A Systematic Approach for Multi-objective Double-side Clock Tree Synthesis

Xun Jiang[1], Haoran Lu[1], Yuxuan Zhao[2], Jiarui Wang[1,3], Zizheng Guo[1], Heng Wu[1], Bei Yu[2], Sung Kyu Lim[4], Runsheng Wang[1,5,6], Ru Huang[1,5,6] and Yibo Lin[1,5,6*]
[1]School of IC, Peking University [2]Department of CSE, The Chinese University of Hong Kong
[3]School of CS, Peking University [4]School of ECE, Georgia Institute of Technology
[5]Institute of EDA, Peking University, Wuxi [6]Beijing Advanced Innovation Center for Integrated Circuits

***Abstract*—As the scaling of semiconductor devices nears its limits, utilizing the back-side space of silicon has emerged as a new trend for future integrated circuits. With intense interest, several works have hacked existing backend tools to explore the potential of synthesizing double-side clock trees via nano Through-Silicon-Vias (nTSVs). However, these works lack a systematic perspective on design resource allocation and multi-objective optimization. We propose a systematic approach to design clock trees with double-side metal layers, including hierarchical clock routing, concurrent buffers and nTSVs insertion, and skew refinement. Compared with the state-of-the-art (SOTA) methods, the widely-used open-source tool, our algorithm outperforms them in latency, skew, wirelength, and the number of buffers and nTSVs.**

I. Introduction

Back-side interconnection [1], [2] has emerged to continue the scaling of semiconductor technologies. With increasingly congested designs and tight timing budgets on the front-side (FS), both the academia [2]–[9] and industry [10], [11] have started to consider utilizing back-side (BS) resources for routing wires, including signal, power, and clock net. Based on the evaluation in [2], the latency of the clock tree is decreased from 50ps to 20ps with back-side metal layers. However, lacking a systematic double-side clock tree synthesis (CTS) algorithm is impeding further exploration of the potential of back-side resources, considering the new complex design space unfolding before researchers.

As shown in the bottom of Fig. 1, the double-side clock net is jointly implemented with connecting the back-side and front-side metal layers via additional nTSVs. Although timing benefits have been reported, the overhead of nTSVs, buffers, and clock wirelength cannot be neglected, as the primary objectives of CTS has included latency, skew, and resource consumption. For instance, many methods, e.g., clock routing [12]–[15], buffer insertion [16]–[19], useful skew [20]–[22], Flip-Flop (FF) clustering [23]–[25] and 3D clock tree [26]–[28] have all once referred to as timing and power optimization techniques involving clock trees. However, considering the challenges of more complex design resource allocation and multi-objective optimization, the exploration of double-side CTS is still insufficient to keep pace with advanced technology.

Existing works [2], [6], [7], [29] investigating double-side CTS all follow the incremental flow shown in the left of Fig. 1. To elaborate, Veloso et al. [2] flip the high-level part of a clock tree to the back-side to reduce clock latency to the maximum extent. Bethur et al. [7] propose leveraging the fanout of driven sinks as the criteria to decide whether the net should be flipped to back-side. Bethur et al. [6] utilize the Graph Neural Network to select the subset of FFs with poor timing performance and flip the connected nets to the back-side. Vanna-iampikul et al. [29] incorporate the back-side design methodologies of Power Delivery Network (PDN) based on the clock synthesis method by Bethur et al. [6]. Although the

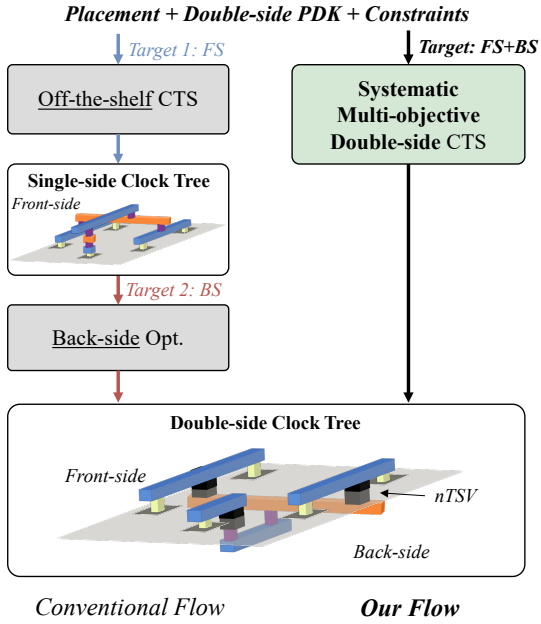


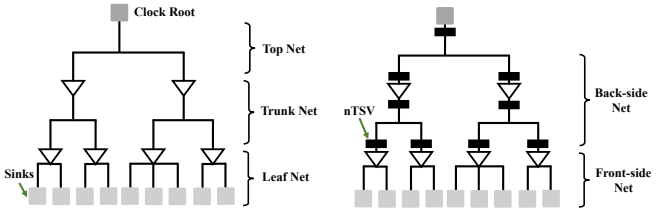
Fig. 1 Clock tree synthesis with double-side metal layers.

benefits of back-side metal layers to CTS are shown explicitly, the incremental flow cannot fully uncover their potential. For instance, the pre-generated single-side clock tree by the off-the-shelf CTS tool only considers the timing model based on the front-side technology parameters to guide the buffer insertion process, whose buffering solutions will deviate from the best one. The follow-up back-side optimization method has to obey these results to assign nets to back side by inserting nTSV, which can severely harm the eventual solution quality due to limited solution space. Thus, a unified design space of buffer and nTSV insertion with efficient solution exploration is urgent for double-side CTS, which is also the core of our work.

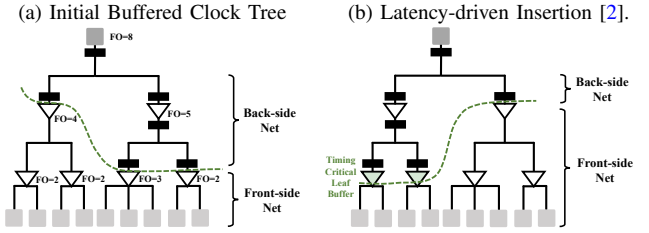
In this work, we aim at unleashing the potential of the back-side technology by handling the challenges from double-side CTS systematically. The major contributions of this work are as follows:

- We propose a systematic double-side clock tree synthesis framework aiming at pushing the boundary of the cutting-edge back-side technology exploration by multi-objective optimization.
- We propose an efficient hierarchical clock routing to reduce clock wirelength and preserve balanced structure.
- We propose a concurrent buffer and nTSV insertion based on multi-objective dynamic programming (DP) and an efficient resource-aware end-point buffer insertion as post-processing method for latency, skew, and resource usage optimization.
- We propose a novel methodology of design space exploration

*Corresponding author: Yibo Lin (yibolin@pku.edu.cn)



(a) Initial Buffered Clock Tree (b) Latency-driven Insertion [2].



(c) Fanout (FO)-driven Insertion [7]. (d) Critical Timing-driven Insertion [6].

Fig. 2 Comparison between the buffered clock tree and the double-side clock trees by different methods to assign top and trunk nets to the back side.

(DSE) of double-side CTS by our framework.

Experimental results demonstrate our superiority over recent works [2], [6], [7], [29]. Take the method [2] with extreme optimization on latency as an example, we can optimize the clock latency by 2.223×, skew by 2.464×, number of buffers by 1.010×, clock wirelength by 1.249×, and number of nTSVs by 1.441×, respectively, with 6.922× speed-up.

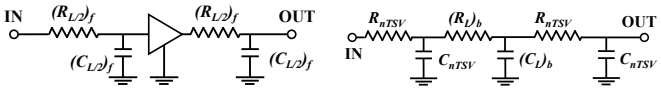
The rest of this paper is organized as follows. In Section II, we demonstrate the preliminaries of our work. In Section III, we explain the details of our algorithms. In Section IV, we present the results of our work. In Section V, we conclude our work and discuss further works.

II. Preliminaries

A. Double-side Clock Tree Structure

The comparison of structure between initial buffered clock tree and double-side clock trees are drawn in Fig. 2. In Fig. 2(a), the initial buffered clock tree assign all leaf nets and trunk nets are on the front side. The leaf net includes clock sink pins, while the trunk net encompasses all other nets excluding leaf nets. Top nets, defined by designers as highest-level trunk nets, can be distinguished from trunk nets for clarity.

We demonstrate three recent post-CTS methods to implement double-side clock tree in Fig. 2(b), Fig. 2(c), and Fig. 2(d), respectively. The post-CTS method from [2] assigns the trunk-level nets to the back-side by inserting nTSVs in Fig. 2(b). Owing to the input and output pins of the buffer on the front side, multiple nTSVs are incorporated to maintain connectivity between the front and back sides. The delay from source to sink pins can be reduced by the lower unit resistance and capacitance of the back-side metal layers, as most paths traverse these common trunk-level nets. It is noteworthy that extra nTSVs also increase the resource usage which needs to be carefully considered for the design of the entire chip. Therefore, [7] utilizes the fanout of driven sinks as the criteria to decide whether the nets should be flipped and [6] leverages the timing criticality of leaf driving buffers to decide the back-side nets assignment. In a word, these methods try to trade-off the timing benefits and the nTSV utilization but limited to the methodology of separated buffers and nTSVs insertion.



(a) Inserting Buffer (b) Inserting nTSV

Fig. 3 Delay modeling for buffer and nTSV insertion.

B. Delay Model for Buffers and nTSVs

In the clock tree synthesis with double-side metal layers, buffers and nTSVs are jointly utilized to minimize the latency of the clock tree. We follow previous work [2], [6], [7] to use the classic *L-type* Elmore delay [30] to compute the delay of wires. We set the front-side and back-side metal layer unit capacitance to c_f and c_b and set the front-side and back-side metal layer unit resistance to r_f and r_b. Furthermore, we take a wire segment with the length L and set the output driven capacitance of wire segment to C_d.

As buffer insertion on the front side in Fig. 3(a), we note C_b as the input capacitance of buffer and D_{buf} as the buffer delay. The delay of inserting a buffer at the middle of the wire is denoted as D_{bufOn}, which is computed as follows.

$$\begin{aligned} D_{bufOn} =& r_f \frac{L}{2}(c_f \frac{L}{2} + C_b) + D_{buf} + r_f \frac{L}{2}(c_f \frac{L}{2} + C_d) \\ =& \frac{r_f c_f}{2} L^2 + \frac{r_f (C_b + C_d)}{2} L + D_{buf}. \end{aligned} \tag{1}$$

We use C_{nTSV} and R_{nTSV} to represent the capacitance and resistance of one nTSV. The delay of inserting two nTSVs at the endpoints of the wire segment in Fig. 3(b) is denoted as D_{nTSVOn}, which is computed as follows.

$$\begin{aligned} D_{nTSVOn} =& R_{nTSV}(C_{nTSV} + C_d) + r_b L(c_b L + C_{nTSV} + C_d) \\ &+ R_{nTSV}(2C_{nTSV} + c_b L + C_d) \\ =& (r_b c_b) L^2 + (r_b C_{nTSV} + r_b C_d + R_{nTSV} c_b) L \\ &+ R_{nTSV}(3C_{nTSV} + 2C_d). \end{aligned} \tag{2}$$

In the double-side scenarios, $r_b c_b \ll r_f c_f$ reduce the delay of back-side metal layers. Meanwhile, the buffer can shield the output capacitance to reduce delay and meet maximum driven-capacitance constraint, whereas nTSV cannot. Therefore, the collaborative optimization of buffers and nTSVs insertion is crucial in the double-side CTS.

C. Multi-objective Optimization

Multi-objective optimization refers to solving problems with multiple conflicting objectives by exploring a set of optimal trade-off solutions. The Pareto frontier [31] is the core concept to represent these solutions, where no objective can be improved without degrading another. For the multi-objective optimization, it is essential to leverage Pareto frontier to comprehensively evaluate the performance of algorithms without being influenced by some parameter preferences. The diversity of solutions across the objective space is also important to avoid the algorithm getting stuck in local optimality.

D. Problem Formulation

Problem 1 (Double-side clock tree synthesis)**.** *Given the clock net, double-side metal layers, and candidate cells, e.g., nTSV and buffer, construct a double-side clock tree to optimize multiple objectives, e.g., latency, skew, wirelength, buffer count, and nTSV count, under connectivity and electricity constraints. The decision variables encompass the positions of inserted nTSVs and buffers, as well as their mutual topology relationships within the clock tree.*

=

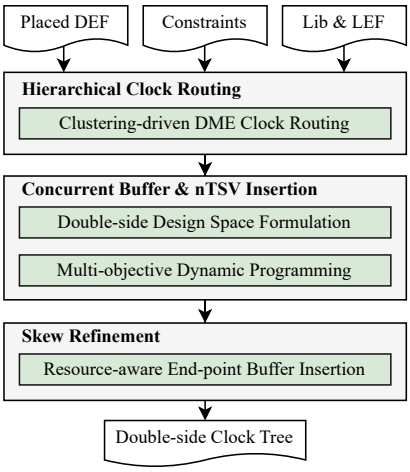


Fig. 4 Overview of our algorithm framework.

In this work, the key challenges are formulating unified double-side CTS design space and developing efficient multi-objective algorithm to explore Pareto-optimal solutions within this design space.

III. Algorithms

A. Overview

The overall flow of our algorithm is shown in Fig. 4. It takes placement results (Placed DEF), PDK, and capacitance and connectivity constraints as input, and output a legal clock tree with buffers and nTSVs. The algorithm mainly consists of three steps: hierarchical clock routing, concurrent buffer and nTSV insertion, and skew refinement (SR). Based on this algorithm, we also support design space exploration of the double-side CTS solutions. We explain each step in details in the following sections.

B. Hierarchical Clock Routing

In the modern CTS, the goal of clock routing is to firstly provide an initial clock tree topology that approximates the optimization of latency, skew, and wirelength. However, many follow-up timing-optimization stages, e.g., buffer insertion and sizing, make latency and skew more resilient to changes in topology, while the wirelength is still largely determined by the clock routing topology and impacts power significantly. We propose a hierarchical clock routing focuses on optimizing wirelength by combining clustering and deferred-merge-embedding (DME) clock routing.

In our hierarchical clock routing, the clustering is performed at two sequential steps, i.e., high-level clustering and low-level clustering, as shown in Fig. 5(a) and Fig. 5(b), respectively. High-level clustering groups the sinks into several large clusters with size H_c by minimizing the total intra-cluster wirelength approximately. Low-level clustering then divides each large cluster into smaller ones with size L_c. The purpose of the dual-level clustering is to obtain a hierarchy according to the spatial proximity of sinks. We also record the centroids of both high- and low-level clustering solutions for later steps. K-means algorithm is adopted as the backbone for both clustering steps. In the experiments, we set H_c to 3,000 and L_c to 30 empirically.

DME is widely used in many clock routing algorithms [13], [14]. It helps to minimize skew and wirelength efficiently. A typical DME is based on matching, as shown in Fig. 5(c). However, such an approach is reported to have poor wirelength when dealing with imbalanced distribution of sinks. Therefore, with the clustering

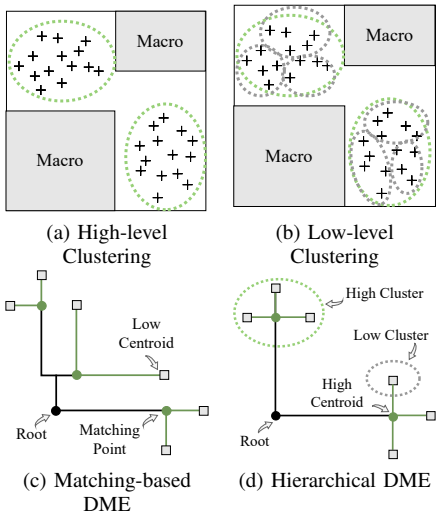


Fig. 5 Dual-level clustering and DME-based clock routing. Symbol "+" refers to sink.

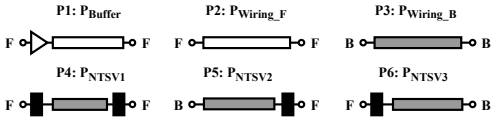


Fig. 6 Patterns of wire segments labeled as P1 $\sim$ P6. **F** refers to the front side. **B** refers to the back side. The **right end** is close to sinks and the **left end** is close to clock root.

results, we perform DME clock routing with the low-level clustering centroids as leafs and the corresponding high-level clustering centroids as root, denoted as hierarchical DME as shown in Fig. 5(d).

C. Concurrent Buffer and nTSV Insertion

In this section, we describe the details of double-side design space formulation and multi-objective DP.

1) Double-side Design Space Formulation: The double-side design space is formulated by the discrete edge patterns and connectivity constraints. Different from the traditional buffer insertion on the single side, the edge patterns in our algorithm should adapt to double sides and the characteristics of buffers and nTSVs. We list six candidate edge patterns, denoted as pattern set P, in the generated clock tree from hierarchical clock routing, as shown in Fig. 6. For instance, since the two pins of nTSVs are situated in different sides, resulting in side types of two endpoints of edge having distinct types as $\mathbf{P_{NTSV2}}$ and $\mathbf{P_{NTSV3}}$. However, the side types of two endpoints of $\mathbf{P_{NTSV1}}$ are still **F** due to two nTSVs flipping side twice. Meanwhile, since the two pins of buffers are located in the front side, the side types of two endpoints of edge have to be **F** as $\mathbf{P_{Buffer}}$. During buffers and nTSVs insertion process, the pattern decisions of adjacent edges cannot violate the **connectivity constraint**, that the shared vertex of any two edges in the clock tree must have the same side type.

2) Multi-objective Dynamic Programming: The multi-objective dynamic programming consists of four steps: build heterogeneous DP graph, bottom-up generation, multi-objective selection, and top-down decision, as shown in Fig. 7. With these steps, we can concurrently insert buffers and nTSVs into the clock tree efficiently.

Step 1 (Build Heterogeneous DP Tree): To build the heterogeneous DP graph, we firstly represent each edge of the clock tree by a node in the DP graph. Two adjacent edges in the clock tree

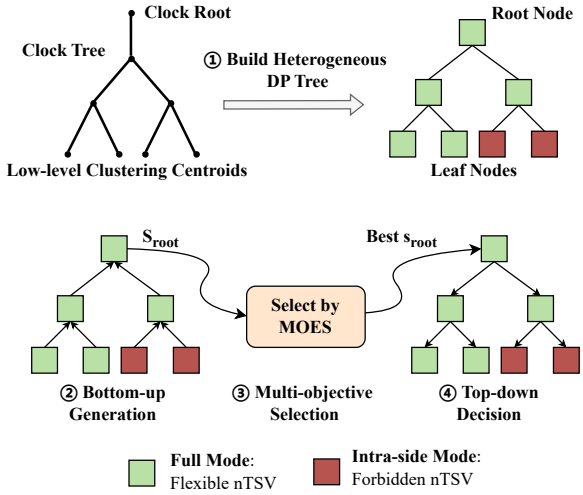


Fig. 7 The process of building DP formulation form clock tree and the bottom-up and top-down with DP execution. Each edge corresponds to two sets marked by different colors.

at different levels are connected by edge in DP graph as shown in Fig. 7. Due to the structure of clock tree, the DP graph is formed as a tree rooted by the node corresponding to the edge of clock root. Then, the pattern selection, i.e., buffer and nTSV insertion, can be conducted on the DP tree. Notice that the clock tree and DP tree are both binary tree.

A novel idea of ours is to further configure the nodes in DP graph with two types of nTSVs inserting mode: full mode (flexible nTSV with $P_1 \sim P_6$ allowed to select) and intra-side mode (forbidden nTSV with only $P_1 \sim P_3$ allowed to select), which could be easily implemented by restricting the allowed patterns on the edges in our framework. Thus, by the inserting mode configurations, we could obtain a heterogeneous DP tree. To support the DP algorithm to explore the solutions in the DP tree, we utilize S to represent the candidate solutions of each node and s to represent the selected final solution of each node.

Step 2 (Bottom-up Generation): In the process of bottom-up generation, we firstly set the undirected edges in DP tree by the topology of clock tree to directed edges, which all point to the root node finally. During the generation process, each node should undergo two operations: merging from two predecessor nodes and inserting in itself. We generated candidate solutions S for all nodes starting from the leaf nodes. For leaf nodes without predecessor nodes, the edge end-point close to sinks are forced to front-side, which restricts the initial insertion of leaf nodes to $\{\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_4, \mathrm{P}_5\}$ without merging. From the candidate leaf node solutions, we can generate the candidate solutions of successor nodes by merging each solution from one predecessor node with every solution from the other one. Meanwhile, these dependencies are recorded in merged solutions for fast traversal in Step 4. Notice that the merging operation is allowed which two predecessor solutions obeying the connectivity constraint. This rule can ensure we generate a legal double-side clock tree just by one turn of DP without any additional legalization steps, which improves the efficiency of our algorithm.

After the merging operations, each node should conduct the inserting operations to assign patterns based on the merged solutions. The patterns must be selected from the set P, while the selection could be restricted by the specific inserting mode. For instance, if one node is configured to the intra-side mode, the patterns

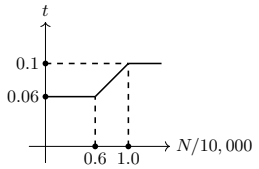


Fig. 8 Adaptive scale factor function $t \sim N/10,000$.

$\{\mathrm{P}_4, \mathrm{P}_5, \mathrm{P}_6\}$ involving nTSVs are forbidden to inserted in that node. By the delay model introduced in Section II-B and [16], we could calculate the effective capacitance and path delay for solutions after inserting patterns. By the iterative execution, we finally obtain the candidate solutions at the root node, denoted as S_{root} at Step 2 in Fig. 7.

Step 3 (Multi-objective Selection): With the generated candidate solutions S_{root}, we try to select the final solution $s_{root} \in S_{root}$ considering multi-objective optimization. With the efficient computation structure of DP, we could easily record the latency, buffer count, and nTSV count for each node during the bottom-up generation.

Thus, we propose a multi-objective enhancement score to approach the multi-objective optimization. With the additional nTSV as resource for insertion, the candidate solutions at the root node have many more combinations of buffers, nTSVs, and different latencies. The distribution of candidate solutions is much more diverse than that in the single-side buffer insertion scenario, as observed in IV-C in the experiment. Therefore, we utilize the multi-objective enhancement score (MOES) to decide the final solution s_{root} as follows.

$$\mathrm{MOES} = \alpha l_{root} + \beta b_{root} + \gamma n_{root}, \tag{3}$$

where l_{root}, b_{root}, and n_{root} are the values of latency, buffer count, and nTSV count for the candidate solutions at the root edge. α, β, and γ are manual parameters to weight each objective.

Step 4 (Top-down Decision): After the decision of s_{root}, we invert the direction of edges in the DP tree for top-down decision. By the recorded dependencies in the merged solutions at Step 2, we can quickly retrace the final solutions for all nodes.

Pruning technique: We extend the *inferior solution* concept in [16], that the effective capacitance and maximum path delay of one solution both worse than those of another solution means this solution will always be viewed as a bad candidate, to the double-side scenarios by pruning candidate solutions at front-side and back-side, respectively. This method ensures our DP algorithm is optimal in terms of latency. Meanwhile, to satisfy the constraints of the max-driven capacitance, we prune the solution with effective capacitance exceeding the maximum threshold.

D. Skew Refinement

As the DP in previous section mainly optimizes clock latency, we further introduce a resource-aware skew refinement technique to mitigate skew degradation by inserting buffers at end-points. This step is triggered when the skew is over $p\%$ of the maximum latency. In the experiments, we set p to 23. N refers to the number of sinks. t refers to an adaptive factor w.r.t. N, as shown in Fig. 8.

1) Set refined end-points number n as $\min(N \times t, m)$. m is the maximum number of refined end-points and set to 33 in the experiments.
2) Refine n end-points in descending order of delay by inserting one buffer at the low-level clustering centroids.

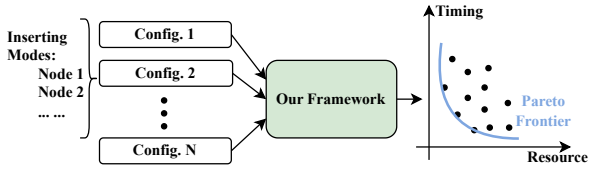


Fig. 9 Design space exploration for double-side CTS.

By the observation that the skew and wire delay within lower clusters have a negligible impact on the overall path, this method is efficient and effective in mitigating skew.

E. Design Space Exploration of Double-side CTS

Based on the concurrent buffer and nTSV insertion approach, we further propose a general double-side design space exploration methodology that demonstrates superiority in multi-objective optimization of double-side CTS. The main idea is to control the inserting modes of nodes in DP tree, as shown in Fig. 7. By setting up various configurations, users could explore more solutions in the objective space, as shown in Fig. 9.

To make the DSE process easy to control, we allow users to control the inserting modes of nodes in DP tree by setting a fanout threshold. Nodes with fanout lower than the threshold will be configured as full mode, while those with fanout larger than the threshold will only allow intra-side mode. Furthermore, more sophisticated methods to control the inserting modes could be incorporated into our framework other than the simple heuristics. The concept of decoupling the DP execution flow and the controlling of nodes inserting allows users to avoid dealing with cumbersome details, e.g., timing calculation, but still have large multi-objective optimization space, which will promote more optimization techniques in double-side CTS problem.

IV. Experimental Results

We perform the experiments on the Linux platform with a 20-core 2.40GHz Intel(R) Xeon(R) Silver 4210R CPU and 320GB memory. We take the ASAP7 PDK [32] to perform our experiments and adopt the unit resistance and capacitance of back-side metal layers and nTSV from [1]. These parameters are listed in TABLE I. We take designs from OpenROAD [33] and use its backend flow to generate benchmarks. The statistics of the benchmarks are listed in TABLE II. Our framework is implemented using C++.

A. Technology Settings

We follow OpenROAD's convention to take the unit resistance and capacitance of M3 for the evaluation of delays in the front-side. For the back-side wires, we compute delays according to the actual usage of layers (`BM1`∼`BM3`). We use the Elmore delay [30], the slew model [34], and the nonlinear delay model (NLDM) [32] for delay computation. In our work, we follow the default flow in OpenROAD where one kind of buffer is used. This is a reasonable setting, because buffer sizing will be further optimized for skew minimization in the follow-up clock tree optimization after clock tree synthesis in a real design flow [33], [35], [36]. We take `BUFx4_ASAP7_75t_R` with a shape of $0.378nm \times 0.27nm$ as the buffer and `nTSV` with a shape of $0.27nm \times 0.27nm$, which is aligned to the other standard cells in the layout. The resistance and capacitance of one `nTSV` are $0.020k\Omega$ and $0.004fF$.

TABLE I Layer resistances and capacitances [1], [32].

Layer	Unit Res. ($k\Omega/\mu m$)	Unit Cap. ($fF/\mu m$)
`M1`	0.138890	0.11368
`M2`	0.024222	0.13426
`M3`	0.024222	0.12918
`M4`	0.016778	0.11396
`M5`	0.014677	0.13323
`M6`	0.010371	0.11575
`M7`	0.009672	0.13293
`M8`	0.007431	0.11822
`M9`	0.006874	0.13497
`BM1`∼`BM3`	0.000384	0.116264

TABLE II The statistics of benchmarks [33].

ID	Design	Statistics		
		#Cells	#FFs	Util.
C1	`jpeg`	54973	4380	0.50
C2	`swerv_wrapper`	148407	14338	0.40
C3	`ethmac`	56851	10018	0.40
C4	`riscv32i`	11579	1056	0.50
C5	`aes`	29306	2072	0.50

B. Comparison with SOTA Methods

We utilize the open-source tool OpenROAD [33] to evaluate the effectiveness of back-side metal layers and the benefits of our algorithm. We generate the `post-place` and `post-cts` DEF files [37] from OpenROAD and use consistent evaluation methods and parameters for all designs. α, β, and γ are set to 1, 10, and 1. The fanout of [7] is set to 100 and the number of timing critical paths in [6] is set to 0.5. The inserting modes of all edges of our algorithm in TABLE III are set to full mode.

We also implement the method from [2] as the baseline, which extremely optimizes latency. We take the `post-cts` DEF generated by the CTS tools in OpenROAD without resizing operations following CTS. According to [2], we flipped the nets above the low clustering centroids to the back-side by inserting nTSV as illustrated in Fig. 2(b) to minimize latency, denoted as `OpenROAD Buffered Clock Tree` + [2] (`OpenROAD` + [2] for short) in TABLE III.

TABLE III summarizes the comparison between our framework and recent studies. Our algorithm outperforms the method [2] in the clock latency by 2.223×, skew by 2.464×, number of buffers by 1.010×, clock wirelength by 1.249×, and number of nTSVs by 1.441×. The significant reduction in latency and #nTSVs comes from our hierarchical clock routing and concurrent buffer and nTSV insertion. We also compare with recent studies [2], [6], [7] using the buffered clock tree generated by our framework, to demonstrate the effectiveness of the concurrent buffer and nTSV insertion. We can see that our algorithm could achieve better quality in almost all objectives. In addition, our algorithm is efficient. We achieve 6.992× speedup over `OpenROAD` + [2].

C. Effectiveness of MOES

In Fig. 10, we validate the effectiveness of MOES (Section III) in concurrent buffer and nTSV insertion compared with the solo buffered clock tree. The points show the best results achieved by using MOES and minimal latency deviate far away in the double-side scenario (see two triangles), while they are much closer in the single-side scenario (see two squares). The reason is that the double-side scenario enlarges the design space, where more combinations of buffers and nTSVs will be preserved at the end of DP and should be carefully considered. Therefore, an objective function considering holistic factors, like MOES, is necessary to improve the solution quality for double-side CTS.

TABLE III Comparison with recent studies on clock tree synthesis with back-side metal layers.

Design	OpenROAD Buffered Clock Tree†				OpenROAD Buffered Clock Tree + [2]						Ours					
	Latency (*ps*)	Skew (*ps*)	Buffers (#)	nTSVs (#)	Latency (*ps*)	Skew (*ps*)	Buffers (#)	Clk WL ($\times 10^6$)	nTSVs (#)	RT (*s*)	Latency (*ps*)	Skew (*ps*)	Buffers (#)	Clk WL ($\times 10^6$)	nTSVs (#)	RT‡ (*s*)
C1	246.154	37.189	**167**	0	172.027	40.171	**167**	4.768	189	4.351	**77.694**	**29.74**	172	**3.664**	**130**	**0.285**
C2	449.208	334.353	576	0	311.214	269.791	576	14.702	674	7.095	**123.355**	**58.632**	**571**	**11.394**	**499**	**1.693**
C3	214.629	25.953	375	0	159.982	20.929	375	6.019	487	4.470	**90.229**	**20.675**	375	**5.326**	**256**	**1.149**
C4	141.382	25.351	40	0	133.958	24.441	40	0.801	46	3.221	**54.296**	20.231	**33**	**0.623**	**24**	**0.038**
C5	213.993	75.261	79	0	192.928	78.307	79	1.723	89	3.436	**90.829**	**46.735**	**74**	**1.418**	73	**0.096**
Ratio	2.900	2.830	1.010	-	2.223	2.464	1.010	1.249	1.441	6.922	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Design	Our Buffered Clock Tree*				Our Buffered Clock Tree + [2]				Our Buffered Clock Tree + [7]				Our Buffered Clock Tree + [6]			
	Latency (*ps*)	Skew (*ps*)	Buffers (#)	nTSVs (#)	Latency (*ps*)	Skew (*ps*)	Buffers (#)	nTSVs (#)	Latency (*ps*)	Skew (*ps*)	Buffers (#)	nTSVs (#)	Latency (*ps*)	Skew (*ps*)	Buffers (#)	nTSVs (#)
C1	144.603	41.839	189	0	130.420	54.757	189	243	129.802	54.300	189	167	130.420	55.068	189	200
C2	273.704	70.390	588	0	244.554	117.383	588	739	244.505	117.334	588	551	244.554	117.383	588	576
C3	130.559	29.076	**366**	0	110.956	30.677	**366**	418	113.196	32.917	**366**	385	110.956	30.677	**366**	382
C4	71.089	**17.289**	42	0	65.768	28.930	42	50	65.768	28.930	42	35	65.768	28.930	42	47
C5	127.997	60.538	85	0	110.083	64.472	85	109	110.083	63.693	85	72	110.083	63.693	85	**66**
Ratio	1.714	1.245	1.037	-	1.516	1.683	1.037	1.588	1.520	1.688	1.037	1.232	1.516	1.680	1.037	1.294

† Use OpenROAD to generate buffered clock tree on front-side only. The Clk WL metric is the same as OpenROAD + [2] since the same clock topology is used.
* Use our framework to generate buffered clock tree on front-side only by three steps: conducting hierarchical clock routing, buffer insertion, and skew refinement. The Clk WL metric is the same as Ours since the same clock topology is used.
‡ The runtime for other methods are omitted due to space limit, as their runtime is either similar to Ours or to OpenROAD + [2], according to which algorithm used to generate the buffered clock tree.

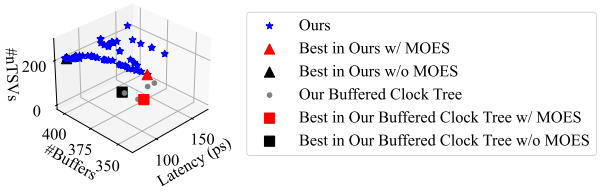


Fig. 10 The effectiveness of MOES with C3 (ethmac) under Ours and Our Buffered Clock Tree.

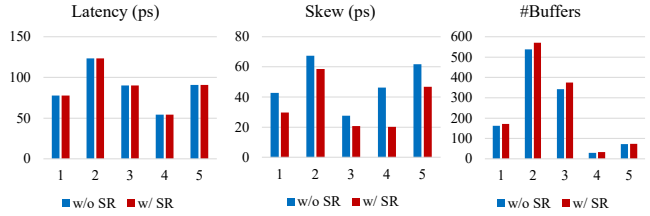


Fig. 11 Effectiveness of skew refinement.

D. Effectiveness of Skew Refinement

In Fig. 11, we demonstrate the effectiveness of skew refinement by the designs in TABLE II. In the skew refinement, we utilize the method in Section III-D to pick up the paths that need skew refinement. From Fig. 11, the skew can be reduced significantly, while the increases in latency and #buffers are very ignorable. This indicates that the method can effectively reduce skew as a post-processing step, which can also provide a better initial solution for the follow-up clock optimization stages.

E. Comparison on Design Space Exploration

To further verify the superiority of our systematic double-side framework, we perform design space exploration on different methods. We set the fanout ranging from 20 to 1000 with step 10 in our DSE flow as introduced in Section III-E. Meanwhile, we set the fanout of [7] also ranging from 20 to 1000 with step 10 and the percentage of critical paths of [6] ranging from 0.2 to 0.9 with step 0.05. To make the comparison fair, the buffered clock tree is all generated by our algorithm.

In Fig. 12, we compare our DSE flow with recent methods [2], [6], [7] in exploration of multi-objective solutions. We can see that the solution space of these methods are restricted to the buffered clock trees, and cannot effectively explore better latency or skew

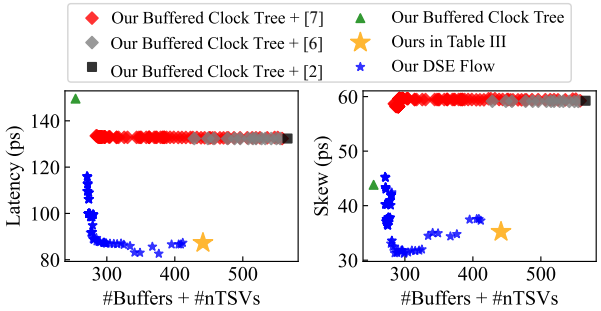


Fig. 12 The comparison of latency and skew of different flows.

even given more nTSVs. A possible reason is the target of buffered clock tree is to minimize clock latency by inserting buffers to shorten wirelength, while the ability of nTSV insertion will be weakened with the wirelength decreasing. Instead, our DSE flow controls the inserting modes in concurrent buffer and nTSV insertion allows much larger solution space. By simply sweeping the fanout threshold, we can find the Pareto frontier trading off latency, skew, buffers, and nTSVs. As our algorithm is very efficient, users can search for suitable solutions with our DSE flow at low cost.

V. Conclusion

In this work, we propose a systematic framework for clock tree synthesis with double-side metal layers and nTSVs. We propose hierarchical clock routing, concurrent buffer and nTSV insertion, and skew refinement to optimize clock tree in multiple objectives. Compared with the recent method [2] with extreme optimization on latency, we can optimize the clock latency by 2.223×, skew by 2.464×, number of buffers by 1.010×, clock wirelength by 1.249×, and number of nTSVs by 1.441×, respectively, with 6.922× speed-up. Meanwhile, our framework offers a DSE flow for multi-objective exploration to further boost the technology and design progressing. In the future, we will investigate placement and routing effects on double-side CTS and develop methodologies for full-flow optimization.

Acknowledgements

The project is supported in part by Grant QYJS-2023-2303-B, the Natural Science Foundation of Beijing, China (Grant No. Z230002), the 111 project (B18001), and Research Grants Council of Hong Kong SAR (No. RFS2425-4S02 and No. CUHK14211824).

References

[1] R. Chen, G. Sisto, A. Jourdain, G. Hiblot, M. Stucchi, N. Kakarla, B. Chehab, S. M. Salahuddin, F. Schleicher, A. Veloso *et al.*, "Design and optimization of sram macro and logic using backside interconnects at 2nm node," in *2021 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2021, pp. 22–4.

[2] A. Veloso, B. Vermeersch, R. Chen, P. Matagne, M. G. Bardon, G. Eneman, K. Serbulova, O. Zografos, S. Chen, G. Sisto *et al.*, "Backside power delivery: Game changer and key enabler of advanced logic scaling and new stco opportunities," in *2023 International Electron Devices Meeting (IEDM)*. IEEE, 2023, pp. 1–4.

[3] D. Prasad, S. T. Nibhanupudi, S. Das, O. Zografos, B. Chehab, S. Sarkar, R. Baert, A. Robinson, A. Gupta, A. Spessot *et al.*, "Buried power rails and back-side power grids: Arm® cpu power delivery network design beyond 5nm," in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 19–1.

[4] T.-C. Lin, F.-Y. Hsu, W.-K. Mak, and T.-C. Wang, "An effective netlist planning approach for double-sided signal routing," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 288–293.

[5] F.-Y. Hsu, T.-C. Lin, W.-K. Mak, and T.-C. Wang, "A bounding box-based net partitioning method for double-sided routing," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, 2024, pp. 397–402.

[6] N. E. Bethur, P. Vanna-Iampikul, O. Zografos, L. Zhu, G. Sisto, D. Milojevic, A. García-Ortiz, G. Hellings, J. Ryckaert, F. Catthoor *et al.*, "Gnn-assisted back-side clock routing methodology for advance technologies," *Proceedings of the 61st Design Automation Conference*, 2024.

[7] N. E. Bethur, "A methodology for back-side clock delivery network design compatible with commercial eda flows," Master's thesis, Georgia Institute of Technology, 2023.

[8] M. M. S. Aly, T. F. Wu, A. Bartolo, Y. H. Malviya, W. Hwang, G. Hills, I. Markov, M. Wootters, M. M. Shulaker, H.-S. P. Wong *et al.*, "The n3xt approach to energy-efficient abundant-data computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 19–48, 2018.

[9] H. Lu, Y. Ge, X. Jiang, J. Sun, W. Peng, R. Guo, M. Li, Y. Lin, R. Wang, H. Wu *et al.*, "First experimental demonstration of self-aligned flip fet (ffet): A breakthrough stacked transistor technology with 2.5 t design, dual-side active and interconnects," in *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2024, pp. 1–2.

[10] M. Shamanna, E. Abuayob, G. Aenuganti, C. Alvares, J. Antony, A. Bahudhanam, A. Chandran, P. Chew, A. Chatterjee, B. Chauhan *et al.*, "E-core implementation in intel 4 with powervia (backside power) technology," in *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2023, pp. 1–2.

[11] "Tsmc a16 technology," https://www.tsmc.com/english/dedicatedFoundry/technology/logic/l_A16.

[12] W. Wang, V. F. Pavlidis, and Y. Cheng, "Zero-skew clock network synthesis for monolithic 3d ics with minimum wirelength," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 399–404.

[13] K. D. Boese and A. B. Kahng, "Zero-skew clock routing trees with minimum wirelength," in *[1992] Proceedings. Fifth Annual IEEE International ASIC Conference and Exhibit*. IEEE, 1992, pp. 17–21.

[14] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *Proceedings of the 30th international Design Automation Conference*, 1993, pp. 612–616.

[15] W. Li, Z. Huang, B. Yu, W. Zhu, and X. Li, "Toward controllable hierarchical clock tree synthesis with skew-latency-load tree," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[16] L. P. Van Ginneken, "Buffer placement in distributed rc-tree networks for minimal elmore delay," in *1990 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1990, pp. 865–868.

[17] K. Han, A. B. Kahng, and J. Li, "Optimal generalized h-tree topology and buffering for high-performance and low-power clock distribution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 478–491, 2018.

[18] M. R. Guthaus, G. Wilke, and R. Reis, "Non-uniform clock mesh optimization with linear programming buffer insertion," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 74–79.

[19] Y.-Y. Chen, C. Dong, and D. Chen, "Clock tree synthesis under aggressive buffer insertion," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 86–89.

[20] J. G. Xi and W. W.-M. Dai, "Useful-skew clock routing with gate sizing for low power design," in *Proceedings of the 33rd annual Design Automation Conference*, 1996, pp. 383–388.

[21] W. Shen, Y. Cai, W. Chen, Y. Lu, Q. Zhou, and J. Hu, "Useful clock skew optimization under a multi-corner multi-mode design framework," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2010, pp. 62–68.

[22] N. Uysal, W.-H. Liu, and R. Ewetz, "Latency constraint guided buffer sizing and layer assignment for clock trees with useful skew," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 761–766.

[23] C. Deng, Y.-C. Cai, and Q. Zhou, "Register clustering methodology for low power clock tree synthesis," *Journal of Computer Science and Technology*, vol. 30, no. 2, pp. 391–403, 2015.

[24] A. D. Mehta, Y.-P. Chen, N. Menezes, D. Wong, and L. Pilegg, "Clustering and load balancing for buffered clock tree synthesis," in *Proceedings International Conference on Computer Design VLSI in Computers and Processors*. IEEE, 1997, pp. 217–223.

[25] S. Bang, K. Han, A. B. Kahng, and V. Srinivas, "Clock clustering and io optimization for 3d integration," in *2015 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*. IEEE, 2015, pp. 1–8.

[26] T.-Y. Kim and T. Kim, "Clock tree synthesis for tsv-based 3d ic designs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 4, pp. 1–21, 2011.

[27] J.-S. Yang, J. Pak, X. Zhao, S. K. Lim, and D. Z. Pan, "Robust clock tree synthesis with timing yield optimization for 3d-ics," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. IEEE, 2011, pp. 621–626.

[28] S. Pentapati, J. Lee, Y. S. Yu, S. K. Lim *et al.*, "Tier partitioning and flip-flop relocation methods for clock trees in monolithic 3d ics," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2019, pp. 1–6.

[29] P. Vanna-iampikul, H. Yang, J. Kwak, J. X. Hu, A. Rahman, N. E. Bethur, C. Hao, S. Yu, and S. K. Lim, "Back-side design methodology for power delivery network and clock routing," in *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2024, pp. 1–2.

[30] J. Hu, G. Schaeffer, and V. Garg, "Tau 2015 contest on incremental timing analysis," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 882–889.

[31] T. Marler and J. S. Arora, "Multi objective optimization: concepts and methods for engineering," 2009.

[32] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[33] "Openroad," https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.

[34] C. Sitik, W. Liu, B. Taskin, and E. Salman, "Design methodology for voltage-scaled clock distribution networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3080–3093, 2016.

[35] "Cadence Innovus Implementation System," http://www.cadence.com.

[36] R. Ewetz, "A clock tree optimization framework with predictable timing quality," in *Proc. DAC*, 2017, pp. 1–6.

[37] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin, and M. Woo, "Datc rdf-2019: Towards a complete academic reference design flow," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–6.