Evolving Challenges and Techniques for Nanometer SoC Clock Network Synthesis

Subhendu Roy^1, Pavlos M. Mattheakis^2, Laurent Masse-Navette^2, David Z. Pan^1
1. Department of Electrical and Computer Engineering, University of Texas at Austin, USA
2. Mentor Graphics, Grenoble, France
subhendu@utexas.edu, {pavlos_matthaiakis, Laurent_Masse-Navette}@mentor.com, dpan@ece.utexas.edu

***Abstract*—With continued technology scaling, increased variability effects and growing design complexity, the problem of clock network synthesis is becoming more challenging. In this paper, we discuss the key issues encountered while synthesizing the clock network. Furthermore, we present a clock tree resynthesis methodology to address some of the above challenges. It involves incremental modification on an already synthesized/routed clock tree for multi-corner multi-mode timing closure and has been validated on industrial designs using cutting-edge technology nodes.**

I. Introduction

Clock Network Synthesis (CNS) is a fundamental problem in physical design to synchronize sequential elements (sinks) in the design. The inaccuracy in clock synchronization is quantified by clock skew which is defined as the maximum difference in clock arrival time between any two sinks. Clock network may be a tree or non-tree structure. Historically, clock tree has been widely used for ASICs, but clock meshes may be used for better robustness at the cost of power overhead [1][2], and microprocessors have used clock meshes. The traditional Clock Tree Synthesis (CTS) problem can be abstracted to a single-net buffering problem with the objective of minimization of clock skew using minimum buffer/routing resources. A lot of work has been done in the past targeting global zero skew [3][4][5], but that requires a large area overhead and insertion delay (clock source to sink delay) for large-scale designs. Industry-tools typically try to minimize the buffer/routing resources with the constraint of a given skew and insertion delay margin.

With aggressive technology scaling, the main challenge in CNS for modern designs is to cope with (i) variation effects, (ii) design complexity and (iii) low power objective. Due to wide variations from chip to chip, multi-corner analysis has become more tedious [6][7]. In addition, on-chip-variation (OCV) considers the local variations within a chip, and OCV-impact has increased significantly in advanced technology nodes. These variations have intrusive impact in controlling clock skew. Furthermore, there has been a paradigm shift from one clock to multiple clock domains or one mode to multiple modes (driven by architectural choices) in modern SoC designs, which creates difficulty in clock balancing. Finally, low power is the most important concern in current VLSI design industry. Several techniques, such as clock gating [8][9], multi-V_{dd} design [10], etc., have been employed to reduce power. But these techniques are disruptive to meeting the CTS objectives and massive clock gating is one of the main reasons for steering away the CTS problem from single net buffering problem with uniform balancing constraint to the problem of multiple clock domains with prescribed clock arrivals among groups of clock pins.

In spite of careful optimization at every step after synthesis, timing violations still persist after placement/routing. Useful clock skew optimization is an effective approach to achieve the timing closure and design convergence [11][12][13][14]. Most of the existing approaches on useful clock skew optimization do not account for implementability of the clock schedules at the later stages of design cycle. Useful clock skew has been implemented in deep routing stage for multi-corner, multi-mode (MCMM) industrial designs in [15], although the approach is local, and not practical for high-performance time-constrained real designs. A recent work [16] tackles the useful clock skew optimization in large-scale industrial designs in a two step approach. The first phase is based on a skew scheduling engine, which calculates the clock schedules at the clock pins after a clock tree has been synthesized and routed. Then this schedule is implemented, resulting in significant timing improvement with marginal resource overhead in the clock tree.

The rest of the paper is organized as follows. Section II illustrates the evolving issues in CNS for modern designs. Section III presents a clock resynthesis methodology to tackle some of these challenges with conclusion in Section IV.

II. Evolving CNS Issues

CNS challenges have evolved from time to time in the last few decades. Initially, minimization of clock skew was the primary objective. But the area/power cost to target zero clock skew is very high. In addition, it limits the maximum operating frequency. So there has been a paradigm shift from zero clock skew to useful clock skew optimization and skew margin. The key challenges for CNS/CTS in modern SoC designs include the following.

A. Variations

Several works targeted chip level clock tree synthesis to tackle clock divergence issues across various corners [17].

But it does not consider any timing information of the data-path. [18] resolves the clock skew scheduling problem in presence of process variations by ILP formulation. Link-insertion technique has been proposed to build variation tolerant clock network synthesis [19][20]. TSV (Through-Silicon-Via) induced stress could cause timing corner mismatch, and is another challenge in 3D clock tree synthesis which has been addressed in [21].

However, it is difficult to model OCV-impact to guide clock tree synthesis. Typically in industrial timers two types of clock arrivals are calculated for each pin, namely early arrival and late arrival to accommodate OCV, and are characterized by OCV-derates [22]. For instance, suppose one level of buffer delay is 60ps and OCV-derates are 0.95-1.05. Then the early delay (late delay) for that level would be $60 \times 0.95 = 57$ps ($60 \times 1.05 = 63$ps). As OCV-derates increase, *i.e.*, derates become more apart from 1.00, it becomes difficult to achieve the timing closure. Also, accurate estimation of OCV is impossible to perform unless the clock tree is synthesized, because the effect of OCV can not be accounted for unless launch/capture path branching points are known. Furthermore, the advanced OCV analysis, which depends on the path length from clock source to sink, relies on the clock tree routing as well. So it is imperative to incrementally modify the clock tree even after clock tree synthesis and routing to tackle OCV-impact.

B. Low Power Objective

Clock nets switch frequently, and as a result the switching power is typically very high for clock nets. 50-70% of total dynamic power of the design can be consumed in the clock network for lower technology nodes [23]. Clock gating is one of the most efficient techniques to reduce power dissipation in the clock network. Several types of clock gating, such as RTL gating, data-driven clock gating [8] or lookahead clock gating [9], have been proposed. In data-driven gating, sequential elements are gated if the D and Q-signals match. An example of data-driven clock gating is shown in Fig. 1. The D-pin and Q-pin output are XOR-ed to generate the enable signal for the Integrating Clock Gating (ICG) cell. But this costs area overhead and needs additional timing margin in the data-path. Recently, the concept of lookahead clock-gating has been introduced [9] which avoids the constraint of timing margin with more resource overhead. For RTL clock gating, enable signals need to be derived. From CTS perspective, this adds constraint on timing margins, as the group of flip-flops driving the enable logic need to be balanced. These groups are termed as skew groups.

Clock gating can also impact the extent of NBTI (Negative Bias Temperature Instability) induced threshold-voltage degradation in the clock buffers leading to increase in clock skew, which is tackled in [24] by a practical design-time technique. Multi-bit-flip-flop (MBFF) is also an effective technique for low power clock network design, and simultaneous consideration of MBFF and clock gating is done in [25] for clock network optimization.

Multi-V_{dd} design [10] is an alternative for power reduction. Level shifters and isolation cells are generally needed for this

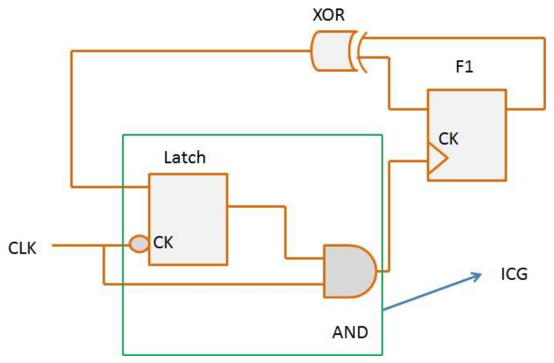


Fig. 1. Data-driven Clock Gating

adding complexity and latency. Clock tree balancing across different power domains is also a challenging task for the designers. In addition, a chip can operate at several modes to reduce power dissipation. For instance, a design can be in active or sleep mode when performance or power is the main concern respectively. To handle multiple power modes, functional timing paths across all active modes need to be analyzed for timing closure adding complexity to CTS.

C. Design Complexity

In order to cope with Moore's law, design size is increasing day-by-day. So flat-CTS for designs with multi-million flip-flops is no more feasible, and the way-out is the hierarchical clock gating structures with tens of thousands of clock gators. Clock-multiplexers and pre-existing structures such as isolation/level-shifter cells also need to be inserted respectively to support multi-mode and multiple power domains. In addition, physical constraints such as blockages/macros create issues in placing and routing the clock buffers.

With growing design complexity, variation effects and low power objective, timing closure has become very challenging job for today's multi-corner, multi-mode designs. Clock tree aware placements are performed in [26][27] to minimize wire-length and switching power. A few studies have been conducted on timing optimization during placement and routing as well [28][29][30]. [31] presents a novel algorithm for permuting latch positions and sizes in the late stages of design flow for timing closure. But in spite of all these efforts, timing violations persist in deep routing stage and the designers need to manually fix those violations for every corner/mode combination. As a result, data-path aware clock scheduling, that too in the deep routing stage, has become an inevitable step to meet stringent delivery targets.

III. Clock Tree Resynthesis

Performing simultaneous optimization of timing and power in presence of variation effects by building the clock network from scratch is a very difficult task, if not impossible. So post-CTS optimizations are essential to meet timing closure. Such optimizations, such as datapath optimization, at the post-CTS stage typically cost very high area/power overhead. This calls for clock tree resynthesis, *i.e.*, incremental clock tree modification after the clock tree has been synthesized. There

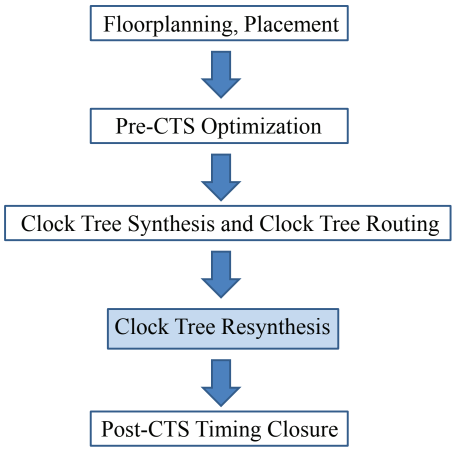


Fig. 2. Clock tree resynthesis in a conventional back-end flow [16]

are a few works on this. [32] formulates an LP problem to optimize clock period by bounded delay buffering at the leaves of the clock trees, *i.e.*, the input pins of the sequential flip-flops. Another work [15] has focused on useful clock skew realization in MCMM designs locally by adding/removing buffers. The key limitations in these approaches are (i) the high area cost, since buffers are added at the leaf level, and (ii) implementability as it is difficult to place and route many buffers in an already synthesized/routed clock tree of modern space-constrained designs.

To address this, a new clock tree resynthesis methodology was proposed in [16]. Fig. 2 shows how this methodology fits into the conventional back-end flow. Instead of calculating clock schedule at the leaf-level registers, it computes the offsets in clock arrival at the clock tree driver pins of a placed design with already synthesized and routed clock tree. Offsets can be positive and negative, translating to respectively delay and acceleration in clock arrival at the clock tree pins. For instance, a positive offset of 50ps at a clock driver pin p means the clock arrival at p needs to be delayed by 50ps. An LP engine [33] is used to estimate these offsets which consider multi-mode, multi-corner scenarios. The computation of offsets is also OCV-aware, and it is realizable since the LP engine has the prior knowledge of depth and path-lengths of the individual clock sinks in the post-CTS and post-routed clock tree. The offsets are discretized in terms of intrinsic buffer delay or the minimum delay of the buffer in the clock tree and the LP engine is capable of restricting the levels of offsets.

The LP engine can only compute the offsets and predict the improvement in the timing metrics, such as total negative slack (TNS) or worst negative slack (WNS), if those offsets can be accurately realized in the targeted synthesized/routed clock tree. Fig. 3 presents the scheme to implement the positive offsets. A positive offset d_{off} at the output pin *op* of the buffer B_1 can be realized by inserting a delay element D (may be a buffer or a chain of buffers). Although this approach has some area overhead, it is non-disruptive to the other parts of the clock tree as there is no impact of D on the siblings of B_1. However, difficulty arises when the negative offsets are exercised. This is because the negative offsets can be realized either by sizing or restructuring the clock tree, and not by inserting any extra delay elements like in case of positive offsets. But these would have negative effects on the timing profile of the rest of the clock tree and can possibly cause severe degradation on the TNS/WNS of the design.

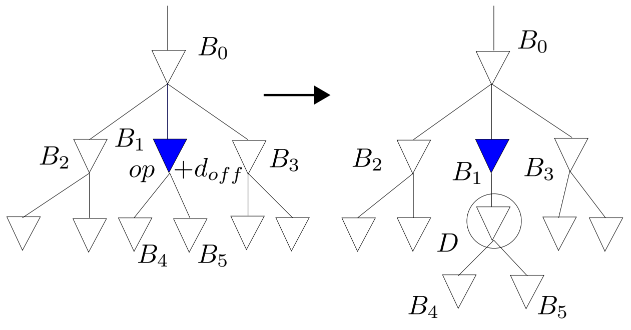


Fig. 3. Positive offset realization [16]

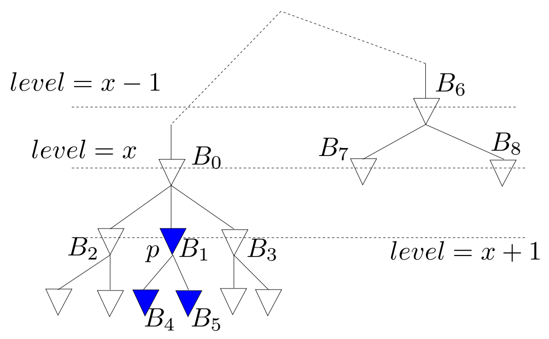
In this context, the capability of the LP engine to restrict the levels of offsets is exploited. Since more the negative offset, more is the difficulty to implement it, a set of experiments are performed with this LP engine on four industrial designs to bound the offset range and a conclusion is derived as potential TNS gain is possible by realizing many levels of positive offsets with one level of negative offset in conjunction.

Clock tree restructuring and sizing are performed to realize this one level of negative offset. It should be noted that restructuring is done within the scope of a hyper-net to guarantee that the clock gating functionality is restored. A hyper-net is a set of logically equivalent or opposite polarity nets separated by buffers/inverters in the same physical partition as the root clock driver of the top net, and essentially connected in a tree-topology.

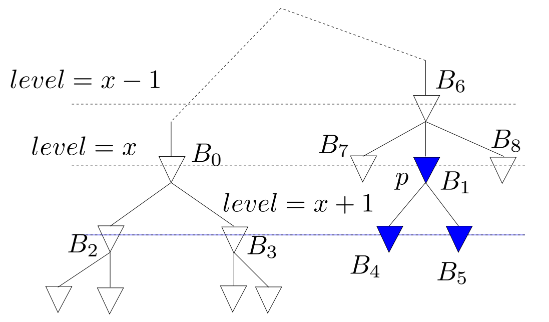
The key strategy for clock tree restructuring is the utilization of the positive slack. If all sequential cells in the transitive fanout (TFO) cone of a clock tree driver pin have positive slack (more specifically slack at the Q-pin), then that driver pin is annotated as a potential acceptors for a pin with negative offset. So for each of the pins annotated with negative offset, the potential acceptors are explored which could give necessary speed-up in the clock arrival, and the best move in terms of accurately realizing negative offset is accepted with backtracking mechanism. Any move which introduces DRC violation is discarded. The geometrical proximity of the moving pin and the acceptor pin is considered in order to avoid any extra buffering which could increase the area. To identify the potential acceptor pins, a slack manager is developed to track the slack values and this is updated after each restructuring.

An example is illustrated in Fig. 4, where the pin p of the buffer B_1 is annotated with a negative offset. With the slack manager, the potential acceptor pins are extracted and it comes out (from the best cost) that the output pin of B_6 could be the best insertion point. So a restructuring is performed by detaching B_1 from B_0's fanout and connecting it to B_6. Fig. 4(a) and 4(b) respectively show the clock tree before and after the restructuring.

This methodology is applied on several industrial designs



(a) Clock tree hyper-net where p has negative offset of 1 clock tree level.



(b) Resultant clock tree hyper-net where the negative offset at p is realized by restructuring.

Fig. 4. Negative offset realization example [16]

using cutting-edge technology nodes (20-32nm). An average improvement of 57%, 12% and 42% in TNS, WNS and failure-end-point (FEP) respectively is achieved with an average clock tree area overhead of 26%. The baseline or the initial timing metrics are after placement, clock tree synthesis and routing by an industrial tool. With only one level of negative offset realization (no positive ones), the corresponding improvements in TNS, WNS, FEP are respectively 16%, 1% and 12% with average clock tree overhead less than 2%. Results show that the negative offset realization is area-efficient, but the improvement in timing metrics is less, while coupling this with positive offset realization results to a significant improvement in timing but at the cost of the area overhead.

IV. Conclusion

The problem of clock network synthesis has evolved over time, initially from building single clock net buffering problem with zero skew to OCV-aware useful clock skew optimization targeting low power in multi-corner multi-mode designs spanning multiple clock domains. In this paper, we have discussed several key challenges in modern clock network synthesis and presented a clock resynthesis methodology [16] to address some of those issues. We anticipate more research in this area, such as OCV-aware data-clock co-optimization to achieve timing closure in MCMM designs with low area/power overhead.

References

[1] A. Rajaram and D. Z. Pan, "Meshworks: An efficient framework for planning, synthesis and optimization of clock mesh networks," *ASPDAC*, pp. 250–57, 2008.

[2] M. Cho, D. Z. Pan, and R. Puri, "Novel binary linear programming for high performance clock mesh synthesis," *International Conference on Computer Aided Design*, pp. 438–43, 2010.

[3] R. Tsay, "An exact zero skew clock routing algorithm," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, pp. 242–249, 1993.

[4] J. L. Tsai, T. H. Chen, and C. C. Chen, "Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, pp. 565–572, 2006.

[5] K. D. Boese and A. B. Kahng, "Zero skew clock-routing trees with minimum wirelength," *ASIC Conference and Exhibit*, pp. 17–21, 1992.

[6] Y. Taur and D. Buchanan, "CMOS scaling in nanometer regime," *Proc. IEEE*, pp. 486–503, 1997.

[7] V. Mehrotra and D. Boning, "Technology scaling impact of variation on clock skew and interconnecet delay," *Interconnect Tech. Conference*, pp. 4–6, 2001.

[8] S. Wimer and I. Koren, "Design flow for flip-flop grouping in data-driven clock gating," *IEEE Transactions Very Large Scale Integration (VLSI) Systems*, pp. 771–78, 2014.

[9] S. Wimer and A. Albahari, "A look-ahead clock gating based on auto-gated flip-flops," *IEEE Transactions on Circuits and Systems-1*, pp. 1465–72, 2014.

[10] M. D. F. Wong, "Low power design with multi-vdd and voltage islands," *International Conference on ASIC*, p. 1325, 2007.

[11] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, pp. 945–51, 1990.

[12] R. Deokar and S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," *ISCAS*, pp. 407–10, 1994.

[13] L. F. Chao and H. M. Sha, "Retiming and clock skew for synchronous systems," *ISCAS*, pp. 283–86, 1994.

[14] X. Liu, M. C. Papaefthymiou, and E. G. Friedman, "Maximizing performance by retiming and clock skew scheduling," *DAC*, pp. 231–36, 1999.

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

[16] S. Roy, P. M. Mattheakis, L. Masse-Navette, and D. Z. Pan, "Clock tree resynthesis for multi-corner multi-mode timing closure," *ISPD*, pp. 69–76, 2014.

[17] A. Rajaram and D. Z. Pan, "Robust chip-level clock tree synthesis for SOC designs," *DAC*, pp. 720–723, 2008.

[18] V. Nawale and T. W. Chen, "Optimal useful clock skew scheduling in the presence of variations using robust ILP formulations," *International Conference on Computer-Aided Design*, pp. 27–32, 2006.

[19] T. Mittal and C. K. Koh, "Cross link insertion for improving tolerance to variations in clock network synthesis," *ISPD*, pp. 29–36, 2011.

[20] A. Rajaram and D. Z. Pan, "Variation tolerant buffered clock network synthesis with cross links," *ISPD*, pp. 157–64, 2006.

[21] J. Yang, J. S. Pak, X. Zhao, S. K. Lim, and D. Z. Pan, "Robust clock tree synthesis with timing optimization for 3D-ICs," *ASPDAC*, pp. 621–26, 2011.

[22] J. Bhaskar and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach.* Springer, 2009.

[23] D. Liu and C. Svensson, "Power consumtion estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuits*, pp. 663–70, 1994.

[24] A. Chakraborty and D. Z. Pan, "Skew management of NBTI impacted gated clock trees," *ISPD*, pp. 127–33, 2010.

[25] S. C. Lo, C. C. Hsu, and M. P. Lin, "Power optimization for clock network with clock gate cloning and flip-flop merging," *ISPD*, pp. 77–84, 2014.

[26] D. Lee and I. L. Markov, "Obstacle-aware clock-tree shaping during placement," *ISPD*, pp. 123–130, 2011.

[27] Y. Wang, Q. Zhou, X. Hong, and Y. Cai, "Clock-tree aware placement based on dynamic clock-tree building," *ISCAS*, pp. 2040–43, 2007.

[28] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhury, and B. Halpin, "Timing driven force directed placement with physical net constraints," *ISPD*, pp. 60–66, 2003.

[29] Y. Liu, R. S. Shelar, and J. Hu, "Delay-optimal simultaneous technology mapping and placement with applications to timing optimization," *International Conference on Computer-Aided Design*, pp. 101–106, 2008.

[30] S. W. Hur, A. Jagannathan, and J. Lillis, "Timing driven maze routing," *TCAD*, pp. 234–241, 2000.

[31] S. Held and U. Schorr, "Post-routing latch optimization for timing closure," *DAC*, 2014.

[32] J. Lu and B. Taskin, "Post-CTS clock skew scheduling with limited delay buffering," *International Midwest Symposium on Circuits and Systems*, pp. 224–227, 2009.

[33] V. Ramachandran, "Functional skew aware clock tree synthesis," *ISPD*, 2012.