An Effective and Efficient Cross-Link Insertion for Non-Tree Clock Network Synthesis

Jinghao Ding[1 2†], Jiazhi Wen[1†], Hao Tang[1], Zhaoqi Fu[1], Mengshi Gong[1†], Yuanrui Qi[1], Wenxin Yu[1†*], Jinjia Zhou[2]
[1]Southwest University of Science and Technology, Mianyang, Sichuan, China
[2]Hosei University, Tokyo, Japan
yuwenxin@swust.edu.cn

***Abstract*—Clock skew introduces significant challenge to the overall system performance. Existing non-tree solutions like cross-link insertion often come with limitations, such as the over-consumption of resource and power. In this work, we propose a cross-link insertion algorithm that effectively reduces the clock skew with minimal power overhead, and prioritize delay optimization on the paths with high sensitivity to the skew. The experimental results from the ISPD 2010 benchmarks show a 17% reduction in the mean of clock skew, a 45% decrease in the standard deviation of clock skew, and a 13% lower power consumption versus the advanced non-tree solutions in literature.**

***Index Terms*—Clock Network Synthesis, Cross-Link Insertion**

I. Introduction

Clock skew, the difference in arrival time of clock signals at various flip-flops, is a critical concern in high-performance synchronous digital systems. Clock skew is volatile due to process, voltage, and temperature (PVT) variations [1] [2]. Excessive clock skew could potentially violate the predefined timing constraints, degrade the operational frequency, and significantly impact the overall system behavior [3].

Research and engineering attempts have been made to explore skew sensitivity towards various redundancy structures. Non-tree topologies of clock signal distribution network, such as cross-link insertion and mesh structure [4] [5], are able to enhance the resistance towards PVT variations by introducing redundant paths. Clock mesh has high sensitivity tolerance but consumes huge amount of power due to the symmetry of redundancy [4]. Cross-link enables signal delivery between two topologically distant nodes in the clock tree. Significant improvements in robustness can be achieved by inserting a small number of cross-links into a clock tree, with negligible overhead on power consumption [6].

Cross-link insertion approach for unbuffered clock trees is proposed in [6] [7]. Extensive methods for inserting cross-links into buffered clock trees are explored in [8] and [9]. However, most of these works are mainly focused on leaf-level link connection, which limits the design flexibility and improvement space. T. Mittal et al [5] inserts cross-links between internal nodes in a clock tree to enhance the correlation of delays for sinks with similar path lengths. D.-J. Lee et al [10] employs the concept of a multi-level fusion tree to insert redundant paths between multiple points, where extra clock trees are utilized to connect critical locations and fused with the original tree. Despite robustness improvement, the overhead of power and cost is noticeable.

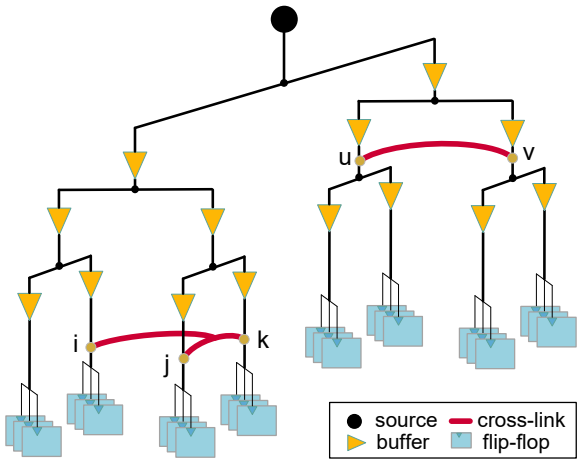

Fig. 1. An example of clock tree with cross-link insertion.

This paper proposes a novel cross-link insertion approach to improve robustness of clock network. The main contributions of this work are as follows.

- We propose two clustering algorithms to partitioning sinks based on their physical locations, timing constraints and capacitance. This ensures efficient and balanced interconnects within the clock tree.
- We employ a delay sensitivity vector to assess the susceptibility among different sinks and guide the insertion of cross-links between subtrees.
- We introduce a level determination algorithm for inserting cross-links, which is a crucial step to select appropriate locations of cross-links.
- We develop a new cross-link structure by applying Steiner tree topology between multiple subtrees, in order to enhance the robustness of the clock network.
- The experimental results validate the effectiveness of our proposed algorithm with 17% reduction in the mean of

†Equal contribution.
*Corresponding author.

978-3-9826741-0-0/DATE25/© 2025 EDAA

clock skew, 45% standard variance improvement and 13% lower power consumption.

The rest of this paper is structured as follows: Section II presents the problem formulation. Section III elaborates on our proposed methodology. Section IV discusses the results of our experiments. Finally, Section V concludes our work.

II. Preliminaries

A. Link-Based Method

The incorporation of cross-links in the clock tree offers redundant paths without disrupting its topology. Figure 1 is an example of the cross-link insertion, as outlined in [6], the insertion of cross-links between nodes u and v within the clock tree alters the clock skew between these nodes according to the following equation:

$$\hat{q}_{u,v} = \alpha q_{u,v} + \alpha\beta \tag{1}$$

where $\alpha = \dfrac{R_l}{R_l + d_u - d_v}$, and $\beta = \frac{C_l}{2}(R_{u,u} - R_{v,v})$. The term $q_{u,v}$ represents the original skew between nodes u and v. The variables d_u and d_v denote the Elmore delay of nodes u and v, respectively. Additionally, C_l and R_l stand for link capacitance and resistance, while $R_{u,u}$ and $R_{v,v}$ correspond to the transfer resistance of nodes u and v.

The insertion of cross-links emerges as a preferred strategy for mitigating variations in clock skew, primarily due to its provision of redundant paths towards the sinks. Even if one of these paths experiences significant variations, adherence to skew constraints remains achievable through the utilization of redundant paths.

B. Timing Constraints

A clock network connects the clock source to a set of clock sinks $S = \{s_1, s_2, ..., s_n\}$. In this network, a pair of sinks are considered *sequentially adjacent* if there exists combinational logic between them. It is necessary to satisfy setup and hold time constraints for each sequentially adjacent sinks. These time constraints can be expressed as follows.

$$\begin{cases} t_i + t_i^{CQ} + t_{ij}^{max} + t_j^S \le t_j + T \\ t_i + t_i^{CQ} + t_{ij}^{min} \ge t_j + t_j^H \end{cases} \tag{2}$$

where t_i and t_j represent the arrival times of the clock signal at s_i and s_j, respectively. t_{ij}^{min} and t_{ij}^{max} denote the minimum and maximum propagation delays through the combinational logic between these sinks. t_i^{CQ} stands for the clock-to-output delay of s_i. T symbolizes the clock period, while t_j^S and t_j^H refer to the setup and hold times of s_j, respectively. The skew between the i^{th} and j^{th} sinks are constrained by lower bound $l_{i,j}$ and upper bound $u_{i,j}$ as below.

$$l_{i,j} \le skew_{i,j} \le u_{i,j} \tag{3}$$

III. Methodology

Figure 2 illustrates the flow of our algorithm, which comprises several steps. During the initial construction phase of the clock network, we build a clock tree that approximately adheres to the corresponding skew and slew constraints. To achieve this, we employ a two-phase construction approach known as DME [11], involving both bottom-up and top-down phases. In the initial phase, a buffered clock tree is created. However, its limited robustness leaves it susceptible to PVT variations, which could potentially degrade the performance of the clock network. Thus, maintaining balanced cross-links without excessive wirelength becomes crucial. The following stages involve partitioning sink clusters and selecting critical sink pairs for cross-link insertion. Finally, cross-links are constructed to enhance the robustness of the clock network.

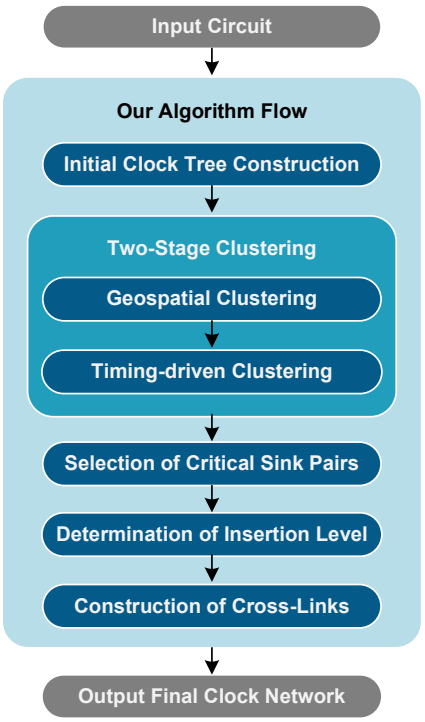

Fig. 2. Flow of our cross-link insertion algorithm.

A. Initial Clock Tree Construction

1) Obstacle-Avoiding Clock Tree: We start by constructing an initial tree using a ZST/DME algorithm [11] and then adjust it to avoid obstacles. Our approach takes into account rectangular obstacles in the layout. In the obstacle-avoiding clock tree, paths from parent to child nodes need to navigate around obstacles that intersect their shortest routes. Initially, we identify all wires intersecting obstacles. For each point-to-point connection, we use the shortest-path maze routing to navigate around obstacles. However, avoiding obstacles may lead to unbalanced wire routing, causing clock skew. To balance delay, we introduce wire snaking on the other branch.

2) Buffer Insertion: To meet slew constraints, buffers are usually necessary. Adjustments to buffer distribution can be made based on the constructed initial clock tree. We employ a bottom-up approach, beginning from the clock sink, to insert buffers of same type and size level-by-level. Traversing the tree edges, a buffer is inserted to a branch when its slew is nearing

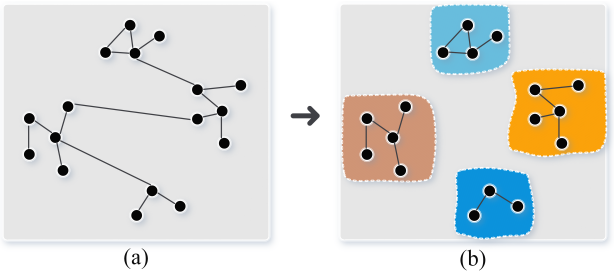

Fig. 3. An example of Geospatial-driven clustering. There are 17 sinks divided into 4 clusters based on capacitance and distance between sinks. (a) shows an undirected timing graph based on the timing constraints among sinks, while (b) represents a set of geospatial clusters.

the limit. Slew is approximated by visible load capacitance, where the visible capacitance refers to the capacitance shielded by inserted buffers and seen by the upstream circuit.

B. Geospatial-driven Clustering

The distribution of sinks significantly influence the propagation delay of the clock signal, which plays a critical role in both circuit robustness and performance. Sinks with timing constraints are often in close proximity. In the initial stage, we introduce a geospatial-driven clustering algorithm to assign sequentially adjacent sinks to the same cluster, thereby facilitating subsequent cross-link insertion to achieve better balance and efficiency.

Throughout the clustering process, we consider not only the timing constraints and distances among sinks but also the total capacitance of each cluster. This helps alleviate the negative impact of longer interconnects caused by cross-links. We use Cap_{max}, as introduced in [12], serving as the total capacitance for each cluster. Cap_{max} is defined as follows:

$$Cap_{max} = \lambda \sum_{i=1}^{N} cap_i + \eta \sqrt{N} D \tag{4}$$

where D represents the diameter of the input sink set, defined as the Manhattan distance between the two farthest sinks in the set; N denotes the number of sinks, while cap_i is the input capacitance of the i^{th} sink. λ and η are scaling parameters.

The distance constraint ensures that the Manhattan distance between the centroid of the current cluster and sinks outside the cluster does not exceed the predefined threshold $L_{\max}$, which is defined as follows:

$$L_{max} = \gamma \sqrt{\frac{A}{N}} \tag{5}$$

where A denotes the area of the bounding box comprising all sinks, and γ is a user-specified parameter utilized for scaling L_{max}. The detailed process is described in Algorithm 1.

Initially, we construct an undirected timing graph based on the timing constraints among sinks, as illustrated in Figure 3(a). The undirected edges represent logical dependencies between sinks (Line 2). Subsequently, starting randomly from an unassigned sink p_i, we perform a breadth-first search algorithm (BFS) on p_i based on this graph (Line 8). For each search result

Algorithm 1 Geospatial-driven Clustering

Input: Sinks P, skew constraints S, capacitance threshold Cap_{max}
Output: Geospatial clusters C_{geo}

1: $L_{max} \leftarrow CalculateMaxL()$
2: Construct undirected timing graph G based on S
3: $C_{geo} \leftarrow \{\}$
4: $P_{unassigned} \leftarrow P$
5: Select $p_i \in P_{unassigned}$ as starting point
6: Create new cluster $c \leftarrow \{\} \cup p_i$
7: **while** $P_{unassigned} \neq \phi$ **do**
8: $\quad p_j \in BreadthFirstSearch(p_i, G, P_{unassigned})$
9: $\quad cap \leftarrow CalculateCap(c)$
10: $\quad L \leftarrow GetDistance(c, p_j)$
11: $\quad$ **if** $cap \leq Cap_{max}$ **then**
12: $\quad\quad$ **if** $L \leq L_{max}$ **then**
13: $\quad\quad\quad c \leftarrow c \cup p_j$
14: $\quad\quad\quad P_{unassigned} \leftarrow P_{unassigned} - p_j$
15: $\quad\quad$ **end if**
16: $\quad$ **else**
17: $\quad\quad C_{geo} \leftarrow c \cup C_{geo}$
18: $\quad\quad c \leftarrow \{\}$
19: $\quad\quad$ Select $p_i \in P_{unassigned}$ as starting point
20: $\quad$ **end if**
21: **end while**
22: **return** C_{geo}

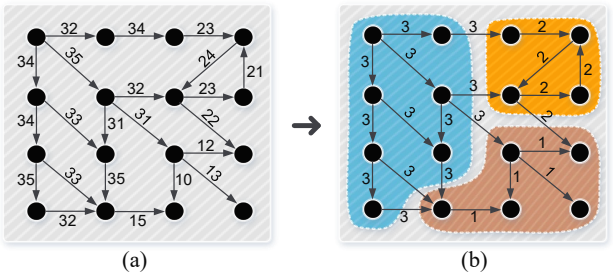

Fig. 4. An example of timing-driven clustering. After normalization, the set of edge weights {10, 12, 13, 15, 21, 22, 23, 24, 32, 33, 34, 35} was mapped to set {1, 2, 3}, then a geospatial cluster was divided into 3 timing clusters.

p_j, if capacitance of the current cluster meets the constraint Cap_{max}, we then check whether it satisfies the distance constraint L_{max}. If p_j meets both constraints, it is assigned to the current cluster (Lines 11-15); otherwise, we proceed to evaluate the next search result. Each assigned sink will be removed from the unassigned set $P_{unassigned}$ (Line 14). Additionally, if the current cluster fails to meet the capacitance constraint, it means the end of clustering for that cluster. We then randomly select another unassigned sink as the starting point for the next cluster and iterate until $P_{unassigned}$ is empty, indicating all sinks have been assigned to their respective clusters (Lines 16-20).

After traversing all sinks, we obtain a set of geospatial clusters where the load capacitance within each cluster and the distances between sinks are relatively balanced. Figure 3

illustrates an example result of the geospatial-driven clustering process.

C. Timing-driven Clustering

In general, paths with stricter timing constraints impose higher demands for cross-links maintain stable functionality. To implement an effective insertion approach, our second stage further organizes sinks based on the skew constraints among them, as illustrated in Algorithm 2.

For each geospatial cluster, unlike in geospatial-driven clustering, a directed timing graph is constructed based on all sinks in the cluster. We assign weights to the edges in the directed sequence graph based on the skew constraint between sinks, where the weight value is defined as the skew constraint between the two vertices of the edge, as illustrated in Figure 4(a).

Furthermore, we normalize the weights of edges with similar clock skew constraints to the same value according to Equation (6):

$$\hat{w} = \left\lfloor 1 + \frac{w - w_{min}}{w_{max} - w_{min}} \times l \right\rfloor \tag{6}$$

where w is the value of the edge before normalization, w_{min} and w_{max} are the minimum weight and maximum weight among edges respectively. l is the maximum level that cross-links can be inserted. By normalization, irregular skew constraint values can be mapped to a small range of uniform weights (Line 2), which will facilitate the determination of subsequent cross-link insertion levels. In the iterative clustering process (Lines 5–11), for each geospatial cluster $c_k \in C_{geo}$, the algorithm processes each sink individually. If a sink connects to multiple edges with different weights after normalization, the edge with the smallest weight is selected to determine the most appropriate timing cluster c_{index}, ensuring compliance with the strictest timing constraints. These decisions dynamically update the temporary clusters C_{temp}, which are consolidated into the final timing clusters C_{timing} once all sinks in the geospatial cluster have been processed.

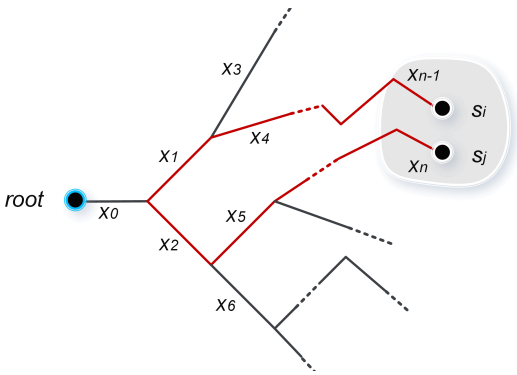

Fig. 5. Clock tree with variations on each segment. Nodes s_i and s_j are a critical sink pair after computing sensitivity of segments in the clock tree.

Algorithm 2 Timing-driven Clustering

Input: Geospatial clusters C_{geo}, skew constraints S
Output: Timing clusters C_{timing}

1: Construct directed timing graph G based on S
2: $Normalize(S)$
3: Initialize $C_{timing} \leftarrow \{\}$
4: **for all** $c_k \in C_{geo}$ **do**
5: Initialize $C_{temp} \leftarrow \{\}$
6: **for all** sink $s_i \in c_k$ **do**
7: $index = MinTimingEdge(s_i)$
8: **if** c_{index} not exits **then**
9: $c_{index} \leftarrow \{\}$
10: **end if**
11: $c_{index} \leftarrow c_{index} \cup s_i$
12: $update(C_{temp}, c_{index})$
13: **end for**
14: $C_{timing} \leftarrow C_{timing} \cup C_{temp}$
15: **end for**
16: **return** C_{timing}

D. Selection of Critical Sink Pairs

After completing clock tree construction and clustering, we analyze the impact of variation on skew between sequentially adjacent sink pairs. Inspired by [13], we calculate a sensitivity vector for these sink pairs to identify critical pairs not robust enough within given timing constraints. Higher sensitivity suggests nodes between critical sink pairs are more susceptible, indicating potential insertion of a link between them for maximum beneficial impact. The delay can be represented as follows:

$$T_i = f_i(x_0, x_1, x_2, ..., x_n) \tag{7}$$

where T_i represents the delay of sink s_i. The segment x_i denotes process factors affecting clock tree delay as shown in Figure 5. These data are obtained from the simulation results.

The skew variation magnitude between two nodes is expressed as follows:

$$M_{i,j} = \sqrt{\sum_{k=0}^{n} (\frac{\partial T_i}{\partial x_k} - \frac{\partial T_j}{\partial x_k})^2} \tag{8}$$

where $\frac{\partial T_i}{\partial x_k} - \frac{\partial T_j}{\partial x_k}$ is the sensitivity vector for skew. The process is detailed in Algorithm 3. We compute delay sensitivity for each segment and skew sensitivity for all sink pairs using Equation (8) (Lines 3-9). M_{ij} determines if inserting a cross-link between the paths is necessary. Sink pairs with sensitivity exceeding a user-specified threshold M_{user} are added to the critical sink pair set (Lines 10-14). Cross-links will then be inserted between these paths. Within critical sink pairs, sinks may be physically close but topologically distant, thus providing many potential insertion locations for cross-links.

E. Determination of Insertion Level

Determining the level for cross-link insertion is crucial after identifying the paths for insertion nodes. Inserting cross-links

Algorithm 3 Critical Sink Pair Selection

Input: Timing Clusters $c_p \in C_{timing}$
Output: Critical Sink Pair Set $P_{critical}$

Initialize Sensitive Vector $V \leftarrow \{\}$
Initialize $P_{critical} \leftarrow \{\}$
for all sink pair s_i and $s_j \in c_p$ **do**
  for all segment $x_i \in$ clock tree T **do**
    $CalculateDelaySensitivity(x_i)$
    $M_{ij} \leftarrow GetSkewSensitivity(p_i, p_j)$
    $update(V, M_{ij})$
  end for
end for
for all $M_{ij} \in V$ **do**
  if $M_{ij} > M_{user}$ **then**
    $P_{critical} = P_{critical} \cup (s_i\ ,\ s_j)$
  end if
end for
return $P_{critical}$

closer to sink generate better results. However, the closer the insertion node to the sink, the greater the number of cross-links required, potentially leading to a significant increase in power consumption. Balancing increased robustness with reduced power consumption poses a trade-off. Our approach considers not only leaf-level cross-links but also higher-level clock edge in selecting insertion nodes.

We determine the insertion level for cross-links based on the magnitude of skew constraints. Equation (9) is used to determine the insertion level.

$$L = min(\hat{w}, LCA) \tag{9}$$

where L represents the level to be inserted, $\hat{w}$ denotes the normalized level value after timing-driven clustering. If this value exceeds the level of the lowest common ancestor (LCA) of the critical sink pair, the cross-link is inserted below the LCA. For example, in Figure 6, if the normalized skew constraint value for a critical sink pair is 2, the cross-link is inserted at the second level.

Once critical sink pairs are identified, we proceed to locate insertion nodes based on the cross-link insertion level within their respective paths. The strategic insertion of cross-links at locations with similar delays has proven most effective [14]. We traverse their paths from bottom to top, searching for points with nearly equivalent delays as potential insertion nodes for cross-link construction.

F. Construction of Multi-pin Cross-Links

Different from traditional strategies that directly construct two-pin cross-links, resulting in a large number of cross-links, we propose a novel cross-link structure. This structure aims to strike a balance between enhancing robustness and minimizing power consumption by strategically inserting cross-links at suitable levels.

The concept of Steiner tree [15] in graph theory refers to the minimum-cost tree connecting a specific subset of nodes within a given graph. We adopt a Steiner tree as our fundamental structure. After determining all insertion nodes within a cluster, we construct Steiner trees among the nodes of each level to be inserted. This approach not only connects critical sink pairs at a reduced cost but also adds paths between sequentially adjacent sinks within the cluster, thereby enhancing the robustness of clock networks.

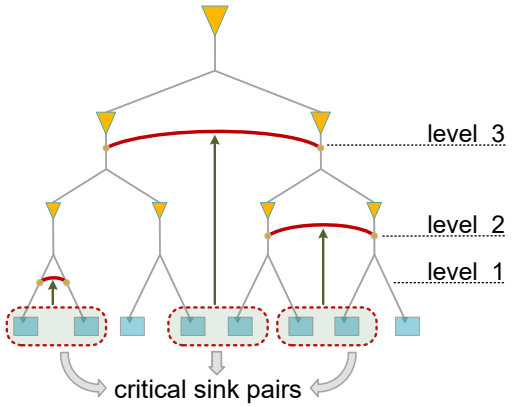

Fig. 6. The level at which the cross-link is inserted. For critical sink pairs, the normalized values of their skew determine which level the cross-link is inserted in.

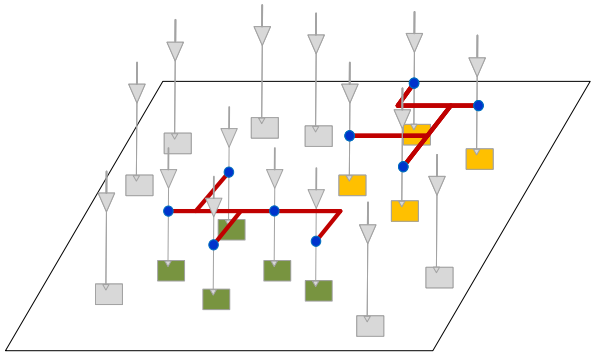

Fig. 7. Steiner-tree based multi-pin cross-link formation from two critical sink groups (green and orange, respectively).

In our approach, all critical sink pairs in a cluster are incorporated into a larger critical sink group. Then the candidate insertion nodes for these sinks are linked by forming Steiner trees applying the SALT algorithm [16] of the Steiner tree. Figure 7 shows an example of forming multi-pin cross-links for the critical sink groups based on the Steiner tree. This example includes two different scenarios, involving the construction of four-pin and five-pin cross-links respectively.

IV. Experimental Results

In this work, we implement our cross-link insertion algorithm using C++ and validated its effectiveness through SPICE simulation. The experiments are conducted on a Linux workstation equipped with an 8-core 2.67 GHz Intel Xeon CPU and 52 GB of memory. The algorithm is tested using the ISPD 2010 benchmark circuit [17]. The parameters are based on a $45nm$ predictive technology model [22], featuring a clock frequency

TABLE I

COMPARISON OF RESULTS FOR ISPD 2010 BENCHMARK SUITE [17], INCLUDING MEAN (μ), STANDARD VARIANCE (σ) OF THE CLOCK SKEW AND THE POWER CONSUMPTION (PWR). THE METRICS OF OUR RESULTS ARE EVALUATED THROUGH MONTE CARLO SIMULATION. THE RESULTS OF THE STATE-OF-THE-ART CLOCK SPINE AND CLOCK MESH REPORTED IN THE PAPER [18] ARE USED FOR COMPARISON. (EXCLUDING CIRCUITS *cns01* AND *cns02* DUE TO THEIR RESULTS NOT BEING LISTED IN [18].)

circuit	#sinks	Spine-based network [18]			Mesh-based network [19]			Ours		
		skew (*ps*)		PWR (*mW*)	skew (*ps*)		PWR (*mW*)	skew (*ps*)		PWR (*mW*)
		μ	σ		μ	σ		μ	σ	
cns03	1200	29.58	14.97	63.94	29.08	11.32	98.08	**27.96**	**9.61**	**61.63**
cns04	1845	42.96	18.92	92.04	35.04	18.72	142.70	**21.21**	**7.77**	**81.05**
cns05	1016	32.70	16.00	47.45	**31.91**	15.91	85.08	36.89	**10.21**	**38.27**
cns06	981	28.71	11.08	57.06	**19.46**	6.37	78.40	24.61	**4.43**	**50.13**
cns07	1915	34.00	15.76	99.92	37.10	8.59	136.13	**30.11**	**8.31**	**76.20**
cns08	1134	21.24	10.74	55.83	24.49	4.57	101.74	**15.84**	**7.54**	**53.87**
Avg	-	1.0	1.0	1.0	0.94	0.75	1.54	**0.83**	**0.55**	**0.87**

TABLE II

COMPARISON OF RESULTS ON BENCHMARK CIRCUITS [20] FOR CLOCK SCHEDULING AND SYNTHESIS, INCLUDING TOTAL CAPACITANCE (CAP), MAXIMUM CLOCK LATENCY (LATENCY), AND TIMING YIELD (YIELD). TIMING YIELD INDICATES THE RATIO OF CIRCUITS THAT MEET CONSTRAINTS TO THE TOTAL TESTED CIRCUITS. OUR APPROACH IS COMPARED TO THE METHOD PROPOSED IN [21].

circuit	#sinks	#skew constraints	RTS-UST [21]			Ours		
			Cap (*pf*)	Latency (*ps*)	Yield (%)	Cap (*pf*)	Latency (*ps*)	Yield (%)
msp	683	44990	1.5	89	100	2.8	94	**100**
fpu	715	16263	1.9	109	100	3.1	120	**100**
ecg	7674	63440	26.9	234	98.8	70.5	256	**99.7**
aes	13216	53382	200.7	1172	82.8	385.2	1245	**89.1**

of 2 GHz, a power supply voltage of $1.0V$, and a maximum slew limited to $50ps$ (10% of the clock cycle). The unit resistance of the wires is set to $0.1\Omega/\mu m$, while the unit capacitance is set to $0.2fF/\mu m$.

The experimental results comparing our method with the state-of-the-art clock spine topology [18] are presented in Table I. The μ (mean) and σ (standard deviation) of skew are obtained through 100 Monte Carlo simulations, adjusting parameters such as voltage and wire width to test the robustness of constructed topology. PWR represents the power consumption of clock network. The table demonstrates that while the mesh structure [19] reduces skew and skew variation compared to the clock spine, it also results in increased power consumption. Compared to the spine-based structure, our cross-link insertion algorithm has shown a 17% reduction in the mean of clock skew, a 45% reduction in the standard deviation of clock skew, and a simultaneous 13% decrease in power consumption.

For clock tree timing yield assessment on a benchmark circuit test set for clock scheduling and synthesis [20], 500 Monte Carlo simulations were executed. Each simulation subjected the clock tree to variations in wire width (±5.0%), supply voltage (±7.5%), and temperature (±15.0%) around their nominal values. Each simulation classified the circuit as good if all skew and slew constraints were met; otherwise, it was considered defective. Timing yield represents the ratio of good circuits to the total tested circuits. Table II presents the corresponding experimental results, showing that in comparison to the method reported in [21], a tree-based method utilizing skew tolerance management technique. Cap denotes the capacitance of clock network, $Latency$ represents maximum clock latency, and $Yield$ indicates timing yield based on 500 Monte Carlo simulations. The experimental results show that our timing yield outperforms it. While our capacitance overhead has increased compared to it, we believe this is acceptable. Because yield is the most critical metric for circuits, and it is necessary to trade off some capacitance overhead for a better yield.

V. CONCLUSION

This paper proposes a novel cross-link insertion method to mitigate clock skew and its variations in clock networks. Our approach employs geospatial- and timing-driven clustering, a delay sensitivity vector to select critical sink pairs, and a Steiner tree-based multi-pin structure to enhance network stability. Results show improvements of 17% in mean skew, 45% in skew standard deviation, and 13% in power consumption compared to leading non-tree designs.

ACKNOWLEDGMENT

This research is supported by the project of the Ministry of Industry and Information Technology High-Quality Development Program (No. CEIEC-2024-ZM02-0056), the Sichuan Provincial Party Committee's Military-Civilian Integration Committee, and the IEDA Laboratory of Southwest University of Science and Technology.

REFERENCES

[1] A. Abdelhadi, R. Ginosar, A. Kolodny, and E. G. Friedman, "Timing–driven variation–aware synthesis of hybrid mesh/tree clock distribution networks," *Integration*, vol. 46, no. 4, pp. 382–391, 2013.

[2] A. Rajaram and D. Z. Pan, "Fast incremental link insertion in clock networks for skew variability reduction," in *7th International Symposium on Quality Electronic Design (ISQED'06)*. IEEE, 2006, pp. 6–pp.

[3] K. Han, J. Li, A. B. Kahng, S. Nath, and J. Lee, "A global-local optimization framework for simultaneous multi-mode multi-corner clock skew variation reduction," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2744769.2744776

[4] M. R. Guthaus, X. Hu, G. Wilke, G. Flach, and R. Reis, "High-performance clock mesh optimization," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 17, no. 3, pp. 1–17, 2012.

[5] T. Mittal and C.-K. Koh, "Cross link insertion for improving tolerance to variations in clock network synthesis," in *Proceedings of the 2011 international symposium on Physical design*, 2011, pp. 29–36.

[6] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 18–23.

[7] A. Rajaram, D. Z. Pan, and J. Hu, "Improved algorithms for link-based non-tree clock networks for skew variability reduction," in *Proceedings of the 2005 international symposium on Physical design*, 2005, pp. 55–62.

[8] A. Rajaram and D. Z. Pan, "Variation tolerant buffered clock network synthesis with cross links," in *Proceedings of the 2006 international symposium on Physical design*, 2006, pp. 157–164.

[9] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Alpert, "Practical techniques to reduce skew and its variations in buffered clock networks," in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* IEEE, 2005, pp. 592–596.

[10] D.-J. Lee and I. L. Markov, "Multilevel tree fusion for robust clock networks," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2011, pp. 632–639.

[11] 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.

[12] A. 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*, 1997, pp. 217–223.

[13] J.-S. Yang, A. Rajaram, N. Shi, J. Chen, and D. Z. Pan, "Sensitivity based link insertion for variation tolerant clock network synthesis," in *8th International Symposium on Quality Electronic Design (ISQED'07)*. IEEE, 2007, pp. 398–403.

[14] R. Ewetz and C.-K. Koh, "Cost-effective robustness in clock networks using near-tree structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 4, pp. 515–528, 2015.

[15] J.-M. Ho, G. Vijayan, and C.-K. Wong, "New algorithms for the rectilinear steiner tree problem," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 9, no. 2, pp. 185–193, 1990.

[16] G. Chen and E. F. Young, "Salt: provably good routing topology by a novel steiner shallow-light tree algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 6, pp. 1217–1230, 2019.

[17] ISPD, "Ispd 2010 high performance clock network synthesis contest," 2010. [Online]. Available: http://archive.sigda.org/ispd/contests/10/ispd10cns.html

[18] Y. Kim and T. Kim, "Algorithm for synthesis and exploration of clock spines," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 263–268.

[19] G. Venkataraman, Z. Feng, J. Hu, and P. Li, "Combinatorial algorithms for fast clock mesh optimization," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, pp. 563–567.

[20] R. Ewetz, S. Janarthanan, and C.-K. Koh, "Benchmark circuits for clock scheduling and synthesis," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:64367944

[21] R. Ewetz and C.-K. Koh, "Clock tree construction based on arrival time constraints," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 67–74.

[22] J. Knudsen, "Nangate 45nm open cell library," *CDNLive, EMEA*, 2008.