SASIMI 2016 Proceedings

A Comparative Study on Multisource Clock Network Synthesis

Wen-Hsin Chen*, Chun-Kai Wang*, Hung-Ming Chen*, Yih-Chih Chou†, Cheng-Hong Tsai†
*Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan
†Physical Design Methodology Dept., Global Unichip Corp., Hsinchu, Taiwan
Email: cwsjudy@gmail.com; hmchen@mail.nctu.edu.tw

***Abstract*—Hybrid clock architecture offers a compromise between tree and mesh. While most of the relative works focus on tree-driven-mesh configuration, we are interested in the performance and optimization of multisource CTS flow provided by a state-of-the-art commercial tool, which applies a coarse mesh with local sub-trees. In this study, we analyze the QoR of conventional clock tree and multisource CTS on a real industrial design. We also propose several heuristic approaches to improving the performance of multisource CTS, especially for skew optimization. According to the experimental results, we reveal the benefits and drawbacks of each method, give some guidelines for determining the proper configuration for a design, and then summarize some future research directions.**

I. Introduction

Clock network design is one of the most important research topics in VLSI design. There are two common clock distribution architectures implemented to meet the timing requirements of the system. One is conventional clock tree, which is commonly used due to low power consumption, less routing resource usage as well as simplicity of implementation and simulation [1]. However, tree-based architecture can be highly sensitive to process, voltage, and temperature (PVT) variations, especially in high-performance chip designs. Clock mesh, the other architecture, provides better tolerance to variations [2]. Nevertheless, with lots of mesh nodes and unbalanced loads, clock mesh is difficult to analyze and automate [3]. Besides, additional metal wires and drivers lead to a lot more routing resource and higher power consumption than conventional clock tree.

Several works for non-tree networks optimization have been presented. [4] used cross links instead of mesh to improve robustness with a small increase in power consumption. Mesh edge reduction algorithms reduced power while a little skew overhead were presented in [5] and [6]. Moreover, [7] proposed a technique to customize mesh for non-uniform sink distributions and [8] developed a buffer reduction method for mesh-based clock distribution.

Hybrid architecture that combines tree and mesh is another method for power and skew trade-off. Research in [1] built a blockage-aware tree driving a mesh considering the loadings to minimize local skew. Others produced a combination of non-uniform meshes and un-buffered trees to reduce skew variations while minimizing power and metal area overhead [3]. Besides, the authors of [9] proposed an algorithm to choose the position of tapping points on a tree-driven-grid clock network that can handle non-uniform loads. Synopsys IC Compiler (ICC) provides multisource clock tree synthesis (CTS) methodology, which applies a coarse mesh with local sub-trees to fill the gap between conventional clock tree and clock mesh. [10], [11], [12] and [13] presented the introduction and implementation about this methodology.

While most of the relative works focus on tree-driven-mesh architecture, we are interested in the performance and optimization of multisource CTS flow provided by a commercial EDA tool. In this paper, we analyze the quality of results (QoR) of conventional clock tree and multisource clock network on a real industrial design. We also propose several heuristic approaches to improving the performance of multisource CTS, especially for skew optimization. From the results, we give some guidelines for determining the proper configuration for a design.

The multisource CTS implementation flow and construction of different mesh configuration are shown in Section II. Section III presents the details of proposed approaches for tap-point and sink assignment. Section IV reports the analysis of our experimental results on a real industrial case. Finally, we give the conclusions in Section V.

II. Multisource Mesh Configuration

A. Overall Flow

Multisource clock network is a hybrid method of conventional CTS and clock mesh. The structure is shown in Fig. 1, which consists of a mesh driven by a pre-mesh tree. Multisource drivers connect to the mesh at a limited number of locations referred to as taps. A multisource clock tree structure driven by the mesh consists of subtrees, each driven by a tap.

With the mixed structure, multisource CTS has lower skew, better QoR and on-chip variation (OCV) tolerance than conventional clock tree because of the increasing common path. Furthermore, by using coarse mesh rather than dense mesh, multisource CTS consumes less power and routing resources and become easier to implement than a traditional clock mesh.

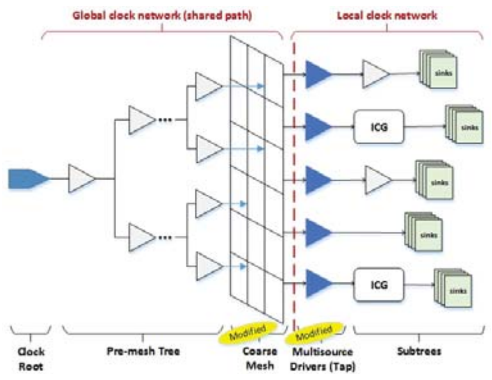


Fig. 1. Structure of multisource clock distribution network. The red dashed line divides the structure into global and local clock network. Our modified parts are marked.

In our work, first, we generate appropriate size of mesh for the design, but there are no guides about how dense the mesh should be. We do some works on mesh configuration to find a good one, and details are presented in Section II.B and II.C. Next, we insert the multisource drivers to the intersections of the horizontal and vertical mesh spines and assign each sink to the corresponding driver. The influence of this stage on final results are significant, so we propose some approaches to minimizing clock skew in Section III. Then we implement the rest of the flow following the steps recommended by [13], including building pre-mesh tree and the subtrees driven by the multisource drivers, routing the clock nets and analyzing the results.

B. Mesh Density Adjustment

The mesh configuration in multisource CTS dominates the latency shared by each sink. The lower the mesh density, it behaves more like conventional clock tree. Conversely, as the mesh density increasing, it becomes like a pure mesh and benefits from better OCV tolerance but pays expense on higher power consumption and routing resources [10]. Thus, determination of the mesh density, which means number of horizontal and vertical straps, is very important.

We first use a design service company in-house utility [14] to get a recommended mesh size. This utility estimates the mesh density by the aspect ratio of bounding box of all sink pins and the experiential average pin number driven by a tap. Take the case we use as an example, the utility recommends the configuration 7*9 (7 horizontal with 9 vertical straps). Then we try other mesh sizes based on the recommended one, and the QoR results are shown in Section IV.

C. Non-uniform Mesh

Because the intersections of the horizontal and vertical mesh segments are potential multisource driver locations, mesh generation affects the following clock network significantly. The mesh configuration mentioned in previous section is uniform structure; however, non-uniform sink distribution and macros in the design might diminish the performance of mesh, so we use two approaches of non-uniform clock mesh to fix the problem.

First one is the macro-avoiding approach. Mesh segments that overlap macros are not useful and will waste the power consumption. We add a function provided by ICC to avoid creating straps and vias intersecting those macros so that the power can be reduced. The other method is creating mesh considering sink distribution. Our flow is as follows: we collect all sink locations first, and then calculate each column and row density under the given uniform mesh configuration. Next, it is necessary to compare a row with its adjacent rows and a column with its adjacent columns before moving straps. The final position of a strap depends on the difference in density between it and its adjacencies. In addition, the straps can only be moved within a limit range in order to maintain the balanced characteristic of mesh structure.

III. Tap-point Determination and Sink Assignment

In multisource CTS implementation, determination of the number and positions of tap points which lower-level sinks are connected to, as well as the arrangement of sink grouping can affect the QoR of the final clock network. Variances between the trees that sit beyond a collection of multisource drivers lead to disparities in local clock skew and insertion delay, which might impact on buffer area minimization and inter-clock delay-balancing efforts adversely [12].

In our flow, we choose locations of multisource drivers on the vias where the horizontal mesh straps intersect the vertical ones for convenience and uniform distribution. However, loading and topology of tree are also factors to be considered when making decision on tap points. Furthermore, ICC assigns sinks to taps based on location and blockages rather than timing relation between them, which is another important element for skew minimization.

We propose several heuristic approaches and analytic methods on tap determination and sink assignment in this section. Because we can control only a few parts in the flow, our works just modify some user-defined parameter according to the limited information we get from ICC and observe if our research is meaningful.

A. Arrival-time-based Approach

Clock skew is defined as the difference of arrival time between any two registers. In order to minimize the skew, we observe the arrival time distribution of all sinks. Take Fig. 2 as an example, the more the color is close to yellow, the bigger the sink arrival time is; the more the color is similar to blue, the smaller the sink arrival time is. It is obvious that arrival time of sinks distribute based on the tap locations since we can easily divide the sinks with different colors by red dashed lines. Thus, we decide to decrease the delay in the region where tap driving sinks with higher arrival time, so that clock skew might be minimized.

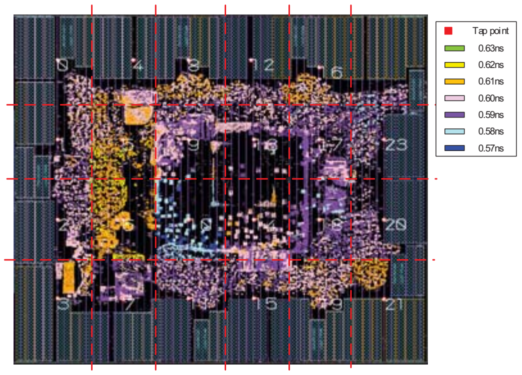


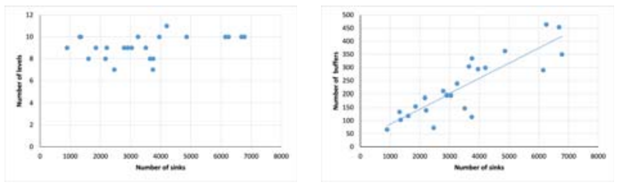
Fig. 2. Arrival time distribution graph of mesh configuration 4*6. We divide the sinks with different colors by red dashed lines.

As shown in Fig. 2, red dots with white text on them denote the tap drivers. We can see that the sinks around Tap 5 and Tap 6 are most orange and yellow, while others are pink, purple and even blue ones. According to this distribution, we insert another tap between Tap 5 and Tap 6 to share the sinks driven by them and hope this can lower the latency of those sinks. The result after tap insertion shows that colors change not only around the inserted tap but also in other regions of the design. It is true that the sinks in the targeted range have lower arrival time than in the original case, but we cannot make sure whether the overall skew would be better after tap insertion.

B. Loading-based Approach

To work on tap location determination, we collect the information of each tap after multisource CTS. From the results, we know how many sinks and levels of sub-tree under each driver, the number of buffers used by each tap and where the tap located in the design.

We find that the differences in levels, the depth of taps, do not change as number of sinks increases (as shown in Fig. 3(a)). Thus, we infer that subtrees with huge different number of sinks have similar levels because of the objective in balancing the latency. In contrast of level, number of buffers is more related to the driven sinks (as shown in Fig. 3(b)). We decide to have the same number of sinks driven by each tap so that the number of used buffers could be similar as well. That is, this approach is based on balancing the number of sinks and buffers, which means the loading of each tap keeps the same.



(a) Sink vs. level (b) Sink vs. buffer

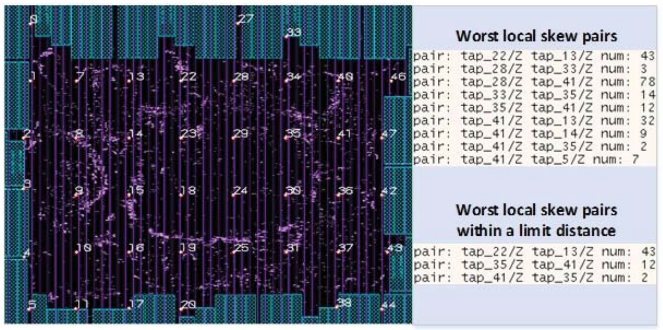
Fig. 3. Relation between number of sinks, buffers and levels. Number of level does not change a lot as number of sinks increases. In contrast, number of buffers is more related to the driven sinks.

We focus on the taps driving bigger number of sinks and find that there are two pairs of these taps next to each other. To share the sinks with them, a new tap is inserted to the middle of each pair. From the results, we confirm that loadings of taps become more balanced than the original since the biggest difference in number of driven sinks and in numbers of buffers are both reduced near 25%.

C. Local-skew-based Approach

The skew between a sequentially related sink pair is called local clock skew. To minimize clock skew, we wonder where the worse local skews appear in the design. We use ICC to report the worst 200 local clock skew pairs of sinks and check which taps the sinks are driven by. Then we make a list of those worse skew tap pairs in Fig. 4.

In the upper right of Fig. 4 shows the worst local skew pairs statistic. We can see that these pairs contain the close tap pairs and the far ones. That is, the sinks with sequential relation may be placed in the locations far from each other. However, in this stage we cannot modify the placement, so if there is a worst local skew pair with long distance between each sink, it is hard to fix during this step. Thus, we choose the skew pairs with distance less than or equal to a given threshold as listed in the lower right of Fig. 4.



```
pair: tap_22/Z tap_13/Z num: 43
pair: tap_28/Z tap_33/Z num: 3
pair: tap_28/Z tap_41/Z num: 78
pair: tap_33/Z tap_35/Z num: 14
pair: tap_35/Z tap_41/Z num: 12
pair: tap_41/Z tap_13/Z num: 32
pair: tap_41/Z tap_14/Z num: 9
pair: tap_41/Z tap_35/Z num: 2
pair: tap_41/Z tap_5/Z num: 7
```

```
pair: tap_22/Z tap_13/Z num: 43
pair: tap_35/Z tap_41/Z num: 12
pair: tap_41/Z tap_35/Z num: 2
```

Fig. 4. Statistics of worst local skew tap pairs.

1) Tap-point Determination: After filtering the worst local skew pairs, it is clear that which adjacent pairs of taps have more skew violations. Since the skew violation occurs due to different taps the sinks assigned to, we hope to fix this by making these sinks driven by the same tap. Insert a tap in the middle or merge two taps are candidates for solution. However, both of them have potential drawback: this does not improve the result but make it worse.

Fig. 5 is an example of tap insertion. Rectangles with same color represent the sinks with timing relation between them. Local skew violations of blue sinks and black sinks happen in the original tap arrangement in the left of Fig. 5. Therefore, we insert Tap 3 between Tap 1 and Tap 2 as shown in the right of Fig. 5. The result is that we successfully assign all blue sinks to the same tap, but the green ones which are grouped to the same tap at first become separated after tap insertion.

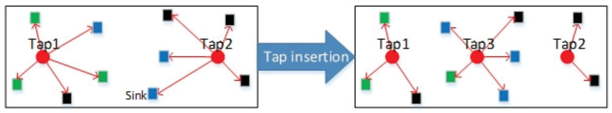


Fig. 5. An example of tap insertion based on local skew violation. After adding Tap 3, some timing-related sinks are assigned to the same tap, while others are not.

To sum up, it is hard to expect the result of tap rearrangement based on local skew violations because other sinks driven by the targeted taps can be affected due to automated sink assignment.

2) Sink Reassignment: In multisource CTS flow, ICC automatically splits sinks to multisource drivers based on their placement and the blockages after tap point determination. As a result, sinks with timing relation might be assigned to different taps. Furthermore, ICC only provides a command to force a sink to assign to a specific tap driver. Although we find a way to check whether a pair of sinks have timing relation between them by checking their local skew report, there is still no way to control sink grouping and assignment.

We decide to fix these violations by reassigning the sinks to the same tap without modifying location and number of taps. We still focus on the sink pairs with distance under a given threshold so that the reassignment would cause little drawback in latency. As illustrated in Fig. 6, sinks can have timing relation with more than one sink. We change the assignment of sinks with most of their relative sinks assigned to another tap. For example, B3 is reassigned rather than B1 or B2, and in the same way A1 and A2 are chosen to be reassigned. Furthermore, if a sink has only one relative sink driven by another tap, we ignore it because this kind of sinks are too many and the reassignment might be insufficient. For instance, we do not change the assignment of both black sinks in Fig. 6.

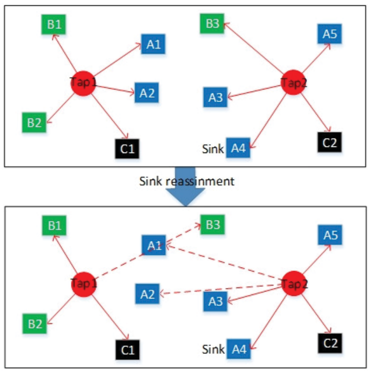


Fig. 6. An example of sink reassignment. We reassign the A1, A2 and B3 because most of their relative sinks are assigned to another tap.

IV. Experiment Results

A. Experiment Setup

Our experiments are performed on a real industrial case applying 28nm process and other characteristics are listed in Table I. All of our flows use the same placement results made from ICC as the input. We implement our works on two clock domains with different number of sinks in this design, and the information of the clocks are displayed in Table II. We analyze the results after the CTS stage and before routing, and we collect the data under the scenario operating on typical conditions and considering OCV effects.

TABLE I
Test Design Data

Case Name	ASIC1
Process	28nm
Number of Macro Cell	101
Number of Std Cell	782460
Total Macro Cell Area	907977.76 μm^2
Total Std Cell Area	756555.47 μm^2
Clock Routing Layers	M3-M6
Clock Mesh Layers	M7,M8

TABLE II
Data of Two Clocks in ASIC1

Clock Name	CLK1	CLK2
Number of Sinks	79380	3007
Period	1.49 ns	2.50 ns
Maximum Transition Time	100 ps	100 ps

B. Different Mesh Sizes

In Table III and Table IV, we present the QoR results of multisource CTS flow under different mesh configurations. We apply different mesh settings on each case based on the rule we mentioned in Section II.B. The CTS row means the result of original CTS flow while others are results of multisource CTS flow with different mesh densities. The ClkCells item is the total number of cells in the clock network. The blue and the red data denote the best and the worst one in each column.

TABLE III
QoR OF DIFFERENT MESH CONFIGURATIONS UNDER CLK1

Clock name **CLK 1**

	Mesh Size	ClkCells	Global Skew(ns)	Longest Path	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
CTS		8368	0.0814	0.5673	0.0784	23	0	38.17	-0.162	-183.06	5613
MSCTS	2*3	9286	0.0654	0.6718	0.0549	21	6	42.252	-0.059	-35.475	2867
MSCTS	3*4	10833	0.0727	0.6456	0.0614	25	12	46.3504	-0.051	-42.562	3034
MSCTS	4*6	10912	0.0522	0.6143	0.0448	25	23	46.4961	-0.076	-29.198	2459
MSCTS	6*8	11282	0.0519	0.5918	0.043	22	41	49.1426	-0.08	-26.244	2480
MSCTS	7*9	11320	0.0467	0.6049	0.0383	25	54	50.543	-0.114	-27.237	2143
MSCTS	8*10	11472	0.0441	0.5976	0.0373	24	68	51.7981	-0.12	-28.868	2381
MSCTS	8*12	11273	0.0535	0.5951	0.0365	24	77	51.7635	-0.15	-45.977	3423

TABLE IV
QoR OF DIFFERENT MESH CONFIGURATIONS UNDER CLK2

Clock name **CLK 2**

	Mesh Size	ClkCells	Global Skew(ns)	Longest Path	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
CTS		265	0.0459	0.4691	0.0361	16	0	38.17	-0.162	-18306	5613
MSCTS	2*3	278	0.0742	0.6176	0.0665	16	6	40.1108	-0.083	-17547	8249
MSCTS	3*4	286	0.0414	0.5542	0.0337	18	12	37.4285	-0.071	-17315	7820
MSCTS	4*6	337	0.1352	0.581	0.115	22	24	38.961	-0.084	-11642	6623
MSCTS	6*8	371	0.0423	0.5492	0.0349	20	42	40.7761	-0.102	-23992	9935

From the results, the difference between multisource CTS and original CTS is obvious. Multisource CTS flow consumes more clock network power and has longer latency than original CTS flow. But on the other hand, it has 20-50% improvement in clock skew and over 70% improvement in total negative slack (TNS).

The trends of variation among multisource CTS with different mesh sizes are not significant, but we can still see that as mesh size growing up, the skew is getting better but the power consumption increases. The timing performance is unstable, but it is clear that when mesh size exceeds a threshold, timing performance starts to decrease. This phenomenon may be caused by congestion. Thus, it seems that the most portable configuration can be 6*8 or 7*9, both of whom are near the default configuration recommended by in-house utility mentioned in Section II.B.

The above analysis is based on the results of CLK1 because multisource CTS flow seems not so efficient on CLK2. We believe that this situation is due to the distribution and number of sinks. As shown in Fig. 7, most of the sinks of CLK2 locate in the left of the chip and are sparse while sinks of CLK1 are much denser and spread all over the chip. We can conclude that multisource CTS performs better when the target clock has to cover larger area.

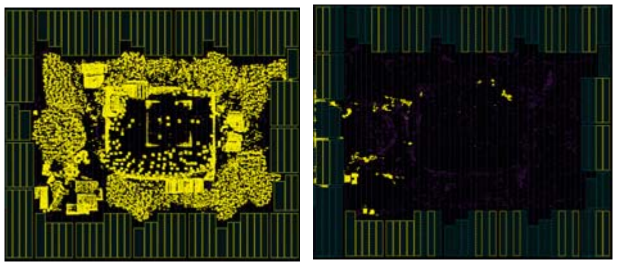


Fig. 7. Sink distribution of 2 target clock domains. The left is CLK1, and the right is CLK2

C. Non-uniform Mesh

Table V represents the results of different non-uniform mesh configurations, including macro-avoiding approach and mesh considering sink distribution. The red data indicate it is worse than the original one while the blue data represents that it is better, and the black ones are similar to the original. The results show that macro-avoiding approach averagely consumes 3-4% less clock power than uniform mesh configuration with a little tradeoff in skew or timing performance.

We use C++ programing language with sink locations and number of straps as input to get non-uniform mesh configuration. In Table V, we can see that clock skew is reduced obviously when creating mesh considering sink locations. However, more clock power is consumed and the reason may be the increase in number of buffers used to balance the non-uniform mesh.

TABLE V
QoR OF DIFFERENT MESH AND NON-UNIFORM MESH CONFIGURATIONS
CLK1 AND CLK2

Clock name **CLK1**

Mesh Size	ClkCells	Global Skew(ns)	Improve (%)	Local Skew(ns)	Improve (%)	Max Level	Taps	Clk Power (mw)	Improve (%)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
3*4	10833	0.0727		0.0614		25	12	46.3504		-0.051	-42.562	3034
3*4+avoid	10389	0.0728		0.0675		24	12	44.9732	2.97%	-0.052	-22.994	2123
3*4_nonuni	10681	0.0557	23.38%	0.0468	23.78%	26	12	47.0877		-0.051	-19.685	1877
4*6	10912	0.0522		0.0448		25	23	46.4961		-0.076	-29.198	2459
4*6_nonuni	11067	0.0466	10.73%	0.0413	7.81%	21	23	47.03		-0.05	-29.056	2638
6*8	11282	0.0519		0.043		22	41	49.1426		-0.08	-26.244	2480
6*8+avoid	11024	0.0452	12.91%	0.0433		22	34	47.7243	2.89%	-0.083	-28.077	2500
6*8_nonuni	11256	0.0471	9.25%	0.0392	8.84%	24	43	49.0268		-0.066	-27.705	2421
7*9	11320	0.0467		0.0383		25	54	50.543		-0.114	-27.237	2143
7*9+avoid	11085	0.045	3.64%	0.0432		24	48	48.7706	3.51%	-0.07	-31.793	2672
7*9_nonuni	11137	0.0435	6.85%	0.0367	4.18%	23	53	49.2179	2.62%	-0.14	-51.507	3246
8*10	11472	0.0441		0.0373		24	68	51.7981		-0.12	-28.868	2381
8*10+avoid	11443	0.0528		0.0479		22	49	50.5666	2.38%	-0.125	-40.047	2642

Clock name **CLK2**

Mesh Size	ClkCells	Global Skew(ns)	Improve (%)	Local Skew(ns)	Improve (%)	Max Level	Taps	Clk Power (mw)	Improve (%)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
2*3	278	0.0742		0.0665		16	6	40.1108		-0.083	-175.47	8249
2*3+avoid	257	0.0732	1.35%	0.065	2.26%	16	6	36.5515	8.87%	-0.073	-105.31	6114

D. Tap Determination

In each approach of tap point determination, we first get the information we need by ICC commands and use TCL or C++ programing language to collect these data in the way we want. Results of tap determination based on arrival time distribution is listed in Table VI. We use the case mentioned in Section III.A, but it seems that balance of arrival time just appear in the limit range, so the overall skew and timing do not have significant improvement compared with the original structure.

TABLE VI
QoR OF ARRIVAL-TIME-BASED TAP ASSIGNMENT FOR CLK1

Clock name **CLK1**

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
4*6	10912	0.0522	0.0448	25	23	46.4961	-0.076	-29.198	2459
4*6 tap+1	11208	0.0587	0.0518	21	24	48.0244	-0.091	-33.113	2632
4*6 tap+2	11082	0.0521	0.0445	23	25	47.7086	-0.113	-32.659	2609

In the experiment of loading-based approach, we combine with non-uniform mesh and the results are shown in Table VII. We insert two taps in 3*4_nonuni configuration, and one of them is located in the middle of four high-loading-taps so this tap is not near any mesh strap. This may be the reason for worse performance of this case. In case 6*8n tap-4, we remove four taps with much less loading than others from 6*8_nonuni configuration, and the result shows that it saves some power with a little increase in skew timing violations. Taps and sink arrival time distribution of case 3*4 tap+1 are shown in Fig. 8. We can see that after adding Tap 11 between Tap 10 and

Tap 12, most sinks in that region are assigned to Tap 11 rather than split to different taps. Thus, the more unbalanced loading of each tap leads to the worse skew, but timing violations reduce because lots of nearby sinks are grouped to the same tap. We can also explain the results of other cases by this reason.

TABLE VII
QoR OF LOADING-BASED TAP ASSIGNMENT FOR CLK1 AND CLK2

Clock name CLK1

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
3*4_nonuni	10681	0.0557	0.0468	26	12	47.0877	-0.051	-19.685	1877
3*4n tap+2	10792	0.0685	0.0614	24	14	47.2992	-0.059	-38.584	3023
4*6	10912	0.0522	0.0448	25	23	46.4961	-0.076	-29.198	2459
4*6 tap+2	10994	0.0485	0.0408	25	25	47.5497	-0.065	-31.845	2809
6*8	11282	0.0519	0.043	22	41	49.1426	-0.08	-26.244	2480
6*8 tap+2	11238	0.0525	0.0443	25	43	49.7433	-0.086	-27.271	2475
6*8_nonuni	11187	0.0495	0.0436	22	43	48.3909	-0.085	-29.503	2476
6*8n tap-4	11161	0.0543	0.0455	23	39	47.6806	-0.072	-30.44	2729

Clock name CLK2

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
3*4	286	0.0414	0.0337	18	12	37.4285	-0.071	-173.15	7820
3*4 tap+1	308	0.0713	0.0534	19	13	36.705	-0.083	-85.044	5469

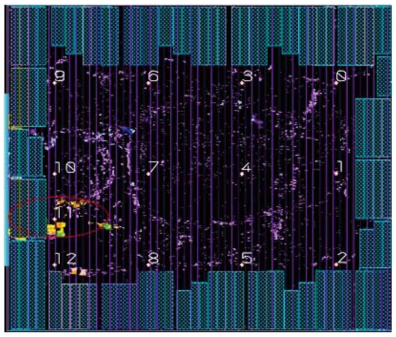


Fig. 8. Taps and sink arrival time distribution of case 3*4 tap+1

Table VIII shows the results of tap assignment based on worse local skew pairs. We apply tap insertion and tap merging in the case 6*8. According to the QoR results, it seems that only clock skew in case 4*6 tap+1 is improved, but nothing gets better in other cases. The results are expected due to the potential risk mentioned in Section 3.1.1. From the reports of worse skew pairs in original and modified configurations, we can see that violations in original case almost disappear in the modified ones with a few new violations. To sum up, though overall skew and QoR performance may not be improved by this approach, it can help reduce the skew in local area.

TABLE VIII
QoR OF LOCAL-SKEW-BASED TAP ASSIGNMENT FOR CLK1 AND CLK2

Clock name CLK1

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
4*6	10912	0.0522	0.0448	25	23	46.4961	-0.076	-29.198	2459
4*6 tap+1	11111	0.0446	0.042	23	24	48.7478	-0.064	-28.178	2700
6*8	11282	0.0519	0.043	22	41	49.1426	-0.08	-26.244	2480
6*8 tap+2	11209	0.0567	0.0467	26	43	49.806	-0.091	-32.651	2797
6*8 merge4->2	11403	0.0627	0.0533	22	39	49.5356	-0.085	-37.821	2944

Clock name CLK2

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
3*4	286	0.0414	0.0337	18	12	37.4285	-0.071	-173.15	7820
3*4 tap+1	311	0.0542	0.0381	19	13	39.4617	-0.089	-195.88	8521

E. Sink Reassignment

The data shown in Table IX is the result of the approach mentioned in Section III.C, which is also based on worse local skew pairs. We only reassign a few sinks to the proper taps where lots of their timing-related sinks assigned to. After the reassignment, the tap pairs shown in the original worse skew report are gone, and it seems that the overall skew is reduced without significant trade-off in clock power consumption. These results show that clock skew and QoR may become better when grouping sinks to taps considering both location and timing relation.

TABLE IX
QoR OF LOCAL-SKEW-BASED SINK REASSIGNMENT FOR CLK1

Clock name CLK1

Mesh Size	ClkCells	Global Skew(ns)	Local Skew(ns)	Max Level	Taps	Clk Power (mw)	SetupWNS (ns)	SetupTNS (ns)	Setup Violations
4*6	10912	0.0522	0.0448	25	23	46.4961	-0.076	-29.198	2459
4*6 reassign7	10887	0.05	0.044	25	23	46.4791	-0.088	-29.252	2447
6*8	11282	0.0519	0.043	22	41	49.1426	-0.08	-26.244	2480
6*8 reassign6	10781	0.0504	0.0464	21	41	48.8009	-0.066	-19.013	2049

V. CONCLUSIONS

In this work, we present a study by analyzing the QoR of conventional clock tree and multisource clock network implemented with state-of-the-art tool on a real industrial design. We focus on the steps which can be controlled in the flow and propose some heuristic approaches to improving the performance of multisource CTS, especially for skew optimization. From the results, we find the proper mesh configuration for a design and that multisource CTS performs better when the target clock covers larger area. Though the results seem to be case-dependent, we can still conclude that skew and QoR may get better by performing tap and sink assignment, considering both location and timing relation of sinks. How to automate this flow is an important future research direction.

REFERENCES

[1] L. Xiao, *et al.*, "Local clock skew minimization using blockage-aware mixed tree-mesh clock network," in *International Conference on Computer-Aided Design*, 2010, pp. 458–462.
[2] C. Yeh, *et al.*, "Clock distribution architectures: a comparative study," in *International Symposium on Quality Electronic Design*, 2006, pp. 85–91.
[3] A. Abdelhadi, *et al.*, "Timingdriven variationaware synthesis of hybrid mesh/tree clock distribution networks," *Integration, the VLSI Journal*, vol. 46, pp. 382–391, sept. 2013.
[4] 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, pp. 515–528, 2015.
[5] A. Rajaram and D. Pan, "Meshworks: An efficient framework for planning, synthesis and optimization of clock mesh networks," in *Design Automation Conference,*, 2008, pp. 250–257.
[6] G. Venkataraman, *et al.*, "Combinatorial algorithms for fast clock mesh optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 131–141, Jan. 2010.
[7] M. R. Guthaus, *et al.*, "Non-uniform clock mesh optimization with linear programming buffer insertion," in *Design Automation Conference*, 2010, pp. 13–18.
[8] J. Reuben, *et al.*, "Buffer reduction algorithm for mesh-based clock distribution," in *Advances in Electronics, Computers and Communications*, 2014, pp. 1–4.
[9] N. Y. Zhou, *et al.*, "Pacman: driving nonuniform clock grid loads for low-skew robust clock network," in *System Level Interconnect Prediction*, 2014, pp. 1–5.
[10] N. Patel, "A novel clock distribution technology multisource clock tree system (mcts)," *Int. Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, pp. 2234–2239, June 2013.
[11] T. Yang and Y. Tang, "Implementation of multi-source clock tree synthesis," in *Synopsys Users Group*, 2014.
[12] R. Helfand, "Application of the multisource cts operative in ic compiler," in *Synopsys Users Group*, 2013.
[13] "Implementing multisource clock trees," in *IC Compiler Implementation User Guide, version I-2013.12-SP4*, 2013, pp. 5.126–5.133.
[14] K. Chou, "Ic compiler multisource cts - practical experience sharing," in *Synopsys Users Group*, 2015.