no tradeoff with timing is considered. The work of [2] performs flip-flop clustering using mean shift algorithm, which leads to clock power reduction and timing performance degradation. In [13], a timing driven placement technique is proposed to be compatible with flip-flop clustering. It demonstrates significant timing improvement without reporting power results. In [7], flip-flops are clustered for optimizations of both power and timing. However, timing is evaluated with a geometric surrogate model, which is inaccurate. The work of [12] involves both clock and combinational logic domains. It is a useful skew optimization technique with focus on circuit timing while clock network power is not considered. In [3], a slack redistribution technique is introduced to improve timing of flip-flop clustering methods. However, the improvement is limited. Recently, a new heuristic is proposed in [9] for flip-flop clustering and demonstrates encouraging results. Overall, existing approaches either do not explicitly optimize timing or use overly simplified models.

3 Overview and Problem Formulation

Given a global placement solution, our FCIP incrementally places flip-flops and a subset of logic cells that are structurally near flip-flops such that a linear combination of Clock Network Power (CNP) and the absolute value of TNS (Total Negative Slack) is minimized subject to cell density constraints. The circuit is represented by a graph $G(V, E)$, where V is the set of cells including flip-flops and E indicates the set of nets. The cells are partitioned into three disjoint subsets V_{FF} for flip-flops, V_{mc} for movable logic cells and V_{fc} for fixed logic cells, i.e., $V = V_{FF} \cup V_{mc} \cup V_{fc}$. The movable logic cells are those within k-hops from any flip-flops in the circuit graph, where k is a parameter. The decision variables are $\mathbf{p} = (\mathbf{x}, \mathbf{y})$, the location vectors for all flip-flops and movable cells $V_{FF} \cup V_{mc}$. The entire layout region is tessellated into a 2D array of bins B and ρ_b represents the cell density in a bin $b \in B$. Then, the FCIP problem is formulated as

$$\min_{\mathbf{p}} \ \Phi(\mathbf{p}) = \alpha P(\mathbf{p}) + (1-\alpha)|T(\mathbf{p})|$$

$$\text{subject to } \rho_b \leq \rho_{max}, \forall b \in B,$$

W where $P(\mathbf{p})$ is the total clock network power, $T(\mathbf{p})$ is the TNS, ρ_{max} is the upper limit for bin cell density, and α is a weighting factor for the tradeoff between clock network power and TNS. TNS and power values are normalized before being added together. Please note that wirelength is implicitly captured in the timing model. Moreover, the overall impact on the wirelength from an incremental placement is small. As shown in Figure 4, placement legalization and CTS (Clock Tree Synthesis) are subsequently performed. The final solutions are evaluated according to post-CTS timing and power analysis.

Figure 4: FCIP (Flip-flop centric incremental placement) in a design flow.

4 The Proposed FCIP Method

In this section, we describe machine learning models for clock network power and post-CTS TNS estimation, and the FCIP optimization method.

4.1 Machine Learning Models for Power and Timing Estimation

Given a global placement solution, the machine learning models are to estimate clock network power and post-CTS TNS resulting from the placement. We employ CNN (Convolutional Neural Network) models for two reasons. First, the model is preferred to be differentiable so that it can provide gradients in solving the nonlinear programming problem of FCIP. Therefore, decision tree-based models, such as random forest and XGBoost, and SVM (Support Vector Machine) are excluded, despite their popularity. Second, it should provide adequate accuracy and CNNs are shown to be more accurate than other differentiable models such as linear regression.

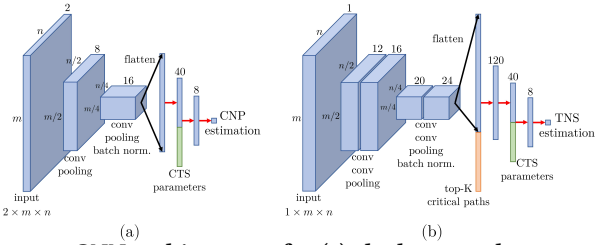

Figure 5: CNN architectures for (a) clock network power and (b) post-CTS TNS estimation.

The CNN architectures for clock network power and post-CTS TNS estimation are depicted in Figure 5. The power model consists of 2 convolution layers, a max-pooling layer, a batch normalization

layer, and 3 fully connected (FC) layers. Similarly, the timing model consists of 2 stages of 2 convolution and a max-pooling layers, a batch normalization layer, and 4 FC layers. In both models, we use the rectified linear unit (ReLU) function for the FC layers. Layout-related spatial features are fed into the convolution layers, while non-spatial features, such as CTS parameters, are directly fed to FC layers. The outputs provide regression results of power and timing estimation.

4.2 Clock Network Power Model Features

Feature selection plays a critical role in the effectiveness of a machine learning model. The model features for clock network power estimation are summarized as follows.

- Flip-flop placement density distribution. Evidently, clock network power heavily depends on flip-flop distributions. For the same number of flip-flops occupies the same area, clustered flip-flops lead to smaller clock network power than uniformly distributed flip-flops. The flip-flop density is described in the same way as cell density in RePlAce [4] such that the density function is differentiable with respect to flip-flop locations. If the entire layout region is tessellated into a 2D array of bins B, then the flip-flop density of a bin $b \in B$ is described by
$$D_{FF}(b, \mathbf{p}) = \sum_{FF_i \in b} q_i \psi_i, \tag{1}$$
where q_i is the electric charge proportional to the area of flip-flop FF_i, and ψ_i the electric potential of flip-flop FF_i. The bin densities form a 2D array similar to an image feature for CNN.
- Density function $D(b, \mathbf{p})$ for all cells including flip-flops, movable logic cells and fixed logic cells. Although logic cells are not part of a clock network, they present placement congestion that may affect clock buffer placement. This density function also forms a 2D array and can be treated as an image.
- The CTS parameter values form a one-dimension vector and serve as additional features fed to an FC layer in the CNN.

4.3 Timing Model Features

There are two important features for the timing (TNS) model: spatial and path-based.

- The **spatial feature** associates placement with timing criticality of each net. For each net $e \in E$, its driver resistance is denoted by R_e and its total sink capacitance is represented by C_e. Let $d_{e,0}$ be the logic depth from this net to its source flip-flop and $d_{e,j}$, j>0 be the logic depth from the j_{th} sink of net e to the destination flip-flop, then the logic path length of net e is defined by
$$L_e = d_{e,0} + \max(d_{e,1}, d_{e,2}, \cdots).$$
The HPWL (Half-Perimeter Wire-Length) of each net e is estimated in the same way as in RePlAce [4], and its x-coordinate component is given by
$$W_e(\mathbf{p}) = \frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)},$$
where x_i is the x-coordinate of pin i in the net and γ is a smoothing factor. Please note that this function is differentiable with respect to cell locations. Then, the spatial feature of net $e \in E$ is given by
$$F_e(\mathbf{p}) = L_e \cdot R_e \cdot C_e \cdot W_e(\mathbf{p}). \tag{2}$$
To a certain degree, this represents a net wirelength weighted by its timing criticality. For each placement bin $b \in B$, the spatial feature is given by
$$F(b, \mathbf{p}) = \sum_{e \text{ overlap } b \in B} \gamma_e(b) \cdot F_e(\mathbf{p}), \tag{3}$$
where $\gamma_e(b)$ shows the proportion of net bounding box of e overlapping with $b \in B$. Nets incident to only fixed cells are also included in the model as context. Even the same features of movable cells may result in different timing estimates in different contexts.
- The **path-based feature** for a single path is defined as
$$F_{path}(\mathbf{p}) = \sum_{e \in path} R_e \cdot C_e \cdot W_e(\mathbf{p}) \tag{4}$$
We extract the top-K timing critical paths according to timing analysis from the initial global placement solution and feed the K features of these paths to an FC layer in the CNN model.
- The CTS parameter values serve additional features fed to an FC layer since clock skew affects circuit timing.

4.4 Simultaneous Timing and Power Optimization

The FCIP problem formulated in Section 3 is solved using a gradient descent heuristic with backtracking. Starting from an initial global placement solution, flip-flops and movable cells are moved iteratively. In each iteration, the gradients of the objective function $\Phi(\mathbf{p})$ with respect to coordinates of V_{FF} and V_{mc} are computed according to the CNN models and the chain rule. Next, flip-flops and movable cells are moved in the negative direction of the gradients with certain step sizes. Movable cells structurally farther away from flip-flops are moved with smaller step sizes to limit perturbation to the initial solution. If these moves cause a violation of the bin cell density constraint $\rho_b \leq \rho_{max}$, some cell moves are backtracked till the constraint is satisfied. Once feasible moves are committed, the algorithm proceeds to the next iteration. The iterations terminate when the gradients are sufficiently small.

4.4.1 Power gradient computation. First, the gradient of clock network power $P(\mathbf{p})$ with respect to features $D_{FF}(b, \mathbf{p})$ and $D(b, \mathbf{p})$ are obtained as $\nabla_{D_{FF}} P(b, \mathbf{p})$ and $\nabla_D P(b, \mathbf{p})$, respectively, for all bins $b \in B$ through backpropagation in the CNN.

For a cell $v_i \in V_{FF} \cup V_{mc}$, which is located in bin $b \in B$, its corresponding power gradient with respect to its x-coordinate is obtained according to the chain rule as
$$\nabla_{x_i} P(\mathbf{p}) = \frac{\partial D_{FF}(b, \mathbf{p})}{\partial x_i} \cdot \nabla_{D_{FF}} P(\mathbf{p})|_b + \frac{\partial D(b, \mathbf{p})}{\partial x_i} \cdot \nabla_D P(\mathbf{p})|_b$$
where $\nabla_{D_{FF}} P(\mathbf{p})|_b$ is the power gradient with respect to feature $D_{FF}(b, \mathbf{p})$ for bin b. In the electrostatic analogy from RePlAce, the gradient of the potential energy is differentiable and computed as
$$\frac{\partial D(b, \mathbf{p})}{\partial x_i} = -q_i \mathcal{E}_{\mathbf{x}}(b),$$
where $\mathcal{E}_{\mathbf{x}}$ is the horizontal electric field in bin $b \in B$. The gradients for y-coordinates of flip-flops and movable cells are computed in the same way.

4.4.2 Timing gradient computation. Similarly, the gradient of TNS with respect to the x-coordinate of cell $v_i \in V_{FF} \cup V_{mc}$ is obtained by
$$\nabla_{x_i} T(\mathbf{p}) = \frac{\partial F(b, \mathbf{p})}{\partial x_i} \cdot \nabla_F T(\mathbf{p})|_b + \frac{\partial F_{path}(\mathbf{p})}{\partial x_i} \cdot \nabla_{F_{path}} T(\mathbf{p}),$$
where b indicates the bin for v_i and $\nabla_F T(\mathbf{p})|_b$ is the timing gradient with respect to feature $F(b, \mathbf{p})$ in bin b. Using the definitions in Section 4.3, we calculate the derivative of $F(\mathbf{p})$ as
$$\frac{\partial F(b, \mathbf{p})}{\partial x_i} = \sum_{e \text{ overlap } b} \gamma_e(b) \cdot L_e \cdot R_e \cdot C_e \cdot \frac{\partial W_e(\mathbf{p})}{\partial x_i},$$
and the derivative $\frac{\partial F_{path}(\mathbf{p})}{\partial x_i} = \sum_{e \in path} R_e \cdot C_e \cdot \frac{\partial W_e(\mathbf{p})}{\partial x_i}$, where $\frac{\partial W_e(\mathbf{p})}{\partial x_i}$ is computed in the same way as in RePlAce and is 0 if net e does not contain v_i.

The overall algorithm of our FCIP heuristic is outlined in Algorithm 1. Line 1 implies that we ensure that the cell density after FCIP does not exceed that of the initial placement. Lines 4-6 check the termination criteria for the iterations. Lines 7-10 compute the power

and timing gradients with respect to cell locations, where subscripts f and p indicate features and placement, respectively. The overall gradient is calculated in line 12. Line 13 moves each cell, and where p_i indicates (x_i, y_i) for cell v_i. Note that the step size is scaled by 2^{-k_i}, where k_i is the number of graph hops from v_i to its nearest flip-flop. Although we allow some logic cells to move, we restrict the moving range if a cell is not structurally close to a flip-flop. This is to limit perturbation to the initial solution, which has been optimized for wirelength minimization. Lines 15-22 enforce the cell density constraint. If the cell density of a bin exceeds ρ_{max}, we backtrack the locations of the cells in the bin one by one till the density constraint is satisfied.

Algorithm 1 FCIP (Flip-flop Centric Incremental Placement)

Input: Initial placement $\mathbf{p}^0$, step size η, weight α, max iterations j_{max}
Output: Optimized placement $\mathbf{p}^{opt}$

1: $\rho_{max} \leftarrow$ max_density($\mathbf{p}^0$) ▷ Density of initial placement
2: $j \leftarrow 0$ ▷ Iteration index
3: **while** $j \leq j_{max}$ **do**
4: **if** gradient is sufficiently small **then**
5: **return** $\mathbf{p}^{opt} \leftarrow \mathbf{p}^j$
6: **end if**
7: $\nabla_f P \leftarrow$backpropagation($P(\mathbf{p}^j)$) ▷ Gradients w.r.t. features
8: $\nabla_f T \leftarrow$backpropagation($T(\mathbf{p}^j)$)
9: Compute $\nabla_{\mathbf{p}} P(\mathbf{p}^j)$ ▷ Gradients w.r.t. placement
10: Compute $\nabla_{\mathbf{p}} T(\mathbf{p}^j)$
11: **for** each $v_i \in V_{FF} \cup V_{mc}$ **do** ▷ Update cell locations
12: $\nabla_{p_i} \Phi(\mathbf{p}^j) \leftarrow \alpha \nabla_{p_i} P(\mathbf{p}^j) + (1-\alpha) \nabla_{p_i} T(\mathbf{p}^j)$
13: $p_i^{k+1} = p_i^k - \eta \cdot 2^{-k_i} \cdot \nabla_{p_i} \Phi(\mathbf{p}^j)$ ▷ k_i hops from FF
14: **end for**
15: **for** each $b \in B$ **do** ▷ Enforce density constraint
16: $i \leftarrow 1$ ▷ Index of cells in b
17: **while** $\rho_b > \rho_{max}$ **do**
18: $p_i^{j+1} \leftarrow p_i^j$ ▷ p_i is location of cell v_i
19: $i \leftarrow i + 1$
20: Update ρ_b
21: **end while**
22: **end for**
23: **end while**
24: **return** $\mathbf{p}^{opt} \leftarrow \mathbf{p}^{j_{max}}$

5 Experiments

5.1 ML Model Training and Testing Data

The machine learning model training and testing data are generated based on 14 circuit designs in the IWLS 2005 benchmarks [1] using 45nm technology. These designs are in the top 14 rows of Table 1. For each design, two logic synthesis results are obtained through Synopsys Design Compiler using different parameter settings. Subsequently, 30 placement results (unclustered solutions) are generated for each synthesis solution using Cadence Innovus with different parameter values. For 30 of the placement solutions, 9 flip-flop clustered solutions are generated according to the K-means algorithm as [15] using different parameter values. Each clustered solution is then perturbed through data augmentation techniques to produce 10 perturbed solutions. Each unclustered solutions is sent to Cadence Innovus to acquire 128 CTS solutions using different parameter values, while each clustered-perturbed solution to acquire a single CTS solution. In total, there are 2×30×128 + 30×9×(1+10)=10650 data samples per design. However, the *leon2* design has 1416 data samples due to the extended data generation time, following the same procedure as the other designs. Consequently, the total number of data samples amounts to 139866. Among these 14 designs, 11 are used for model training and validation during the training procedure, while the remaining 3 are for testing (boldface in Table 1). Thus, training and testing data are strictly separated.

Table 1: Designs for ML model and optimization: Top 11 for model training, middle 3 (in boldface) for model testing, and bottom 7 for testcases of optimization.

Design	Nestlist 1		Netlist 2	
	# FFs	# Total	# FFs	# Total
aes_core	530	15,906	530	15,529
des3_area	128	1,865	128	1,833
mem_ctrl	1,065	4,078	1,083	4,182
systemcaes	670	4,932	670	4,975
systemcdes	190	2,444	190	2,401
tv80	359	5,851	359	5,758
usb_funct	1,745	8,951	1,746	8,356
vga_lcd	17,057	56,441	17,079	56,266
wb_conmax	770	25,881	770	24,679
wb_dma	523	2,470	563	2,581
leon2	364,521	513,965	-	
ac97_ctr	2,199	7,279	2,199	6,975
ethernet	10,544	38,998	10,544	37,971
pci_bridge32	3,315	11,296	3,359	11,128
des_perf	8,808	13,139	-	
leon3-avnet	451,560	636,525	-	
leon3mp	265,666	374,494	-	
netcard	248,453	346,249	-	

5.2 Power and Timing Model Performance

The performance of the machine models is evaluated using the MSE (Mean Squared Error) and the coefficient of determination (R2) metrics on the testing dataset. MSE is defined as $MSE=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y_i})^2$, where y_i is the ground truth in the testing dataset and $\hat{y_i}$ is the prediction by the model. The R2 coefficient is defined as $R^2=1-\sum_{i=1}^{n}(y_i-\hat{y_i})^2/\sum_{i=1}^{n}(y_i-\bar{y})^2$, where $\bar{y}$ is the mean of ground truth values in the testing dataset.

Table 2: Performance of CNN and linear regression models.

Design	Model	Power		Timing	
		MSE	R2	MSE	R2
ac97_ctrl	CNN	0.224	0.937	0.905	0.958
	Linear R.	0.738	0.793	3.519	0.836
ethernet	CNN	0.256	0.890	1.099	0.924
	Linear R.	0.784	0.663	4.153	0.713
pci_bridge32	CNN	0.253	0.926	1.012	0.953
	Linear R.	0.782	0.771	3.604	0.832
Average	CNN	0.244	0.918	1.005	0.945
	Linear R.	0.768	0.742	3.759	0.794

We compared our proposed CNN models with linear regression models, an alternative differentiable model, and the results are shown in Table 2. Evidently, CNN significantly outperforms linear regression in terms of both MSE and R2 coefficient, confirming our choice of CNN as architecture for our power and timing models. The total CNN training time for the timing model is approximately 2.4 hours, while for the power model it is 0.6 hours, using an NVIDIA GeForce MX130 GPU during the training procedures. The CNN and linear regression models are implemented using PyTorch and scikit-learn libraries in Python language, respectively.

5.3 Optimization Testcases and Experiment Setup

The three testing designs with the first synthesis parameters (netlist 1 and boldface) and four designs at the bottom of Table 1 are employed for evaluating the following four placement optimization techniques.

- **Baseline**: RePlAce [4], a state-of-art placer, is used for global placement of a design to obtain the initial placement solution. Then, the solution is legalized and used as the baseline.
- Our **FCIP** is performed upon initial global placement solutions obtained through RePlAce and then legalized. The step size η is typically set to 0.001, K timing critical paths to 30, hop k_i to {2,3,4} and max iteration j_{max} to [100,200] based on the design size. The feature extraction, chain-rule computation, and gradient-based optimizer are implemented using C++ language.
- **FCIP-P** is a variation of FCIP that aims to optimize clock network power without degrading post-CTS TNS. It is performed upon initial global placement obtained from RePlAce, followed by legalization of the solution.
- **W-Kmeans** [15], is a previous work of flip-flop clustering based on an initial global placement, which is RePlAce. The placement solution after clustering is also legalized.
- **FLA** [7] is a state-of-the-art previous work that optimizes both timing and power, although it employs a very simple timing model. Its ILP (Integer Linear Programming) is solved using the Gurobi solver. The solutions after FLA are legalized.

The legalization for all methods is obtained by *OpenDP* [5]. CTS is performed using Cadence Innovus for all legalized solutions. The CTS parameters used in the models and CTS procedure are max skew, max fanout, max cap trunk, max cap leaf, max trans trunk, max trunk leaf, and max latency. All timing and power results are based on post-CTS analysis using Innovus. The experiments were run on an Intel Core i7-1065G7 CPU 1.3GHz with 16GB RAM.

5.4 Main Results on Timing and Power Optimization

Table 3 shows the results for the four methods FCIP, FCIP-P, W-Kmeans and FLA. The results indicate that our FCIP and FCIP-P introduce almost similar disturbance compared to FLA [7], and much less than W-Kmeans [15]. Our FCIP and FCIP-P achieve average saving in clock network power of 9.4% and 14.1%, respectively, compared to 7.3% for FLA and 5.3% for W-Kmeans. In terms of post-CTS TNS, our FCIP shows an average improvement of 15.9%, while FLA of 8% and W-Kmeans of 8.9%. Our FCIP disturbs the HPWL in the initial solution by an average of 2%, while FLA of 1.9% (a similar amount) and W-Kmeans of 3.5%. In FCIP-P, the timing improvement is constrained using the weighting factor α, allowing for an additional 4.7% saving in clock network power over FCIP at the cost of a small increment of HPWL. In terms of CPU runtime, our FCIP is 3× faster than FLA and takes slightly more time than W-Kmeans.

5.5 Importance of CTS Parameters for Timing/Power Optimization

CTS parameters are included in the features for our CNN-based timing and power models. To demonstrate the importance of including CTS parameters, we compare optimizations with CNN models that do not use CTS parameters in the following variants.

- FCIP-woCTS. Only the CTS parameters are skipped in the CNN models while keeping everything else the same as in FCIP. Please note that one placement solution leads to multiple CTS solutions with different parameters in training data.
- FCIP-1train-sameCTS. Similar to FCIP-woCTS, except that only one CTS solution is used for each placement in the training. The CTS parameters in the final solution evaluation are the *same* as those in the training data.
- FCIP-1train-diffCTS. Similar to FCIP-woCTS, except that only one CTS solution is used for each placement in the training. The CTS parameters in the final solution evaluation are *different* from those in the training data.

The normalized average results for three circuits are shown in Figure 6. It can be seen that FCIP-woCTS worsens WNS, TNS, and clock network power. Training models with multiple CTS solutions for a single placement solution means that one set of features corresponds to different labels, making model training difficult and ineffective. The results from FCIP-1train-diffCTS are also poor as the CTS parameters in the final solution evaluation are different from those in the training. Although the results of FCIP-1train-sameCTS are good, its use is highly restrictive, requiring CTS parameters in a design flow to be fixed and the same as those in the model training data. Overall, FCIP produces the best results with much better flexibility.

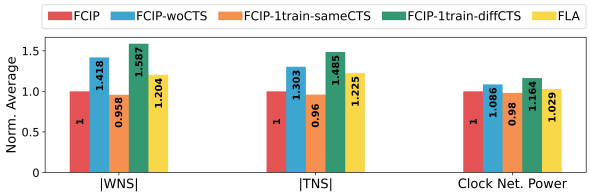

Figure 6: Comparing results using timing and power models without CTS parameters as features.

5.6 Power-Timing Tradeoff

We study the impact of the weighting factor α on post-CTS TNS and clock network power across optimization iterations in Figure 7 for the design *ac97_ctrl*. In the timing-only mode, where α=0, the absolute value of TNS decreases over iterations and eventually reaches 12.86ns in TNS, while the power increases from 5.75 to 5.92mW. On the other hand, the power-only mode achieves a power reduction to 4.85mW at the cost of worsening TNS from −24.20 to −25.49. The co-optimization mode (α=0.45) achieves results in between the timing- and power-only modes.

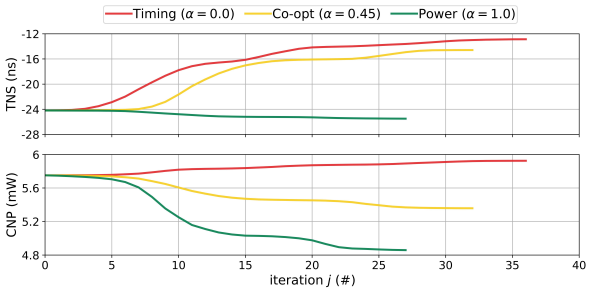

Figure 7: Optimization of design *ac97_ctrl* in timing-, power-only and co-optimization modes.

In Figure 8, we present the timing-power tradeoff for two designs (*ac97_ctrl* and *ethernet*) obtained through our FCIP by varying the weighting factor α and hops k, which is the max-hop-distance from movable cells to a flip-flop. The tradeoff curves are also compared with solutions from alternative methods. In most times, our FCIP solutions dominate those of the baseline, W-Kmeans and FLA. Setting k=0 is similar to many previous works where all logic cells are fixed. It is evident that allowing some logic cells to move generally improves solution quality. The knee-point solutions are usually around α=0.5. On the other hand, increasing the hops k would eventually degrade the solution quality in both timing and power.

5.7 Impact of Step Size η in Results

The step size η is an important parameter that determines the convergence and the number of iterations in the gradient search. Figure 9 shows that there is a sweet spot around η=0.001, which is adopted in FCIP, and either smaller or larger step size η degrades both timing and power results.

Table 3: Optimization results of post-CTS TNS and clock network power for 7 designs in the IWLS 2005 benchmark.

Design	Method	HPWL ($\times 10^8$)	Timing (ns)		Power (mW)		CPU (sec)
			WNS	TNS	Clock net.	Total	
ac97_ctrl	Baseline	1.54	-0.039	-24.198	5.75	12.47	-
	FCIP	1.58	-0.026	-15.093	5.28	12.09	25
	FCIP-P	1.57	-0.037	-23.864	4.91	11.62	26
	W-Kmeans	1.59	-0.050	-27.620	5.43	12.38	10
	FLA	1.56	-0.030	-20.298	5.35	12.09	15
ethernet	Baseline	8.36	-0.320	-64.169	45.19	267.56	-
	FCIP	8.60	-0.199	-53.715	40.13	263.43	63
	FCIP-P	8.58	-0.316	-63.679	37.94	256.82	60
	W-Kmeans	9.02	-0.268	-71.146	42.91	269.00	28
	FLA	8.53	-0.252	-60.196	41.58	265.72	72
pci_bridge32	Baseline	3.52	-0.076	-6.193	5.60	20.33	-
	FCIP	3.62	-0.058	-4.708	4.80	19.71	33
	FCIP-P	3.64	-0.073	-6.142	4.59	19.34	35
	W-Kmeans	3.65	-0.107	-7.671	5.31	22.16	20
	FLA	3.60	-0.068	-5.689	4.98	20.25	56
des_perf	Baseline	6.71	-0.086	-67.100	28.21	149.03	-
	FCIP	6.82	-0.074	-57.290	24.50	145.48	44
	FCIP-P	6.85	-0.083	-66.719	23.39	144.56	39
	W-Kmeans	6.86	-0.092	-71.367	25.69	146.56	23
	FLA	6.83	-0.079	-60.164	25.63	146.07	64
netcard	Baseline	45.33	-1.463	-676.470	202.75	495.88	-
	FCIP	45.81	-1.204	-611.823	187.12	482.04	337
	FCIP-P	45.84	-1.461	-672.392	174.29	466.73	311
	W-Kmeans	46.16	-1.497	-697.182	192.84	485.17	295
	FLA	45.92	-1.292	-619.861	188.61	483.26	1024
leon3mp	Baseline	50.27	-1.655	-738.824	289.71	633.51	-
	FCIP	51.32	-1.529	-698.831	270.38	615.92	356
	FCIP-P	51.71	-1.638	-735.525	257.89	604.26	344
	W-Kmeans	52.19	-1.691	-754.973	275.97	619.98	312
	FLA	51.89	-1.576	-709.187	274.30	618.46	1270
leon3-avnet	Baseline	89.17	-1.920	-1718.146	498.48	1102.67	-
	FCIP	90.24	-1.817	-1656.362	475.31	1081.25	517
	FCIP-P	90.51	-1.917	-1711.384	459.11	1064.38	502
	W-Kmeans	91.16	-1.948	-1751.285	482.76	1087.19	496
	FLA	90.72	-1.892	-1665.731	480.92	1085.63	1792
Norm. Average	Baseline	1	1	1	1	1	-
	FCIP	1.020	0.799	**0.841**	**0.906**	0.975	1
	FCIP-P	1.022	0.978	0.993	**0.859**	0.953	0.958
	W-Kmeans	1.035	1.092	1.089	0.947	0.988	0.861
	FLA	1.019	0.884	0.920	0.927	0.982	3.122

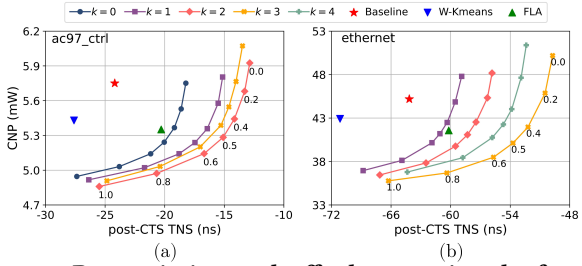

Figure 8: Power-timing tradeoff when varying the factor α from 0 to 1 and the hops k in FCIP for (a) *ac97_ctrl* and (b) *ethernet* designs.

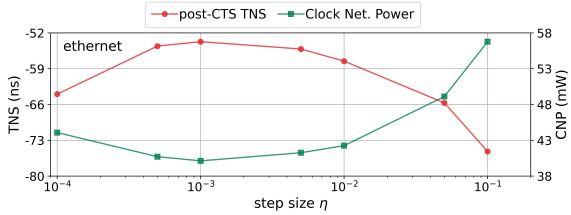

Figure 9: Optimization results for *ethernet* design using several values of step size η.

6 Conclusions and Future Research

Flip-flop placement significantly affects both circuit timing and clock network power. Previous works are mostly focused on power reduction and handle timing in a very conservative manner. We propose a simultaneous timing and clock network power optimization technique integrated with flip-flop centric incremental placement. The optimization is guided by machine learning models, enabling post-CTS timing and power estimation and capturing the effect of CTS parameter setting. Experimental results on benchmark circuits show that the proposed method significantly outperforms the state-of-the-art previous works in terms of timing, power and computation runtime. In future research, we will integrate FCIP with multi-bit flip-flop generation to obtain further power reduction.

Acknowledgments

This work is partially supported by NSF (CCF-2106725 and CCF-2212346) and SRC (GRC-CADT-3013.001/3014.001). We thank Mr. Hailiang Hu for discussions throughout this work.

References

[1] Christoph Albrecht. 2005. IWLS 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS)*, Vol. 9.

[2] Ya-Chu Chang, Tung-Wei Lin, Iris Hui-Ru Jiang, and Gi-Joon Nam. 2019. Graceful register clustering by effective mean shift algorithm for power and timing balancing. In *Proceedings of the 2019 International Symposium on Physical Design*. 11–18.

[3] Yen-Yu Chen, Hao-Yu Wu, Iris Hui-Ru Jiang, Cheng-Hong Tsai, and Chien-Cheng Wu. 2024. Slack Redistributed Register Clustering with Mixed-Driving Strength Multi-bit Flip-Flops. In *Proceedings of the 2024 International Symposium on Physical Design*. 21–29.

[4] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong Wang. 2018. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2018), 1717–1730.

[5] SangGi Do, Mingyu Woo, and Seokhyeong Kang. 2019. Fence-Region-Aware Mixed-Height Standard Cell Legalization. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI* (Tysons Corner, VA, USA) *(GLSVLSI '19)*. Association for Computing Machinery, New York, NY, USA, 259–262. https://doi.org/10.1145/3299874.3318012

[6] Chaochao Feng, Daheng Yue, Zhenyu Zhao, and Zhuofan Liao. 2018. A parameterized timing-aware flip-flop merging algorithm for clock power reduction. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 881–884.

[7] Chau-Chin Huang, Gustavo Tellez, Gi-Joon Nam, and Yao-Wen Chang. 2020. Latch Clustering for Timing-Power Co-Optimization. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218617

[8] Iris Hui-Ru Jiang, Chih-Long Chang, and Yu-Ming Yang. 2012. INTEGRA: Fast multibit flip-flop clustering for clock power saving. *IEEE Transactions on computer-aided design of integrated circuits and systems* 31, 2 (2012), 192–204.

[9] Andrew B Kahng, Sayak Kundu, and Shreyas Thumathy. 2024. Scalable Flip-Flop Clustering Using Divide and Conquer For Capacitated K-Means. (2024).

[10] Andrew B Kahng, Jiajia Li, and Lutong Wang. 2016. Improved flop tray-based design implementation for power reduction. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[11] Yongqiang Lu, CN Sze, Xianlong Hong, Qiang Zhou, Yici Cai, Liang Huang, and Jiang Hu. 2005. Register placement for low power clock network. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. 588–593.

[12] Yi-Chen Lu, Wei-Ting Chan, Deyuan Guo, Sudipto Kundu, Vishal Khandelwal, and Sung Kyu Lim. 2023. RL-CCD: Concurrent clock and data optimization using attention-based self-supervised reinforcement learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[13] Dimitrios Mangiras, Apostolos Stefanidis, Ioannis Seitanidis, Chrysostomos Nicopoulos, and Giorgos Dimitrakopoulos. 2019. Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2019), 2835–2848.

[14] Ioannis Seitanidis, Giorgos Dimitrakopoulos, Pavlos Mattheakis, Laurent Masse-Navete, and David Chinnery. 2017. Timing driven incremental multi-bit register composition using a placement-aware ILP formulation. In *Proceedings of the 54th Annual Design Automation Conference 2017*. 1–6.

[15] Gang Wu, Yue Xu, Dean Wu, Manoj Ragupathy, Yu-yen Mo, and Chris Chu. 2016. Flip-flop clustering by weighted K-means algorithm. In *Proceedings of the 53rd Annual Design Automation Conference* (Austin, Texas) *(DAC '16)*. Association for Computing Machinery, New York, NY, USA, Article 82, 6 pages. https://doi.org/10.1145/2897937.2898025

[16] Chang Xu, Guojie Luo, Peixin Li, Yiyu Shi, and Iris Hui-Ru Jiang. 2016. Analytical clustering score with application to postplacement register clustering. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 21, 3 (2016), 1–18.