Revisit MBFF: Efficient Early-Stage Multi-bit Flip-Flops Clustering with Physical and Timing Awareness

Yichen Cai
cai_yichen@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Linyu Zhu
linyuzhu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Xinfei Guo*
xinfei.guo@sjtu.edu.cn
Shanghai Jiao Tong University &
State Key Laboratory of Integrated Chips and Systems (SKLICS)
Shanghai, China

ABSTRACT

Despite the maturity of Multi-bit Flip-Flops (MBFF) clustering in modern Electronic Design Automation (EDA) tools for saving power, there remains a trade-off between the flexibility to cluster flip-flops and the overall quality of results (QoR). This paper proposes a novel approach to MBFF clustering, integrating early-stage physical and timing awareness to optimize the design quality. Our pre-placement MBFF clustering algorithm addresses this trade-off by incorporating early distance estimation and predicted skews, improving timing conditions without compromising power savings. We evaluate our approach using widely-used benchmark circuits, demonstrating significant improvements in power savings and timing compared to state-of-the-art techniques. Notably, our method achieves an average improvement of 22.5% in Worst Negative Slack (WNS) and 33.91% in Total Negative Slack (TNS), while reducing power by 3.01% compared to commercial tools with MBFF clustering enabled at placement. Against the state-of-the-art pre-placement MBFF clustering algorithm, our methodology shows 25.59% and 37.97% improvements in WNS and TNS, respectively, while further reducing power by 3.08%. In addition, the proposed approach proves robust against variations in early-stage path delay estimation, maintaining superior performance even with deviations of over 20%.

CCS CONCEPTS

• **Hardware → Physical synthesis**; **Electronic design automation**.

KEYWORDS

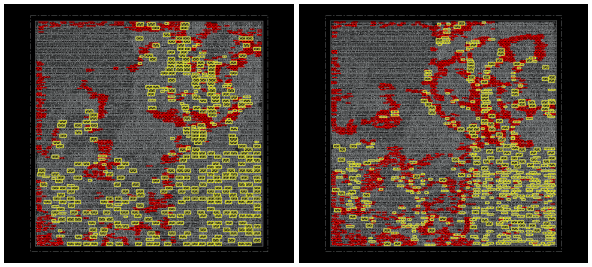
Low power, Multi-bit flip-flops, Clock skew, Quality of results

ACM Reference Format:
Yichen Cai, Linyu Zhu, and Xinfei Guo. 2025. Revisit MBFF: Efficient Early-Stage Multi-bit Flip-Flops Clustering with Physical and Timing Awareness. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697725

*Corresponding author.

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697725



(a) Clustering after synthesis. (b) Clustering after placement.

Figure 1: Layout comparison of the design `or1200` when performing MBFF clustering in two different stages: after synthesis and after placement. Cells highlighted in red represent SBFFs, and cells highlighted in yellow represent MBFFs.

Table 1: Comparison of design `or1200` between MBFFs clustering after synthesis and placement using MBFF flows provided by commercial tools. P_{clk} and P_{total} are clock power and total power in mW, while WNS and TNS are in ns.

	Clustering after synthesis	Clustering after placement
P_{clk}	50.88	57.87
P_{total}	116.34	125.71
WNS	-0.172	-0.153
TNS	-289.664	-226.893
Bits/Flop	2.709	1.732

1 INTRODUCTION

Power has remained one of the most challenging metrics to optimize in modern chip designs. Among various power-saving techniques, multi-bit flip-flop (MBFF) clustering has proven to be an effective backend method for reducing clock power [7, 12, 15]. By sharing a common input clock pin and two internal driving inverters, the total power needed for storing all bits using flip-flops (FFs) can be reduced. Additionally, by minimizing the number of clock drivers and interconnects, MBFFs simplify the clock distribution network and improve routability. According to [7], normalizing power consumption per bit reveals that a 2-bit MBFF can reduce power by 14% compared to single-bit flip-flops (SBFFs), while only requiring 4% less normalized area for the 2-bit MBFF cells.

To incorporate MBFFs in the design, we need to convert the existing SBFFs to MBFFs, a process known as MBFF clustering. This

Table 2: Summary of related works.

Clustering Stage	Work	Goals	Highlights
Pre-placement	[19]	Minimize power	Consider locality of flip-flops
In-placement	[8]	Minimize power Minimize clock latency	Consider clock trees
	[14]	Minimize power Increase flexibility for clustering	Guide the placement of flip-flops to better position
Post-placement	[18] [16] [17]	Minimize power	Consider the feasible region between flip-flops
	[4]	Minimize power & wirelength	
	[13] [6] [1]	Minimize power Minimize timing degradation	Consider slack redistribution and useful skew optimization
	[5]	Minimize power Minimize timing degradation	Use LC-MBFF and consider useful skew optimization
Post-route	[9]	Minimize power	Introduce a new type of MBFF named LC-MBFF

can be done at different stages of the design flow, either during synthesis or placement. Although MBFF clustering is a mature feature in modern EDA tools and has been researched for decades, there is still no consensus on the optimal stage for conversion to achieve better QoR. Fig. 1 shows two layouts of the OpenRISC 1200 (`or1200`) design [11] using a state-of-the-art commercial EDA tool, performing MBFF clustering at different stages. Fig. 1a illustrates the results when the tool inserts MBFFs after synthesis, while Fig. 1b shows the results when MBFFs are inserted after placement. It is evident that significantly fewer SBFFs remain when MBFF clustering is done after synthesis compared to after placement. This is because clustering after synthesis allows for greater flexibility, as none of the cells have been placed, and the placer can optimize the clustering result afterwards. In contrast, clustering after placement can only merge nearby flip-flops to ensure final timing. The phenomena can also be demonstrated by the resulting metrics of the two flows, as shown in Table 1. In this table, "Bit/Flip" indicates the average number of bits per flip-flop. If MBFFs are inserted after synthesis, there will be 56% more bits per flip-flop compared to clustering after placement. The additional merged MBFFs also lead to a reduction in clock tree power (P_{clk}) and total power (P_{total}), which aligns with designer expectations. However, it should also be noticed from the table that both the worst negative slack (WNS) and total negative slack (TNS) are worse if we cluster MBFF after synthesis instead of after placement. This is expected, as there is no physical information available just after synthesis. With detailed or partial placement and other physical information, MBFF clustering after placement can determine whether several flip-flops can be clustered based on position and timing information, improving the timing conditions of the final design.

The discussion above highlights an interesting trade-off between the extent to which SBFFs can be clustered and the impact on timing conditions—in other words, between clustering flexibility and QoR. Multiple attempts have been made to introduce MBFF clustering techniques at various stages of the digital circuit design flow to reduce clock power, as summarized in Table 2. [19] proposed clustering multiple SBFFs into MBFFs just after logic synthesis and before placement. They calculated the intersection ratio of fanout for each flip-flop group and applied a threshold to limit the number of FF groups considered. Iterations involving placement and routing were needed to meet timing requirements, as there was no timing or physical information available during the logic synthesis phase.

To overcome the lack of timing and physical information, many works [4, 16–18] chose to cluster flip-flops after placement. Given the already placed flip-flops, different approaches have identified feasible regions where flip-flops can be clustered while still satisfying slacks as launch and capture registers. [14] considered the MBFF clustering just after the initial global placement. They provided guidance to position flip-flops in more optimal locations to satisfy timing constraints and allow for more MBFF clustering. Similarly, [8] also chose to cluster MBFF at this stage. In addition to considering the physical locations of flip-flops, their work took clock latency changes into account. Clock tree synthesis was performed, and the clock latency after clustering was estimated. This approach could reduce final clock latency but was not as effective as other optimization methods like useful skew scheduling [2].

Useful skew scheduling adjusts some of the flip-flops' clock arrival time to fully utilize the available timing slack on the signal path. [13] constructed a useful-skew clock tree to identify the overlap region of arrival times that meet the timing constraints. [5] used a circuit graph to track the clock arrival time of flip-flops dynamically to apply useful skew, and also used loosely coupled MBFF (LC-MBFF) technique first proposed in [9]. [1] considered slack redistribution when constructing feasible regions of flip-flops to improve timing. [6] made use of empty space inside the MBFF cells by introducing extra clock inverters, allowing flexible clock skews within a single MBFF cell for useful skew scheduling. This work also proposed an MBFF debanking technique to identify and debank MBFFs that failed to meet timing and were infeasible for useful skew scheduling. Both studies enabled useful skew optimization, making it easier to meet timing constraints. However, these techniques needed to be applied at a later stage of the design flow, as they required information on placed positions, clock trees, and routing, which involves long iterations. Consequently, these approaches could not fully exploit useful skew scheduling due to the limitations imposed by the already placed flip-flops. In this work, we propose an efficient MBFF clustering algorithm in pre-placement stage with early physical and timing awareness to address this trade-off. The

MBFF clustering algorithm is applied before placement to maximize clustering flexibility while incorporating timing information, including clock skew, at this stage to ensure high-quality final results. This allows MBFFs to be inserted at an earlier stage, enhancing the overall design flexibility. The main contributions of this work are summarized as follows:

- Instead of running through the entire design flow for physical and accurate timing information, we propose sampling this information early through estimation and prediction.
- By integrating early-stage physical and timing awareness, the proposed method achieves superior timing performance while further reducing power consumption. It is also robust against errors in timing prediction.
- The proposed approach integrates closely with state-of-the-art commercial tools and cooperates easily with other optimization techniques.

2 PROPOSED APPROACH

An overview of the proposed early-stage MBFF clustering methodology is given in Fig. 2. After the completion of traditional logic synthesis, three additional steps are introduced: flip-flop distance estimation, signal path timing estimation, and MBFF clustering based on the conflict graph. Flip-flop distance estimation step estimates the physical distances between flip-flops to identify potential clusters based on the swift coarse placement result. Then in the signal path timing estimation step, path delay for input and output signal path, or input/output delay, is predicted based on a pre-trained path-based delay estimator. The last step performs MBFF clustering based on the conflict graph, which represents the relationships and constraints between flip-flops, helping to identify which flip-flops can be clustered together without causing conflicts. By considering both physical and timing constraints, this step ensures that the resulting MBFFs are optimized for performance and reliability. This section will discuss details about each step.

2.1 Flip-flop Distance Estimation

Given the synthesised netlist, we obtain a set $\mathcal{F}$ which contains all possible combinations of flip-flops in the design. For example, in a design with three flip-flops f_1, f_2, and f_3, $\mathcal{F}$ would be $\{\{f_1, f_2\}, \{f_2, f_3\}, \{f_1, f_3\}\}$, assuming we only consider 2-bit MBFFs for now[1]. However, most pairs in $\mathcal{F}$ should not be clustered due to low locality. Clustering flip-flops with low locality can increase signal wire length and degrade timing performance in later stages. To incorporate physical locality without invoking the compute-intensive placement stages, we estimate the positions of all flip-flops immediately after synthesis by performing a coarse placement. This is achieved using a commercial placement tool in floorplan mode so that a fast prototype of cell placement with low congestion effort and without considering any design rule checks (DRC) is done, and the coarse placement can result in a more than 87 times speed-up compared to a detailed placement. Fig. 3 illustrates a possible coarse placement for the aforementioned three flip-flops.

Once the locations of all the flip-flops are estimated, the Manhattan distance of each flip-flop pairs in $\mathcal{F}$ can be calculated. We

[1]The proposed approach is scalable to accommodate MBFFs with more bits; however, for the sake of clarity and simplicity, this paper primarily focuses on two-bit MBFFs.

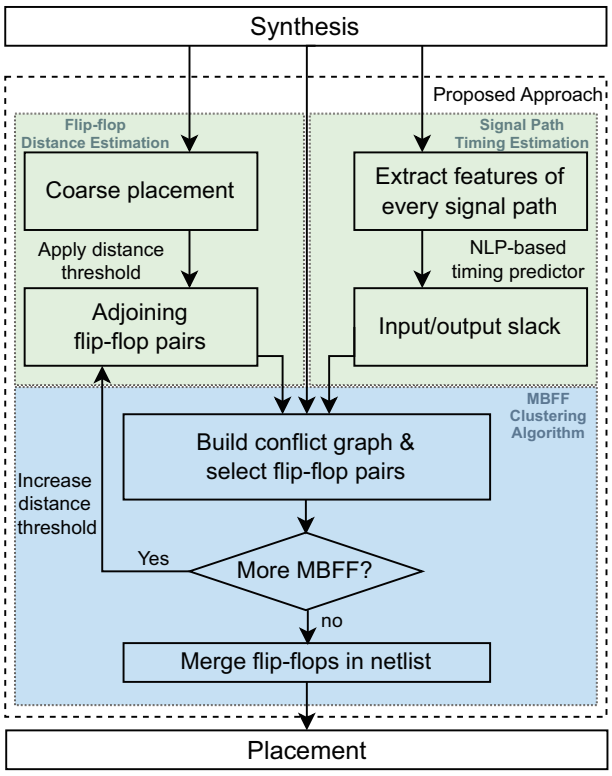


Figure 2: The proposed MBFF clustering flow.

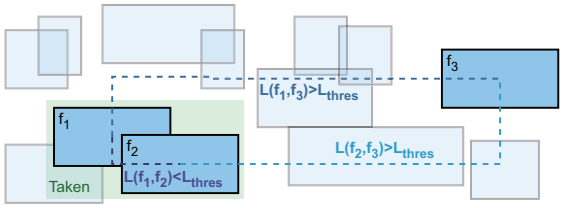


Figure 3: An example of flip-flop distance estimation.

can then apply a distance threshold L_{thres} to $\mathcal{F}$ to derive a new set of adjacent flip-flop pairs, $\mathcal{F}'$, defined as follows:

$$\mathcal{F}' = \{\{f_i, f_j\} : \{f_i, f_j\} \in \mathcal{F}, L(f_i, f_j) \leq L_{thres}\} \quad (1)$$

where $L(f_i, f_j)$ is the Manhattan distance between flip-flop f_i and f_j. The distance threshold L_{thres} should be chosen appropriately so that number of flip-flop pairs are reduced but a reasonable number of elements can be left for selection. As shown in Fig. 3, after applying the distance threshold L_{thres} on $\mathcal{F}$, only the flip-flop pair $\{f_1, f_2\}$ is considered for the subsequent MBFF clustering. Compared with the traditional pre-placement MBFF clustering strategy without physical awareness, this approach significantly reduces the number of elements in $\mathcal{F}'$ while ensuring the physical locality of flip-flop pairs.

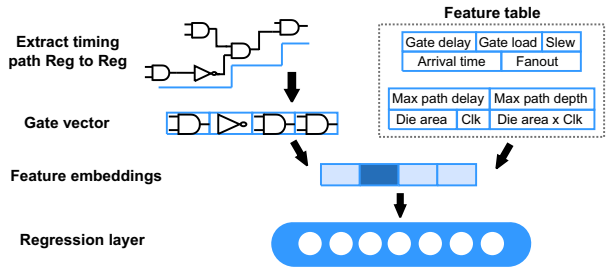


Figure 4: The NLP-based for predicting the signal path timing operates as follows: First, timing paths are extracted from netlists into gate vectors. After being looked up in the feature table, the gate vector is embedded as feature embeddings. Then, the regression layer outputs the predicted input delay (d_{in,f_n}) and the output delay (d_{out,f_n}) for flip-flop f_n.

2.2 Signal Path Timing Estimation

In the pre-placement stage, the timing information is typically obtained based on either Wire Load Model (WLM) or Physical Layout Estimation (PLE) model. These models relax setup timing without considering actual routing information, potentially resulting in unexpected timing issues in later stages. Recently, machine learning-based timing prediction models have shown promise in predicting interconnect delay in the early design phase, such as after logic synthesis. One proposed Natural Language Processing (NLP)-based timing predictor is leveraged, which has demonstrated great prediction accuracy [20, 21]. The model has been retrained to specifically predict the signal path timing concerning the input and output sides of the flip-flops, as shown in Fig. 4. In addition to basic gate features, global design constraints are also selected as global features. The NLP-based timing prediction model is then trained as a regression model using golden path delay results extracted from Static Timing Analysis (STA) in a standard design flow without involving any MBFFs. This enhancement ensures that our timing predictions are more accurate and relevant for capturing the physical awareness of the signal paths.

Since the layout will significantly change after the MBFF clustering algorithm, the results from the NLP-based timing predictor show not so perfectly accurate. Therefore, the estimation results are only treated as guidance for path delay rather than golden values, allowing the MBFF clustering algorithm to capture the relative differences of path delays. Furthermore, since the estimation results are not treated as golden values, we design the clustering algorithm to be robust against predictor errors. This approach ensures that the accuracy of the timing prediction model does not significantly impact the effectiveness of our proposed clustering approach.

After training the timing prediction model, critical input and output paths for each flip-flop are identified. This process allows us to estimate and denote both the input delay (d_{in,f_n}) and the output delay (d_{out,f_n}) for flip-flop f_n.

2.3 MBFF Clustering Algorithm

Based on the estimated flip-flop distances and signal path delays, a weighted conflict graph $G(V, E, W)$ can be constructed. Each vertex in the graph, $v_i \in V$, represents an adjoining flip-flop pair $\{f_{i,1}, f_{i,2}\} \in \mathcal{F}'$ obtained in the flip-flop distance estimation step, while an edge $e_{ij} \in E$ between vertices v_i and v_j exists if there is a common flip-flop element f_n that exits in both of the two flip-flop pairs $\{f_{i,1}, f_{i,2}\}$ and $\{f_{j,1}, f_{j,2}\}$. Each vertex is assigned with a weight $w_i \in W$ as the saved power calculated by $w_i = 2 * P_{SBFF} - P_{MBFF}$, where P_{SBFF} and P_{MBFF} denote the power consumption of a single-bit flip-flop and a 2-bit multi-bit flip-flop respectively.

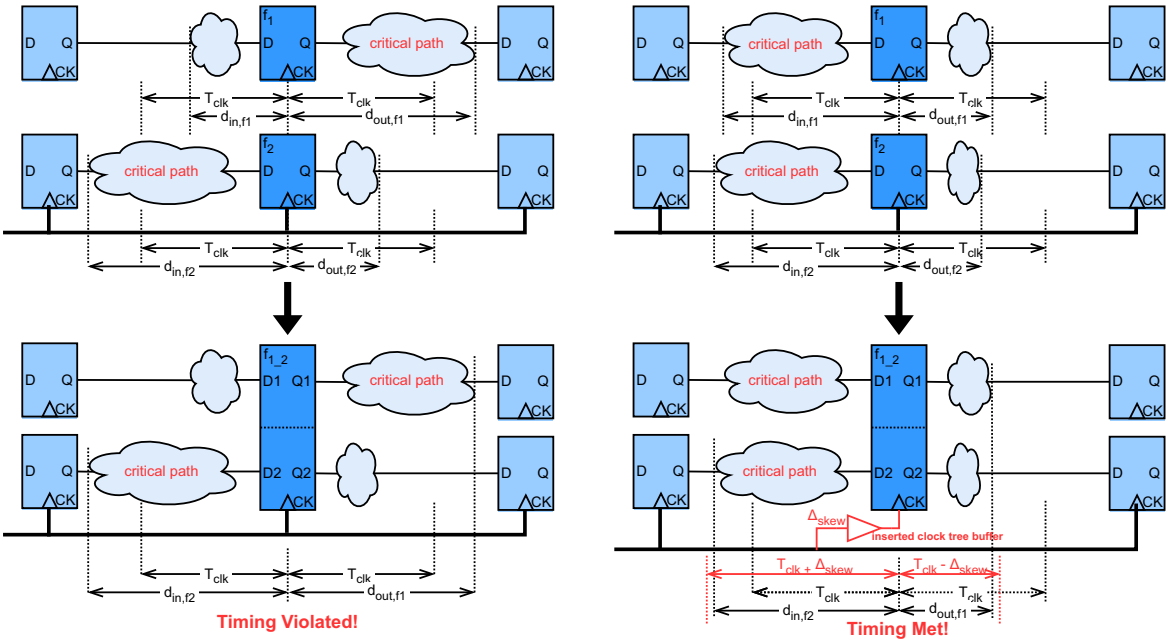
After constructing the weighted conflict graph, the selection of flip-flop pairs for clustering is performed using a greedy algorithm. Connected vertices in the graph can only be chosen for once since they share a common flip-flop as an intersection between their respective flip-flop pairs. Consequently, once a vertex v_i is selected, all directly connected vertices cannot be selected further. Each vertex v_i is assigned with a value c_i, indicating the ratio between overall potential saved power $\Sigma_{adjacent} w_j$ and the actual saved power w_i after selecting the vertex v_i. To incorporate timing considerations, an additional timing coefficient is introduced. The final formula for the value c_i is formulated as follows:

$$c_i = \frac{\Sigma_{adjacent} w_j}{w_i} * (1 + \frac{|\Delta slk_{f_i} - \Delta slk_{f_j}|}{|\Delta slk_{f_i} + \Delta slk_{f_j}|}) \tag{2}$$

where $\Delta slk_{f_i} = (T_{clk} - d_{out,f_i}) - (T_{clk} - d_{in,f_i}) = d_{in,f_i} - d_{out,f_i}$ and $\Delta slk_{f_j} = (T_{clk} - d_{out,f_j}) - (T_{clk} - d_{in,f_j}) = d_{in,f_j} - d_{out,f_j}$ are the subtraction result between slacks of input and output signal path, or we call between the input and output slack, for flip-flop f_i and f_j.

As shown in Fig. 5, there are typically two types of flip-flop pairs of interests for clustering. Fig. 5(a) shows two flip-flops f_1 and f_2, with a large difference between Δslk_{f_1} and Δslk_{f_2}. In this case, when we try to cluster the two flip-flops together, we can't apply any of the optimization methods to deal with the long input and output critical paths. As a result, the timing condition of the final result will become worse than the one before clustering. For the second type of flip-flop pair, as shown in Fig. 5(b), Δslk_{f_1} and Δslk_{f_2} have only a small difference. In this case, even though the original critical path still exists and will result in a timing violation, skew optimization techniques can be applied. By inserting additional buffers before the MBFF in the clock tree and introducing extra clock skew, the timing violation can be fixed. Therefore, skew optimization techniques can improve the timing condition of the whole design only when the slack differences between input and output paths are fully utilized. Based on the above analysis, we conclude that the difference between the Δslk of the two flip-flops to be cluster needs to be as small as possible, or the timing coefficient $(1 + \frac{|\Delta slk_{f_i} - \Delta slk_{f_j}|}{|\Delta slk_{f_i} + \Delta slk_{f_j}|})$ should be close to 1. If a flip-flop pair has a large Δslk difference, its c_i will be penalized by the timing coefficient as a result.

After calculating the value c_i for each node, the node with the smallest c_i is selected, indicating that it is able to save the most power with minimal negative impact on the overall timing condition of the design. The flip-flop pair associated with the selected node is then added to a list for clustering. All vertices connected by an edge to the selected node are deleted from the conflict graph. Once these steps are completed, new values for c_i are calculated, and iterations continue until the graph is empty. The distance threshold L_{thres} is then slightly increased so that more remaining flip-flops



(a) Type I: FF pair with large Δ_{slk} difference. (b) Type II: FF pair with small Δ_{slk} difference.

Figure 5: The timing condition when two different types of FF pairs clustered into a 2-bit MBFF.

Algorithm 1: The proposed MBFF clustering algorithm

Input: Flip-flop positions, path delays, power of each type of flip-flop, $\mathcal{F}$
Output: Flip-flop pairs for clustering

Initialize L_{thres};
while *still remains flip-flops which can be clustered* **do**
  $\mathcal{F}' = \{\{f_1, f_2\} : \{f_1, f_2\} \in \mathcal{F}, L(f_1, f_2) \leq L_{thres}\}$;
  Build conflict graph $G(V, E, W)$;
  for *node i in graph* $G(V, E, W)$ **do**
    $w_i = 2 * P_{SBFF} - P_{MBFF}$;
  end
  while *graph is not empty* **do**
    for *node i in graph* $G(V, E, W)$ **do**
      $\Delta slk_{f_i} = d_{in,f_i} - d_{out,f_i}$;
      $\Delta slk_{f_j} = d_{in,f_j} - d_{out,f_j}$;
      $c_i = \frac{\sum_{adjacent} w_j}{w_i} * (1 + \frac{|\Delta slk_{f_i} - \Delta slk_{f_j}|}{|\Delta slk_{f_i} + \Delta slk_{f_j}|})$;
    end
    Find the smallest c_i in graph $G(V, E, W)$;
    Append the selected flip-flop pairs into list waiting for clustering;
    Remove selected flip-flops from $\mathcal{F}$;
    Delete adjoining nodes in graph $G(V, E, W)$;
  end
  Increase L_{thres};
end

can be considered in the clustering algorithm. The algorithm stops when fewer than 1% of new MBFFs are selected in one iteration. The whole process is shown in Algorithm 1.

3 EVALUATION RESULTS

3.1 Experimental Setup

We implement our proposed approach with Python and Tcl scripts, and commercial EDA tools are used for logic synthesis, placement and routing. The evaluation experiment is conducted using a 28nm technology node PDK, a standard cell library, and a 2-bit MBFF library. In terms of the benchmark circuits, the proposed approach is tested on an OpenRISC core implementation and six IWLS 2005 [3] and OpenCores [10] benchmark circuits. Statistic details of the benchmark circuits we used can be found in Table 3. The selection of these benchmarks is based on their usage in other works in the same area, as well as the differences in design scale and behavior. The signal path timing prediction model is trained using five unique open-source designs that are not listed in Table 3 to ensure that the designs are unseen.

Table 3: Statistics of benchmark circuits in the evaluation.

Benchmarks	#FFs	#Cells	#Nets	Clock Period (ns)
or1200	3627	23417	23716	1
mem_ctrl	1002	5008	5124	0.15
ac97_ctrl	2161	6519	6604	0.5
wb_conmax	770	23531	24663	0.35
ethernet	10534	34025	34123	3
vga_lcd	17050	51885	51974	6
des_perf	8808	66951	67188	0.35

3.2 Clustering Results and Comparisons

To convincingly evaluate the effect of our proposed approach, critical metrics are compared with a flow without MBFF clustering and two state-of-the-art MBFF clustering flows, as shown in Table 4. The two MBFF clustering flows for comparison represent state-of-the-art flows that cluster flip-flops at different stages, after placement and after synthesis, respectively. The first state-of-the-art MBFF clustering flow is executed using a commercial tool by enabling the provided "use MBFF for optimization" attribute after the initial placement. Only SBFFs and 2-bit MBFFs are enabled in this flow, and the MBFF optimization effort is set to maximum before placement

Table 4: Critical metrics of our proposed approach compared with a commercial tool with and without MBFF enabled, and a recent pre-placement MBFF clustering method. P_{clk} and P_{total} are clock and total power in mW, WNS and TNS are in ns.

Benchmarks	Commercial Tool without MBFF		Commercial Tool with MBFF		Pre-placement MBFF Clustering [19]		Proposed Approach	
or1200	P_{clk}	17.88	**13.74**	**23.15%**	14.10	21.14%	**13.74**	**23.15%**
	P_{total}	27.71	23.68	14.54%	23.60	14.83%	**23.26**	**16.06%**
	WNS	-0.206	-0.238	-15.53%	-0.250	-21.36%	**-0.217**	**-5.34%**
	TNS	-20.491	-17.517	14.51%	-17.723	13.51%	**-16.737**	**18.32%**
mem_ctrl	P_{clk}	32.49	28.91	11.02%	**26.03**	**19.88%**	26.60	18.13%
	P_{total}	39.44	36.17	8.29%	**33.44**	**15.21%**	33.83	14.22%
	WNS	-0.318	-0.337	-5.97%	-0.327	-2.83%	**-0.324**	**-1.89%**
	TNS	-156.437	-171.942	-9.91%	-179.379	-14.67%	**-164.090**	**-4.89%**
ac97_ctrl	P_{clk}	22.06	17.15	22.26%	17.23	21.89%	**16.56**	**24.93%**
	P_{total}	25.48	20.84	18.21%	20.93	17.86%	**20.10**	**21.11%**
	WNS	-0.281	-0.321	-14.23%	-0.333	-18.51%	**-0.275**	**2.14%**
	TNS	-16.554	-31.108	-87.92%	-23.016	-39.04%	**-9.017**	**45.53%**
wb_conmax	P_{clk}	15.30	11.38	25.62%	**9.76**	**36.21%**	11.13	27.25%
	P_{total}	37.61	33.50	10.93%	**31.15**	**17.18%**	33.65	10.53%
	WNS	-0.118	-0.112	5.08%	-0.209	-77.12%	**-0.084**	**28.81%**
	TNS	-13.802	-19.201	-39.12%	-41.450	-200.32%	**-14.097**	**-2.14%**
ethernet	P_{clk}	16.22	**11.47**	**29.28%**	12.94	20.22%	12.47	23.12%
	P_{total}	19.74	**15.03**	**23.86%**	16.56	16.11%	16.08	18.54%
	WNS	-2.608	-2.789	-6.94%	-2.943	-12.85%	**-2.380**	**8.74%**
	TNS	-273.830	-336.379	-22.84%	-352.083	-28.58%	**-260.849**	**4.74%**
vga_lcd	P_{clk}	12.52	**9.22**	**26.36%**	9.53	23.88%	9.29	25.80%
	P_{total}	15.70	13.80	12.10%	14.93	4.90%	**13.02**	**17.07%**
	WNS	-8.052	-16.796	-108.59%	-13.228	-64.28%	**-10.239**	**-27.16%**
	TNS	-367.614	-792.290	-115.52%	-610.260	-66.01%	**-456.827**	**-24.27%**
des_perf	P_{clk}	121.71	97.95	19.52%	100.73	17.24%	**89.22**	**26.69%**
	P_{total}	227.76	211.82	7.00%	208.64	8.39%	**192.33**	**15.56%**
	WNS	-0.271	-0.300	-10.70%	-0.259	4.43%	**-0.188**	**30.63%**
	TNS	-151.965	-171.134	-12.61%	-155.087	-2.05%	**-122.473**	**19.41%**
Average	P_{clk}	-	-	22.46%	-	22.92%	-	**24.15%**
	P_{total}	-	-	13.56%	-	13.50%	-	**16.16%**
	WNS	-	-	-22.41%	-	-27.50%	-	**5.13%**
	TNS	-	-	-39.06%	-	-48.16%	-	**8.10%**

optimization is executed. The second state-of-the-art MBFF clustering flow [19] is a pre-placement MBFF clustering algorithm. MBFF clustering is decided based on the netlist generated after synthesis by a commercial EDA tool. In the original flow proposed in [19], a threshold value needs to be adjusted through iterations until the design meets timing. However, in real-world cases, each iteration, including placement and clock tree synthesis, is time-consuming, making it impractical to go through all iterations. To ensure a fair comparison, only the result of the first iteration is included in the comparison. All the three flows for comparison enable the option for useful skew optimization before placement and routing.

The metrics we consider include the power of the clock tree (P_{clk}) in mW, the power of the total design (P_{total}) in mW, WNS and TNS in ns. All these metrics are collected at the post-routing stage using a commercial EDA tool. The percentages beside the metrics for each flow represent the percentage improvement compared to the flow without MBFF clustering, where a positive percentage indicates better performance than the flow without MBFF, and a negative percentage indicates worse performance. Based on the results in Table 4, compared with the flow without MBFF, our approach manages to save 24.15% clock tree power on average. Considering the total power consumption, our approach saves between 10.53% and 21.11%. Unlike the other two state-of-the-art MBFF clustering flows, our approach reduces TNS by up to 45.53% without negatively impacting the timing condition. This improvement is due to the special consideration of useful skew optimization in our proposed methodology.

When comparing our approach with the two state-of-the-art MBFF clustering flows, our proposed method reduces the total power by 3.01% and 3.08%, respectively. Additionally, WNS and TNS are improved by 22.50% and 33.91% on average compared to the commercial tool with MBFF clustering. These improvements increase to 25.59% and 37.97% when compared to the pre-placement MBFF clustering method proposed in [19].

3.3 Robustness against Delay Prediction Errors

To prove that our proposed approach is robust against the early stage path delay estimation errors, a separate experiment is conducted by scaling all the estimated path delays with randomly picked scaling factors. The scaling factor are randomly picked within the ranges [95%-105%], [90%-110%], [85%-115%], and [80%-120%]. Combined with the original results of our proposed approach without scaling, the five results of total power and absolute value of TNS are compared with results gets from [19], commercial tool with MBFF clustering, and flow without MBFF, as shown in Fig. 6. Since a design is considered to have better quality with lower absolute TNS and total power, a point closer to the bottom-left corner in figure represents a better result. For most designs, the quality of results from our proposed approach is better than the

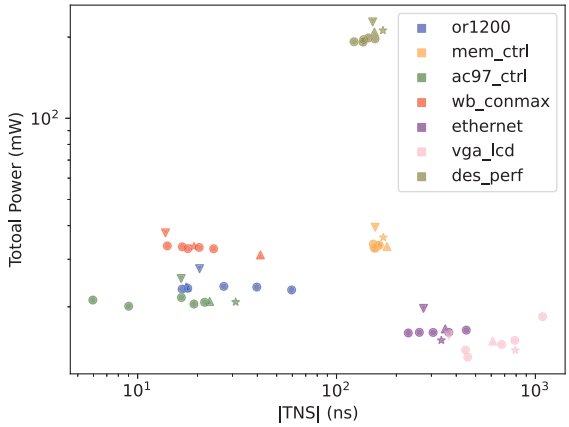


Figure 6: Total power vs. TNS (absolute value) for all benchmark circuits under various flows. Each color represents a benchmark circuit, and each symbol represents a flow. ▼: commercial tool without MBFF clustering, ★: commercial tool with MBFF clustering, ▲: pre-placement MBFF clustering [19], and ●: our proposed approach with randomly scaled path delays from ±0% to ±20%. Results closer to the bottom-left corner indicate better quality.

other flows, even when randomly picked scaling factors are applied to the predicted path delays. Consequently, our proposed approach does not necessarily depend on the chosen NLP-based timing estimator and can accommodate other path delay predictors with reasonable accuracy.

4 CONCLUSIONS & FUTURE WORK

In this paper, we proposed a novel approach for MBFF clustering that integrates early-stage physical and timing awareness to address the trade-off between clustering flexibility and design quality. The proposed method demonstrates improved timing conditions without compromising the power savings typically associated with MBFFs, achieving a better trade-off compared to methodologies that insert MBFFs either during placement or pre-placement. The evaluation of our approach using various benchmark circuits demonstrated significant improvements in power savings and timing metrics compared to both the commercial tool and state-of-the-art pre-placement MBFF clustering techniques. This methodology is portable and can be integrated with both commercial and academic design flows and tools. As for future work, we aim to scale this methodology up to support more variations of MBFFs, such as 4-bit and 8-bit, to further increase flexibility. Additionally, we plan to open-source this framework to benefit the community.

ACKNOWLEDGEMENT

This work is supported in part by the National Science and Technology Major Project (Grant No. 2021ZD0114701), the State Key Laboratory of Integrated Chips and Systems (SKLICS) open fund, and the SJTU Explore-X fund.

REFERENCES

[1] 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 (ISPD '24)*. Association for Computing Machinery, New York, NY, USA, 21–29. https://doi.org/10.1145/3626184.3633327

[2] J.P. Fishburn. 1990. Clock skew optimization. *IEEE Trans. Comput.* 39, 7 (July 1990), 945–951. https://doi.org/10.1109/12.55696

[3] IWLS. 2005. IWLS 2005 Benchmarks. https://iwls.org/iwls2005/benchmarks.html

[4] Iris Hui-Ru Jiang, Chih-Long Chang, Yu-Ming Yang, Evan Y.-W. Tsai, and Lancer S.-F. Chen. 2011. INTEGRA: fast multi-bit flip-flop clustering for clock power saving based on interval graphs. In *Proceedings of the 2011 international symposium on Physical design (ISPD '11)*. Association for Computing Machinery, New York, NY, USA, 115–122. https://doi.org/10.1145/1960397.1960424

[5] Hsu-Yu Kao, Chu-Han Hsu, and Shih-Hsu Huang. 2019. Two-Stage Multi-bit Flip-Flop Clustering with Useful Skew for Low Power. In *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*. 178–182. https://doi.org/10.1109/ICCET.2019.8726883

[6] Suwan Kim and Taewhan Kim. 2023. Design and Technology Co-Optimization for Useful Skew Scheduling on Multi-Bit Flip-Flops. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD57390.2023.10323866

[7] Mark Po-Hung Lin, Chih-Cheng Hsu, and Yao-Tsung Chang. 2011. Post-Placement Power Optimization With Multi-Bit Flip-Flops. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 12 (Dec. 2011), 1870–1882. https://doi.org/10.1109/TCAD.2011.2165716

[8] Mark Po-Hung Lin, Chih-Cheng Hsu, and Yu-Chuan Chen. 2015. Clock-Tree Aware Multibit Flip-Flop Generation During Placement for Power Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 2 (Feb. 2015), 280–292. https://doi.org/10.1109/TCAD.2014.2376988

[9] Hyoungseok Moon and Taewhan Kim. 2016. Design and allocation of loosely coupled multi-bit flip-flops for power reduction in post-placement optimization. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 268–273. https://doi.org/10.1109/ASPDAC.2016.7428022

[10] OpenCores. 2024. OpenCores. https://opencores.org/

[11] OpenRISC. 2024. OpenRISC 1200 implementation. https://github.com/openrisc/or1200

[12] Ya-Ting Shyu, Jai-Ming Lin, Chun-Po Huang, Cheng-Wu Lin, Ying-Zu Lin, and Soon-Jyh Chang. 2012. Effective and efficient approach for power reduction by using multi-bit flip-flops. *IEEE transactions on very large scale integration (vlsi) systems* 21, 4 (2012), 624–635.

[13] Chuan Yean Tan, Rickard Ewetz, and Cheng-Kok Koh. 2018. Clustering of flip-flops for useful-skew clock tree synthesis. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 507–512. https://doi.org/10.1109/ASPDAC.2018.8297374

[14] Chang-Cheng Tsai, Yiyu Shi, Guojie Luo, and Iris Hui-Ru Jiang. 2013. FF-bond: multi-bit flip-flop bonding at placement. In *Proceedings of the 2013 ACM International symposium on Physical Design (ISPD '13)*. Association for Computing Machinery, New York, NY, USA, 147–153. https://doi.org/10.1145/2451916.2451955

[15] 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 Transactions on Design Automation of Electronic Systems* 28, 5 (2023), 1–26.

[16] Shao-Huan Wang, Yu-Yi Liang, Tien-Yu Kuo, and Wai-Kei Mak. 2011. Power-driven flip-flop merging and relocation. In *Proceedings of the 2011 international symposium on Physical design (ISPD '11)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/1960397.1960423

[17] Jin-Tai Yan, Meng-Tian Chen, and Chia-Heng Yen. 2016. Cell-aware MBFF utilization for clock power reduction. In *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 648–651. https://doi.org/10.1109/ICECS.2016.7841285

[18] Jin-Tai Yan and Zhi-Wei Chen. 2010. Construction of constrained multi-bit flip-flops for clock power reduction. In *The 2010 International Conference on Green Circuits and Systems*. 675–678. https://doi.org/10.1109/ICGCS.2010.5542978

[19] Dongyoun Yi and Taewhan Kim. 2016. Allocation of multi-bit flip-flops in logic synthesis for power optimization. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–6. https://doi.org/10.1145/2966986.2966998

[20] Haisheng Zheng, Zhuolun He, Fangzhou Liu, Zehua Pei, and Bei Yu. 2024. LSTP: A Logic Synthesis Timing Predictor. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 728–733.

[21] Linyu Zhu and Xinfei Guo. 2023. Delay-Driven Physically-Aware Logic Synthesis with Informed Search. In *2023 IEEE 41st International Conference on Computer Design (ICCD)*. 327–335. https://doi.org/10.1109/ICCD58817.2023.00057