Using Spatial-Correlation to Deal With PVT Variations During Clock-Tree Synthesis

[Blind Review Version]

ABSTRACT

With the rapid rising process, voltage, and temperature (PVT) variations impact to the interconnect performance, it is crucial to consider process-variation during clock tree synthesis. Although there are several remedies to lessen the variations impacts to clock such as link insertion or mesh-tree structure, the basic structure of clock topology plays the most crucial role to determine the variation caused skew due to the spatial-correlation nature of PVT variation.

In this paper, we propose to consider spatial-correlation during clock tree construction to take advantage of locality. During the clock topology construction, the clock sinks with tight timing constraints will be clustered or partitioned together and therefore increases the spatial-correlation to reduce clock skew. With the reduction of clock skew, the algorithm can recover the margin from clock-skew uncertainties and reallocate the slacks to the path-delay and their uncertainties. Therefore, the total negative slacks are significantly improved. The experimental result shows that our algorithm can significantly reduce the total negative slack and speedup the timing convergence of a design. For ISCAS89 benchmark circuits, our algorithm achieves up to 32% total negative slack reduction compared to a traditional balanced bipartition algorithm.

1. INTRODUCTION

As technologies move into deep sub-wavelength era, process, voltage, and temperature (PVT) variation effects gradually become serious. As pointed out by Sani Nassif from IBM [1], the interconnect variation effects may dominate the gate delay in the future technology. Since clock net are interconnect-dominated circuit structure, the process-variation impacts to clock will soon be significant. As a result, it is very important to consider PVT variation effects during clock synthesis.

We now classify the variation impacts to clocks. The *process-variation induced clock-skew* is the clock-skew introduced by manufacturing processes and varies from chip to chip. The *environmental variation induced clock-skew* is caused by supply voltage and temperature variations and varies over time on the same chip. The design-inherited clock-skew can be eliminated using various zero-skew (zero design-inherited clock-skew) clock-tree optimization techniques [1, 2, 3, 4, 5]. The environmental variation induced clock-skew is usually tackled by design methodologies, such as

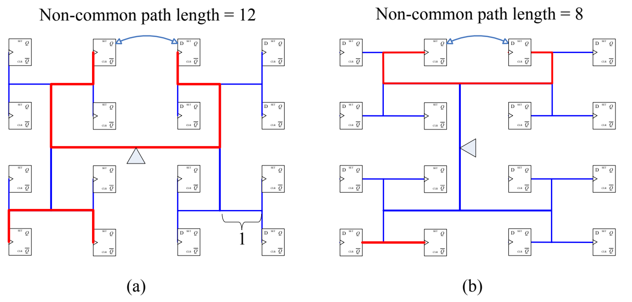

Figure 1: Two clock-tree topologies for the same set of clock sinks.

placing clock buffers near power/ground sources and avoid routing clock nets through hot spots.

In vision of this effect, there are several algorithms proposed to to balance or fine-tune the variation impacts to clock such as link insertion after clock tree construction [8]. However, since the basic structure of clock topology plays the most crucial role in determining the variation caused skew due to the spatial-correlation nature of PVT variations. There is a need to develop an clock tree topology generation algorithm to consider this effect. Velenis et al. [6, 7] proposed a greedy clustering-based algorithm to reduce process-variation induced clock-skews on timing critical paths. The algorithm creates the clock-tree topology by first clustering the source and target clock sinks of the most timing critical path together, then clustering the two clock sinks of the second most timing critical paths, and so on so forth. However, the clustering based algorithm lacks the global view and has practical limitations. For example, the algorithm will cluster the two clock sinks of the most timing critical path together even if the two clock sinks are located at the opposite corners of the chip.

It is observed that the maximum process-variation induced clock-skew of a clock-tree is positively correlated to its clock-delay. Previous works [3, 4, 5] reduce the maximum process-variation induced clock-skew by clock-delay minimization using combinations of routing, buffer-insertion, buffer-sizing, and wire-sizing techniques. Although these works reduce the maximum process-variation induced clock-skew, they do not necessarily reduce the total clock-skew uncertainty on timing critical paths and the total slack requirement of a design. This is because these works all start from a given clock-tree topology that is not aware of the locations of timing critical paths. For example, Figure 1 shows two clock-trees for the same set of clock sinks from two different starting clock-tree topologies. The arc in Figure 1 indicates the most timing critical path of the circuit. Although the two clock-trees have the same maximum process-variation induced clock-skew (assuming clock-trees are symmetric and process-variations are uniform), the clock-tree in Figure 1(a) requires more slack and can slow down the timing convergence, hence the clock-tree in Figure 1(b) is a preferred topology.

In current design flows, slacks are reserved for timing uncertainties to ensure correct functionality of manufactured chips. However, the increasing timing uncertainties and demand on slacks can slow down the timing convergence. The timing convergence status is usually monitored by two parameters: *Worst Negative Slack* (WNS) and *Total Negative Slack* (TNS). While WNS indicates the design effort that is needed to fix the worst timing violation, TNS indicates the aggregated design effort that needs to be done to achieve timing closure. In modern VLSI system designs, clock distribution networks and logic designs are usually done in separate steps. Therefore, separate slack requirements are imposed for path-delay uncertainty and clock-skew uncertainty.

In this paper, we propose to consider spatial-correlation during clock tree construction to take advantage of locality. During the clock topology construction, the clock sinks with tight timing constraints will be clustered or partitioned together and therefore increases the spatial-correlation to reduce clock skew. With the reduction of clock skew, the algorithm can recover the margin from clock-skew uncertainties and reallocate the slacks to the path-delay and their uncertainties. Therefore, the total negative slacks are significantly improved. The experimental result shows that our algorithm can significantly reduce the total negative slack and speedup the timing convergence of a design. For ISCAS89 benchmark circuits, our algorithm achieves up to 32% total negative slack reduction compared to a traditional balanced bipartition algorithm.

The rest of the paper is organized as follows: in Section 2, we review clock-tree design issues and study general properties of a buffered clock-tree. In Section 3, we present details of our clock-tree topology generation algorithm. Experimental results are presented in Section 4. Finally, Section 5 concludes this work.

2. PRELIMINARIES

The maximum and minimum path-delays between FF_i and FF_j are D_{ij} and d_{ij}, respectively. Let CP be the clock-period, and T_{setup} and T_{hold} be the setup-time and hold-time of a flip-flop. Define the clock arrival time to FF_i as T_i and the clock-skew between FF_i and FF_j as $s_{ij} = T_i - T_j$. By absorbing the flip-flop delays as part of the path-delays, the hold-time and setup-time constraints can be written as:

$$s_{ij} + d_{ij} \geq T_{hold} + \kappa[\sigma(s_{ij}) + \sigma(d_{ij})]. \quad (1)$$

$$CP - D_{ij} - s_{ij} \geq T_{setup} + \kappa[\sigma(s_{ij}) + \sigma(D_{ij})]. \quad (2)$$

The terms $\kappa[\sigma(s_{ij}) + \sigma(d_{ij})]$ and $\kappa[\sigma(s_{ij}) + \sigma(D_{ij})]$ in (1) and (2) are the hold-time and setup-time slack requirements for s_{ij}, which depend on the choice of κ. Due to the increasing design size, a 6σ slack, or $\kappa = 6$, is required for each timing constraint to guarantee a reasonable yield. However, the soaring clock frequency (decreasing CP) and increasing process-variations(increasing $\sigma(\cdot)$) have made it more difficult to achieve timing closure. It is necessary to reduce the slack requirement of a design to speedup timing convergence.

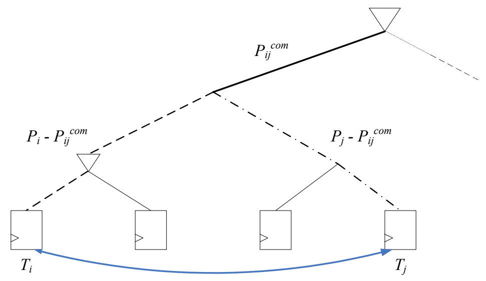

Figure 2: Clock-skew uncertainty and clock distribution paths.

If the hold-time slack $\delta_{ij}^h = s_{ij} + d_{ij} - T_{hold}$ or the setup-time slack $\delta_{ij}^s = CP - D_{ij} - s_{ij} - T_{setup}$ of s_{ij} are smaller than its slack requirements, the differences need to be counted toward the total negative slack. The total negative slack of a design can be reduced by minimizing the total clock-skew uncertainty on the paths with negative slack. This can be done by improving the clock-tree topology generation process. Let P_i and P_j be the clock distribution paths of FF_i and FF_j. Velenis et al. [6, 7] observe that the *common part* of P_i and P_j, or P_{ij}^{com}, do not contribute to the clock-skew uncertainty of s_{ij}. As shown in Figure 2, $\sigma(s_{ij})$ is likely to reduce if we reduce the lengths of the non-common part of P_i and P_j, $P_i - P_{ij}^{com}$ and $P_j - P_{ij}^{com}$. A greedy algorithm that clusters source and target clock sinks of timing critical paths is likely to reduce the total clock-skew uncertainty and total slack requirement. However, this greedy approach can also result in an increase on total wire length of the clock-tree, which in turn increases the total clock-skew uncertainty and cancels the benefit.

Recently, a new approach for reducing clock-skew uncertainty is proposed [8]. The principle is that if we add a link, an extra wire segment, to connect a pair of nodes in the clock-tree with equal clock-delay, the clock-skew uncertainty between these two nodes may be reduced. This is a post processing approach and still relies on a good clock-tree topology. For example, a better clock-tree topology will require fewer link insertions. Therefore, there is a need to investigate clock-tree topology generation algorithms.

3. SPATIAL CORRELATION DRIVEN CLOCK PIN PARTITIONING AND CLUSTERING FOR CLOCK SYNTHESIS

We use recursive bipartition to generate a binary clock-tree topology. Our algorithm can easily be extended to generate k-ary tree topologies by replacing bipartition with k-way partition. In each recursion step, a *partition graph* is created from the given clock sinks and their timing constraints. A min-cut algorithm is then used to partition the graph into two balanced graphs with minimum partition cost.

We first discuss practical design issues for clock-tree topology generation and define bipartition objectives. We then briefly review the steps of our algorithm. The details of each step are then presented.

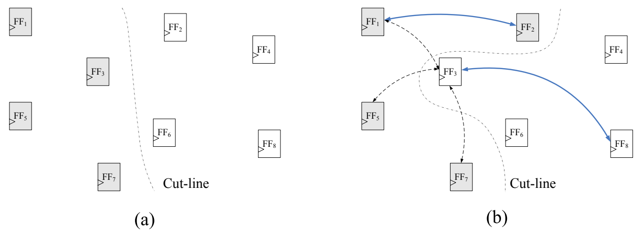

Figure 3: Total clock-skew uncertainty increases due to non-common clock distribution path reduction.

3.1 Topology Design Objectives

The principle for reducing the clock-skew uncertainty on a timing critical path is to reduce the non-common part of the clock distribution paths to its source and target clock sinks. However, the total clock-skew uncertainty as well as total clock power may increase if non-common clock distribution path reduction is not conducted carefully. Figure 3(a) shows the partitioning of eight clock sinks using a traditional balanced bipartition algorithm. The two solid arcs in Figure 3(b) show the two most timing critical paths of the circuit while the three dashed arcs show the timing paths with slightly more slacks. In Figure 3(b), FF_2 and FF_3 are moved to the left and right partitions to reduce clock-skew uncertainties on the most timing critical paths. However, this results in an increase on clock-skew uncertainties of s_{31}, s_{35}, and s_{37}. As a result, the partitioning in Figure 3(b) may actually be worse than the partitioning in Figure 3(a).

Based on the above observation, we believe that bipartition objectives should include the followings.

- **Balanced loading**
  Balanced loading ensures that no excessive snaking or buffering is required to balance the clock-delays of both partitions. This not only prevents unnecessary increase on clock power but also reduces the clock-delay of the final clock-tree, which in turn prevents an increase on total clock-skew uncertainty.

- **Small extra wire length**
  The minimum achievable clock-delay is likely to increase when the total wire length of a clock-tree increases by a large amount. Therefore, bipartitioning should not cause an excessive wire length increase.

- **Short non-common clock distribution paths**
  The clock-skew uncertainty between a pair of clock sinks is likely to decrease if the non-common clock distribution paths to both clock sinks are short. Therefore, clock sinks of timing critical paths should be partitioned into different groups as late as possible.

3.2 Overview

Our clock-tree topology generation algorithm is based on a recursive bipartition framework. In each iteration, multiple partition graphs are created and partitioned using a min-cut algorithm. The partitioning that has the lowest partition cost is then selected. Figure 4 shows the steps for determining the best partitioning in each iteration. Each step is explained in detail in following sections.

3.3 Reference Set

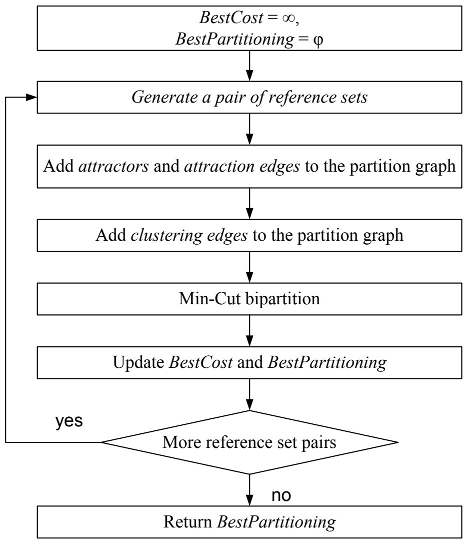
BestCost = ∞,
BestPartitioning = φ
Generate a pair of reference sets
Add *attractors* and *attraction edges* to the partition graph
Add *clustering edges* to the partition graph
Min-Cut bipartition
Update *BestCost* and *BestPartitioning*
More reference set pairs
yes
no
Return *BestPartitioning*

Figure 4: The steps in each bipartition iteration.

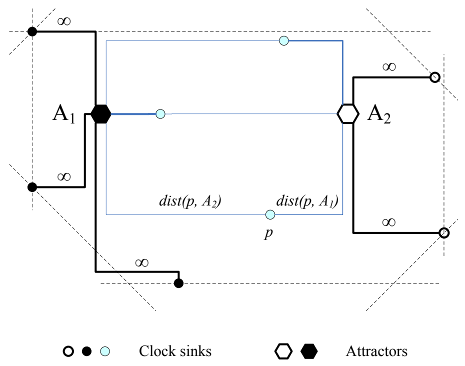

Figure 5: Illustration of bounding octagon, attractors and attraction edges.

A common technique for partitioning a set of nodes into multiple partitions is to first find the *reference set* of each partition [9] [2]. A reference set is a set of nodes that will most likely fall into the same partition. For example, the two clock sinks, which are far apart, are likely to belong to different partitions. Therefore, we can start with two reference sets where each set contains one of the two clock sinks. The remaining clock sinks can then be partitioned based on which reference set they are closer to.

Chao et al. [2] observe that the clock sinks that fall on the bounding *octagon* are usually partitioned into two groups with consecutive elements. Let S be the set of clock sinks to be partitioned and $Oct(S)$ be the set of clock sinks that are on the bounding octagon, there are $|Oct(S)|$ ways to create a pair of reference sets with $\lfloor \frac{1}{2}|Oct(S)| \rfloor$ and $\lfloor \frac{1}{2}(|Oct(S)|+1) \rfloor$ clock sinks. For each pair of reference sets, we create a partition graph and use a min-cut algorithm to find the best partitioning. Figure 5 shows the bounding octagon of eight clock sinks with $|Oct(S)| = 5$.

3.4 Attractor and Attraction Weight

To ensure that the clock sinks in a reference set are grouped together, we introduce two artificial *attractors*, A_1 and A_2,

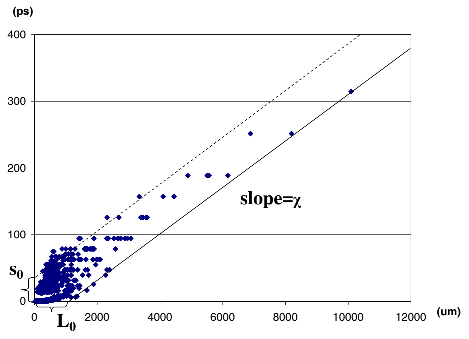

Figure 6: Diameter of S versus maximum clock-skew uncertainty in $s35932$.

for reference sets REF_1 and REF_2, then add an *attraction edge* with an infinite attraction weight from each reference set element to its attractor. For each of the remaining clock sinks, we create an attraction edge to each attractor. The attraction weight is calculated based on the distance measure from the clock sink to the reference set of the attractor. The distance measure should reflect the wire length increase if the node is grouped with the reference set. Chao et al. [2] use

$$dist(p, REF_k) = \min_{r \in REF_k} dist(p, r) + \max_{r \in REF_k} dist(p, r) \quad (3)$$

to measure the distance between clock sink p to reference set REF_k, where $dist(p, r)$ is the Manhattan distance between clock sinks p and r. We use the same metric as our distance measure.

One of the major improvement of our algorithm is that, instead of assigning clock sinks to reference sets greedily based solely on the distance measure, we keep the distance as the attraction weight. This enables wire length and timing constraints to be considered simultaneously as described in the next section.

We prefer to preserve the edges with smaller distance measure during partitioning. This can be done by using the negative distance measure for the edge weight. Alternatively, we use $dist(p, REF_2)$ and $dist(p, REF_1)$ as the edge weights for the edges from clock sink p to A_1 and A_2. Figure 5 shows the reference sets, attractors, and attraction edges of eight clock sinks.

3.5 Clustering Weight

The most important enhancement of our algorithm over previous algorithms is that we consider both clock sink positions and timing constraints for partitioning. By introducing a *clustering edge* between the two clock sinks of a timing path and controlling the clustering weight, we can control when those two clock sinks will be partitioned into different groups.

During the topology generation phase, it is difficult to predict what will be the clock-skew uncertainty on a clustering edge being partitioned. However, we can estimate which clustering edge can introduce a larger negative slack if it is partitioned. Let the *diameter* of S be the maximum distance of two clock sinks in S, or

$$Dia(S) = \max_{p,q \in S} dist(p, q). \quad (4)$$

Figure 6 shows $Dia(S)$ versus the maximum clock-skew un-

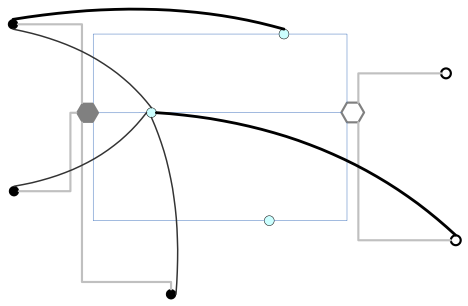

Figure 7: The partition graph after adding clustering edges.

certainty within S, where S is extracted from a zero-skew buffered clock-tree of $s35932$. The clock-skew uncertainty is approximated by 30% clock-delay of the non-common clock distribution path. It is shown that the minimum achievable clock-skew uncertainty increases linearly as the distance of two clock sinks increases. The minimum clock-skew uncertainty of a clock sink pair within $900\mu m$ is negligible.

From the above analysis, the *best-effort clock-skew uncertainty* of a clock sink pair is defined as follows.

$$\beta_{ij} = max\{0, \chi[dist(i, j) - L_0]\} \quad (5)$$

Likewise, the *worst clock-skew uncertainty* of S is approximated as follows.

$$\alpha(S) = s_0 + \chi Dia(S). \quad (6)$$

The parameters s_0, L_0 and χ depend on the clock sink density of the chip as well as interconnect and clock buffer parameters. They can be obtained by analyzing a zero-skew clock-tree generated by zero-skew buffered clock-tree optimization algorithms as illustrated in Figure 6.

Let the *equivalent slack* of s_{ij} be

$$u^k_{ij} = max(\delta^k_{ij}, \beta_{ij}), k \in \{h, s\}. \quad (7)$$

When $\delta^k_{ij} < u^k_{ij}$, a negative slack of $u^k_{ij} - \delta^k_{ij}$ is unavoidable. Therefore, the maximum clock-skew uncertainty that may be avoided by keeping a clustering edge is $\alpha(S) - u^k_{ij}$. Thus, the clustering weight function is defined as follows.

$$W^k_{ij} = max\{0, \frac{Dia(S)}{M} \times \frac{\alpha(S) - u^k_{ij}}{\alpha(S)}\}, k \in \{h, s\}. \quad (8)$$

We take the average of W^h_{ij} and W^s_{ij} as the clustering weight. It is also possible to put more weight on W^h_{ij} or W^s_{ij} to target specifically on negative hold-time or setup-time slack reduction.

The relative importance of timing constraints over wire length is controlled by M. The optimal M depends on how much TNS the design has and the amount of wire length increase designers are willing to tolerate. It can be determined empirically by running our algorithm repeatedly with an increasing M. In our experiments, we find that $M = 10$ gives good results without introducing too much extra wire length.

3.6 Min-Cut Bipartition

The set of clock sinks S, two attractors, attraction edges, and clustering edges form a *partition graph*. The complete

Circuit	# F.F.	s_0 (ps)	L_0 (μm)	χ $(ps/\mu m)$
s5378	263	10	500	0.036
s9234.1	286	10	550	0.044
s13207.1	852	10	700	0.041
s35932	2083	20	800	0.034
s38584.1	1768	20	900	0.045

Table 1: Analysis results of the clock-trees generated by a traditional balanced bipartition algorithm.

partition graph of the example circuit in Figure 1(b) is shown in Figure 7. The topology generation problem becomes a sequence of standard min-cut problems on partition graphs. The balanced loading objective is handled by assigning vertex weights according to clock sink capacitances (attractors have zero vertex weight) and enforcing a cut-ratio close to one. We use an efficient and publicly available partition tool, METIS [10], to find the cut line for a partition graph. The complete clock-tree topology is obtained by recursive bipartitioning.

4. EXPERIMENTAL RESULTS

We implement our algorithm in C++ and run it on a 1.7GHz 512MB Pentium-M laptop computer. We use SIS [11] to synthesize ISCAS89 benchmark circuits using a $0.13\mu m$ cell library, then use Dragon [12] for placement. We take the clock-tree topologies from both our algorithm and a traditional balanced bipartition algorithm [2] and construct zero-skew unbuffered clock-trees using the Deferred Merge Embedding algorithm [2]. The clock-trees are then optimized for minimum clock-delay using the zero-skew buffered clock-tree optimization algorithm from [5]. We assume gate-delays are independent Gaussian distributions with $(\mu, \sigma) = (50ps, 10ps)$. We impose a 6σ slack requirement for path-delay uncertainty and clock-skew uncertainty. For example, the slack requirement for a path with n gates is $6\sqrt{n}\sigma$. The 6σ slack requirement for clock-skew uncertainty is approximated by 30% clock-delay of the non-common clock distribution path.

4.1 Comparison and Analysis

Table 1 shows the parameters of the clock-trees generated by the traditional balanced bipartition algorithm. The analysis is done using the analysis method described in Section 3.5 and the parameters are used to guide our enhanced bipartition algorithm. We also found that in $s9234.1$, the TNS is contributed by only a few timing constraints. Moreover, many circuits have a predominant negative hold-time or setup-time slack.

Figure 8 shows the clock-trees of S13207.1 generated by traditional and enhanced bipartition algorithms. It can be seen that the local clock-tree structures generated by our enhanced bipartition algorithm are slightly different from that of the traditional method. For example, our algorithm sometimes will generate unbalanced clock load to reduce clock skew uncertainty. To balance the clock load at a higher level of the clock-tree, wire snakings (shown as dark segments) are sometimes required. However, this only increases the total wire length by a small amount ($< 6\%$).

As shown in Table 2, our algorithm achieves a TNS reduction from 2% $\sim$ 32% on ISCAS89 circuits. Although our algorithm also causes a slight increase on clock-tree wire

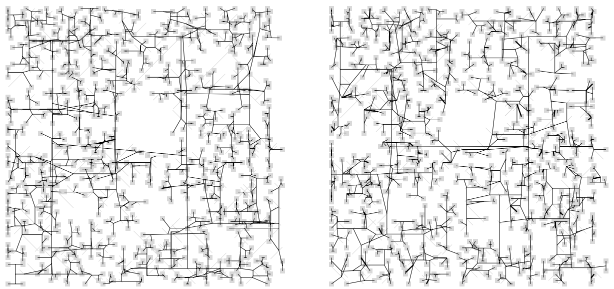

Figure 8: The clock-trees of S13207.1 using traditional and enhanced bipartitions.

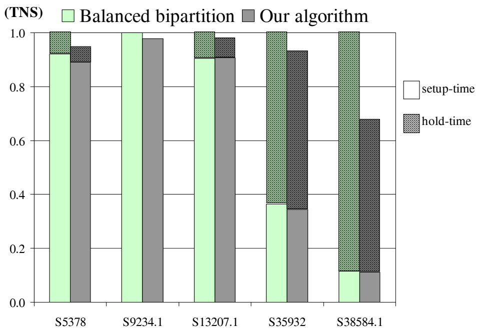

Figure 9: TNS improvement analysis.

length and switching capacitance, this does not necessarily increase the minimum clock-delay. In fact, we even achieve better clock-delays on $s9234.1$, $s13207.1$ and $s38584.1$. Therefore, wire length is more correlated to clock-power than to clock-delay.

Figure 9 shows the composition of the TNS using both the traditional balanced bipartition algorithm and our algorithm. We separate TNS into total negative hold-time and setup-time slacks, TNS^h and TNS^s, and normalize the results with respect to that from the traditional algorithm. It shows that our algorithm is especially effective in reduce TNS^h. This is because the source and target clock sinks of a hold-time critical path are usually close to each other and our algorithm is able to reduce clock-skew uncertainties on those paths without introducing much extra wire length. However, those of a setup-time critical path are usually separated far apart and there is not much room for improvements. The result that the total negative hold-time slack is easy to reduce is very exciting because hold-time violations are more serious than setup-time violations. While setup-time violations can be removed by increasing the clock-period CP (sell the chip at a lower frequency), hold-time violations can not be fixed after the chip is manufactured. Therefore, our algorithm is especially versatile for addressing hold-time problems in modern high speed designs.

4.2 Enhancements

In our experiments, we incorporate several heuristics to improve the results of our algorithm and discover a few areas that can provide further improvements.

4.2.1 Handling Reference Set Elements

Circuit	Delay (ns)		Cap. (pF)		Wire Length (mm)			TNS (ns)			CPU
	Old	New	Old	New	Old	New	Increase	Old	New	Reduce	New
s5378	0.218	0.218	10.89	11.45	38.018	40.245	5.9%	3.507	3.315	5.5%	31s
s9234.1	0.291	0.218	12.33	13.82	47.636	49.433	3.8%	6.124	5.989	2.2%	42s
s13207.1	0.436	0.363	33.19	35.52	120.651	124.752	3.4%	29.484	28.742	2.5%	118s
s35932	0.727	0.727	106.14	107.46	370.873	391.947	5.7%	478.915	445.95	6.9%	295s
s38584.1	0.872	0.727	84.71	87.32	321.803	329.340	2.3%	47.659	32.217	32.4%	255s

Table 2: Experimental results of a traditional balanced bipartition algorithm.

s38584.1	TNS^s (ns)	TNS^h (ns)	TNS (ns)	Improvement (%)
Traditional	5.480	42.197	47.659	-
Our(setup+hold)	5.277	26.940	32.217	32.4%
Our(hold only)	5.186	10.970	16.155	66.1%

Table 3: Further TNS reduction by targeting only hold-time constraints.

There can be a large number of clock sinks lying on one of the the bounding octagon edges. For example, many clock sinks may be placed on one edge of the chip. In this case, we only pick the first and the last clock sink on this edge into $Oct(S)$. This heuristic avoid creating a pair of reference sets that are dis-proportional in their diameters.

Setting the clustering weight to infinity for edges from reference set elements to their attractors can cause unsatisfactory results. This is because we do not take timing constraints into account when we break $Oct(S)$ into two reference sets. A simple heuristic provides significant improvement: We determine reference sets as described in Section 3.3 and treat clock sinks in $Oct(S)$ the same way as we treat other clock sinks in S.

4.2.2 Targeting Hold-Time Constraints

We have shown that negative setup-time slacks are more difficult to reduce through clock-tree topology generation. Therefore, clustering edges and weights for setup-time constraints can introduce redundant information to the partition graph and degrade the partitioning quality. We conduct a separate experiment on $s38584.1$ by considering only hold-time constraints. As shown in Table 3, this improves the TNS reduction from 32.4% to 66.1%.

4.2.3 Approximate Worst Clock-Skew Uncertainty

In this paper, we use a simple linear approximation to estimate the worst clock-skew uncertainty of S based on s_0, χ and $Dia(S)$. This can be too optimistic in some cases and too pessimistic in other cases. It is possible to use both the number of clock sinks and clock sink density besides $Dia(S)$ to give better approximations and results.

4.2.4 Alternative Min-Cut Algorithms

We found that sometimes METIS does not produce very good partitionings. In other words, clock sinks may be grouped to wrong partitions and cause unnecessary wire length increases. We are currently incorporating MLPart [13] with our code and we expect to see an improvement on wire length.

4.3 Runtime

Although min-cut problems are NP-Complete problems, it is usually solved very efficiently with near-optimal partition costs. For each bipartitioning, we write the partition graph into a file, call METIS for partitioning, and read the results back into our program. Even with the high disk I/O overhead, the runtime of our algorithm is still less than five minutes for the largest circuit. This runtime is much smaller than the runtime for the subsequent clock-tree optimization step and our algorithm scales well for large problems.

5. CONCLUSION AND FUTURE WORK

We present a novel partition-based clock-tree topology generation algorithm. The algorithm takes into consideration both positions of clock sinks and timing constraints. As a results, the total negative slack is significantly reduced. In the future, we plan to investigate techniques for total negative slack reduction during clock-tree optimization for further timing-convergence improvements.

6. REFERENCES

[1] R.-S. Tsay. Exact zero skew. In *Proceedings of the international conference on Computer-aided design*, pages 336–339, 1991.

[2] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and A.B. Kahng. Zero skew clock routing with minimum wirelength. *Circuits and Systems II: Analog and Digital Signal Processing*, 39(11):799–814, Nov. 1992.

[3] I-Min Liu, Tan-Li Chou, Adnan Aziz, and D. F. Wong. Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion. In *Proceedings of the international symposium on Physical design*, pages 33–38. ACM Press, 2000.

[4] Jeng-Liang Tsai, Tsung-Hao Chen, and Charlie Chung-Ping Chen. Epsilon-optimal minimum-delay/area zero-skew clock-tree wire-sizing in pseudo-polynomial time. In *Proceedings of the international symposium on Physical design*, pages 166–173. ACM Press, 2003.

[5] Jeng-Liang Tsai, Tsung-Hao Chen, and Charlie Chung-Ping Chen. Zero-skew clock-tree optimization with buffer-insertion/sizing and wire-sizing. *IEEE Transactions on Computer-aided design*, 23(4):565–572, April 2004.

[6] Dimitrios Velenis, Eby G. Friedman, and Marios C. Papaefthymiou. A clock tree topology extraction algorithm for improving the tolerance of clock distribution networks to delay uncertainty. In *Proceedings of the international symposium on Circuits and systems*, pages 4.422–4.425, 2001.

[7] Dimitrios Velenis, Marios C. Papaefthymiou, and Eby G. Friedman. Reduced delay uncertainty in high performance clock distribution networks. In *Proceedings of the conference on Design, Automation*

and Test in Europe, page 10068. IEEE Computer Society, 2003.

[8] Anand Rajaram, Jiang Hu, and Rabi Mahapatra. Reducing clock skew variability via cross links. In *Proceedings of the 41st annual conference on Design automation*, pages 18–23. ACM Press, 2004.

[9] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57. ACM Press, 1984.

[10] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[11] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. *Electronics Research Laboratory, Memorandum No. UCB/ERL M92/41*, 1992.

[12] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Dragon2000: standard-cell placement tool for large industry circuits. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 260–263. IEEE Press, 2000.

[13] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Improved algorithms for hypergraph bipartitioning. In *ASP-DAC '00: Proceedings of the 2000 conference on Asia South Pacific design automation*, pages 661–666. ACM Press, 2000.