International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)
Manuscript Received: 02 January 2022, Received in Revised form: 01 February 2022, Accepted: 05 February 2022 DOI: 10.46338/ijetae0222_10

Clock-Latency-Aware Pre-CTS for better Timing Closure in VLSI Design

Jalal Benallal[1], Abdellah Bouamama[2], Lekbir Cherif[3], Mohammed Darmi[4], Rachid Elgouri[5], and Nabil Hmina[6]
[1,2,6]*Sultan Moulay Slimane University, Med V Av, B.P 591, Béni Mellal, Morocco.*
[2,3]*Mentor Graphics, a Siemens business, Rabat, Morocco.*
[4,5]*Laboratory of Advanced Sciences Engineering, ENSA, Ibn Tofail University, B.P 241, Kenitra, Morocco.*

***Abstract*— A common strategy for reducing the time jump due to a clock tree insertion is to add a clock uncertainty value. If a small clock uncertainty value is selected for the Pre-CTS optimization, then a significant timing jump is observed when the clock tree is inserted, and clock timings are propagated. On the other hand, if the clock uncertainty is large, then the place and route (PNR) flow does not converge. This approach has been proven to be insufficient as each timing path is affected differently by the creation of the clock tree. In this paper, we suggest a more targeted approach whereby all pins in the clock tree will be annotated with their estimated latency, and therefore clock arrival times under ideal clocks should closely model the post-CTS arrival times. This objective is achieved by introducing an accurate and fast clock prediction early in the flow at the pre-CTS stage. As a result, the transition from the pre-CTS stage to the post-CTS stage becomes easier without significant timing jumps. An experiment on nine commercial test cases led to a significant TNS timing improvement of up to 68%, with an average of 35% at the end of the route stage. The full PNR flow runtime is not impacted. We achieved a 3% average runtime reduction over all test cases.**

***Keywords*—Clock-latency estimation, Clock-tree Synthesis, Integrated circuit conception, Physical implementation, Static timing analysis, Static timing closure.**

I. Introduction

Achieving high performance with new very large-scale integration (VLSI) designs is a complicated and delicate task. As the functional frequency of operation of the digital designs increases, the need for a robust clock tree synthesis (CTS) process is more evident. This is especially true when timing constraints are complex and combined with technology process variations, such as on-chip-variation (OCV) effects [1-3]. Additionally, a significant rise in the impact of uncertainties, such as process, voltage, or temperature variations [4], and the development of new applications, such as the Internet of Things, pose new challenges to IC design [5-7].

Thus, EDA tools must take all of that into account when working towards better timing convergence.

To reach high frequency requires taking into account various new challenges, so timing convergence counts on many techniques. Thomas and Kiran explained some techniques to optimize paths with negative timing slacks such as cell resizing, placement optimization, and routing optimization [8]. For an optimized clock tree, they presented two methods for further timing closure: clock pushing for setup and clock pulling for hold optimizations. Jhao et al. [9] introduced a new method at the route stage for timing closure by playing with the clock skew. Kao et al. [10] proposed an OCV-aware clock tree. Liu et al. [11] developed a CTS flow for latency minimization. Farshidi et al. [12] described an optimal method to solve the clock network buffer sizing, considering the power consumption and clock skew. Another study done by Lu and Taskin [13] proposed a post-CTS solution by adding delays in the clock pins, making a nonzero clock skew that helps with slack reduction.

From a design point of view, recent studies [2,14] have proposed new solutions reducing the complexity of clock generator architectures. The objectives are area, latency, skew, OCV impact, delay, and power reductions. However, even with an optimized clock generator bloc, we will always have clock gating structures that need special care at the integrated circuit (IC) physical implementation phase.

The above optimizations are carried out at the CTS and post-CTS phases, or at the end during the route stage. In many cases, this flow has some limitations because it can leave difficult timing violations unresolved at the end of the place and route (PNR) flow. That said, any early prediction of the properties of the clock tree network is essential for better time closure.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

This paper introduces a new method to calculate clock latencies on each clock pin using estimated clock latencies annotations. The goal is to replace slack margin overrides with accurate clock tree latency estimations. Every pin in the clock tree will be annotated with its expected latency; therefore, the clock arrival times under ideal clocks will closely model the post-CTS arrival times. The experimental results from the nine test cases show the great benefit of the new method compared to previous ones. Latencies comparisons between estimated and actual values are very close, and the global QoRs are better.

This paper makes the following contributions:

- Starting from the baseline using the existing method, we have executed the full PNR flow and collected all metrics as references.
- We executed the same PNR flow after enabling the new clock estimation method at the pre-CTS stage.
- We prove the benefit of the new method by running both flows on nine different commercial use cases made with diverse advanced technology nodes.

This paper is organized as follows. The method is described in Section 2. Section 3 details the new technique to incorporate clock latency estimation at the pre-CTS stage. Section 4 presents and analyzes the experimental results. Finally, Section 5 concludes the paper.

II. Background Of Clock Latency Annotations

During the pre-CTS phase, ideal clocks are enabled, which means that the arrival times for all clock signals are zero or have constant values in timing constraints. This is a realistic way of measuring the slack for paths between any two clock tree leaves, because the CTS will ensure that both leaves have the same arrival time (even if it is greater than zero).

However, clock-gating cells (CKG) are usually higher in the clock tree than leaves and, thus, have earlier clock arrival times. For paths that start or end on such intermediate clock pins, ideal clocks are not enough to correctly model the slack (see Fig. 1).

After the CTS, as shown in Fig. 2, when clock propagation is enabled, the slack of the flip-flop to flip-flop paths remains almost unchanged (changes, if any, are due to clock tree skew). However, the slack of the flip-flop to clock-gating path is completely different. The clock arrives much earlier to the clock gate than to the other flip-flops.

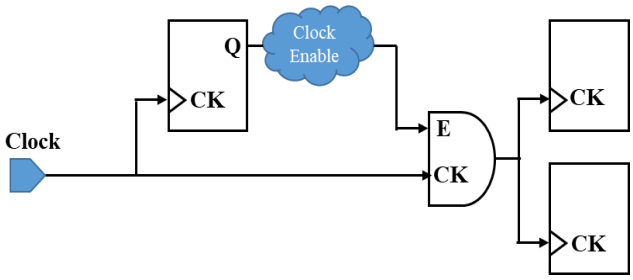

Fig. 1. Pre-CTS clock gating timing path.

In the figure above:

CK is the flip-flop's input clock pin.
E is the flip-flop's input data enable pin.
Q is the flip-flop's output data pin.

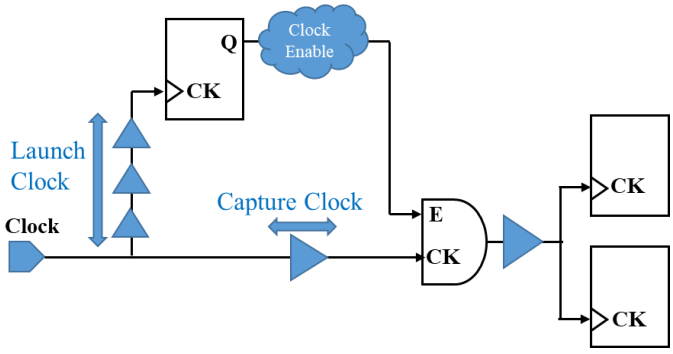

Fig. 2. Post-CTS clock gating timing path.

The solution to resolve clock-gating timing issues at the pre-CTS stage is to add the clock latency estimation. However, inaccuracies in estimation may have significant unwanted effects and waste time for optimizing noncritical paths, which could potentially lead to increased area, power, and congestion.

To address those problems and have an accurate slack estimate of clock-gating paths, we introduced a mechanism to annotate the expected clock tree latencies before CTS. This means that the timer will have information on the expected delays of the clock network before synthesis and will be able to compute a more accurate slack value for problematic paths.

Clock latency annotations using the "slack margins" approach already exists in recent EDA tools such as Nitro-SoC 2019.1 [15]. To fix clock-gating path slack issues, the process specifically combines:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

- A clock tree latency estimator based on a series of heuristics.
- Slack margin overrides.

The clock latency estimator provides the expected delay of every clock net in the design. Then, and only for clock-gating paths, a slack margin override is set as shown in Fig. 3.

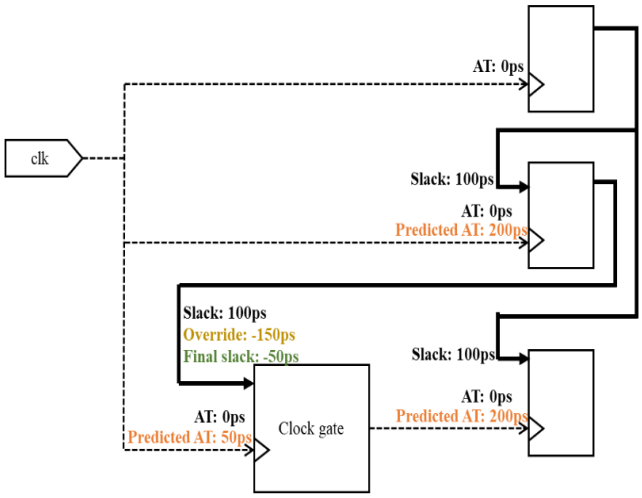

Fig. 3. Slack margin overrides-based clock annotations.

However, this method has two main limitations:

- Only one slack margin can be set for a specific end-point, and those can override the ones set by the user.
- Currently, it also produces erroneous results when the source endpoint is not on a clock tree leaf.

III. Clock Latency Annotations At The Pre-CTS Stage

Estimated clock latency annotations are a mechanism to estimate the latency of clock networks before clock tree synthesis. The estimation only reflects the expected latency of the clock nets (i.e., the delay from clock source to sync pin).

There are two versions of the latency estimator:

- *Pre-cell-placement:* used before global placement. The locations of clock sources and clock leaves are unknown. We estimate the total clock wire length based on the entire partition.
- *Post-cell-placement:* used after global placement. The locations of clock sources and clock leaves are used to estimate the clock wire length.

The latency estimator uses the following data as inputs:

- Clock sources and clock markings as provided by the timer.
- With pre-cell-placement estimator only: Size of partitions containing clock nets.
- With post-cell-placement estimator only: Locations of clock sources, clock leaves, and fixed clock gates.
- Global CTS constraints such as: List of repeaters, max transition, max fanout, nondefault rules (NDRs), CTS corner, layer capacitance, resistance, etc.

The core algorithm is divided into the following stages:

1. *Pin graph constructor:* Construct the pin graph from the input netlist. We start at the clock roots and stop whenever we reach a pin outside the clock marking.
2. *Assign delays:* For every arc in the pin graph, we estimate the delay. We distinguish between cell arcs and net arcs. The latency estimator has to consider the delays caused by cells already present in the clock tree.
3. *Compute latencies:* Compute the latency at every clock pin.

By applying the new algorithm on the schema in Figure 3, new estimated latencies will be annotated on clock pins as shown in Fig. 4.

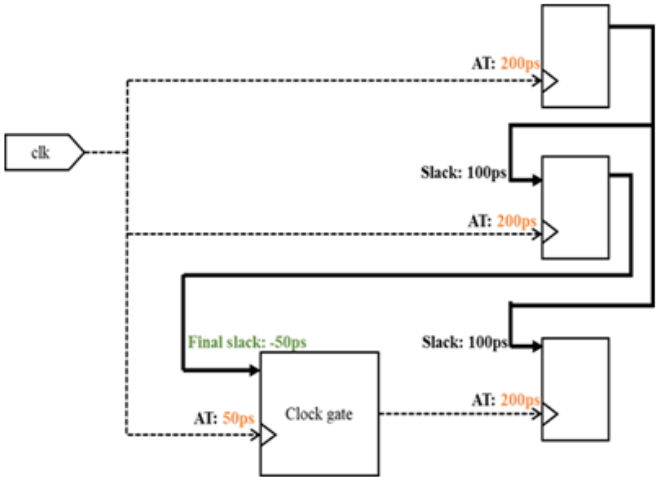

Fig. 4. Estimated clock-based clock annotations.

As we can see, this means that effectively STA report commands will now display an "ideal network clock latency" different from zero, just like if the user had done a "set_clock_latency" command. However, this latency will be different for intermediate clock tree pins.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

The following example is a clock-gating path. Both reports are done with ideal clocks. Unlike the timing report in Fig. 5, the timing report in Fig. 6 is done with annotations. The clock network latency for the source endpoint is 501.0 ps (the leaf flip-flop ff1) while the target clock network latency is 408 ps (cg2, clock gate, an intermediate clock tree pin). Thus, the launch path latency increases by about 100 ps compared to the capture path latency, and the slack is made negative consequently.

Accurate delays and slacks have a positive impact on cell placement because the estimated placement of the CKG cells is now based on knowledge of the clock tree network and clock pins' latencies.

Without annotations

```
Nitro-SoC> report_timing -to cg2/E
 Timing Path to cg2/E

 Path Start Point : ff1 (DFFQX1)
 (rising edge-triggered flip-flop clocked by clk)
 Path End Point : cg2 (TLATNTSCAX2)
 (rising edge-triggered flip-flop clocked by clk)
 Mode : default
 Corner : slow
 Path Type : max
 Path Group : **clock_gating**
```

Pin	Cell	Edge	Arrival	Delay	Slew	Total Cap
ff1/CK	DFFQX1	Rise	0.0	0.0	0.0	
ff1/Q	DFFQX1	Fall	128.0	128.0	52.0	2.68822
-> cg2/E	TLATNTSCAX2	Fall	128.0	0.0	52.0	

	Time	Total
target clock clk (rise edge)	0.0	0.0
target clock cycle shift	300.0	300.0
library setup check	-99.0	201.0
data required time	201.0	
data required time	201.0	
data arrival time	-128.0	
pessimism margin	0.0	
slack	73.0	

Fig. 5. Timing report without estimated clock latency annotations.

With latency estimation annotations (as virtual clock latencies)

```
Nitro-SoC> report_timing -to cg2/E
 Timing Path to cg2/E

 Path Start Point : ff1 (DFFQX1)
 (rising edge-triggered flip-flop clocked by clk)
 Path End Point : cg2 (TLATNTSCAX2)
 (rising edge-triggered flip-flop clocked by clk)
 Mode : default
 Corner : slow
 Path Type : max
 Path Group : **clock_gating**
```

Pin	Cell	Edge	Arrival	Delay	Slew	Total Cap
ff1/CK	DFFQX1	Rise	501.0	501.0	0.0	
ff1/Q	DFFQX1	Fall	629.0	128.0	52.0	2.68822
-> cg2/E	TLATNTSCAX2	Fall	629.0	0.0	52.0	

	Time	Total
target clock clk (rise edge)	0.0	0.0
target clock cycle shift	300.0	300.0
target clock ideal network latency	408.0	708.0
library setup check	-99.0	609.0
data required time	609.0	
data required time	609.0	
data arrival time	-629.0	
pessimism margin	0.0	
slack	-20.0	

Fig. 6. Timing report using estimated clock latency annotations.

IV. Results And Analysis

A. *Validation plan*

This project provides a new method for an existing functionality related to the clock tree latency estimation at the pre-CTS stage.

With most designs, there should be little quality of results (QoRs) change after pre-CTS, though the total negative slack (TNS) will be very different during the pre-CTS optimization steps. A validation should be done by running both the pre-CTS and post-CTS stages before comparing the QoRs.

Specifically, when comparing "slack margin-based" annotations against the new "estimated clock latency annotations," the following validations should be assured:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

- Execute the full pre-CTS stage for the flow testing and verify the usual QoRs criteria at the end.
- The estimated latencies at pre-CTS and real latencies at CTS should be very close.
- Execute both pre-CTS and post-CTS stages and measure the QoRs at the end: it should be similar between both runs as annotations are no longer in the picture once clocks are propagated.
- Pre-CTS runtime mentoring: Annotating using estimated clock latencies is slower than slack margins. We should know by how much.
- Full flow runtime: runtime for both pre-CTS and post-CTS stages should be close.

B. Test cases used for validation

Nine test cases are used to validate the proposed methodology. The objective is to use diverse designs with different complexity and technology nodes. From a design perspective, the timing constraints are derived by high clock frequency and circuits contain more than 500,000 logic cells. From the technology side, a wide range of the most advanced technology nodes is chosen, from 65 nm to 16 nm. The choice of test cases for validation is important for two reasons. First, to validate the new method, we need to apply it to a large variety of designs to build a robust solution. Second, test cases should be real designs used in commercial and recent ICs, which will convince customers to use them. Table I shows their main characteristics.

TABLE I
TEST CASES: MAIN SPECIFICATIONS

TEST CASES	NUMBER OF INSTANCES	MAIN CLOCK FREQUENCY (GHZ)	PHYSICAL AREA (μM^2)	TECHNOLOGY NODE (NM)
TEST #1	539,268	1.45	554,566.21	22
TEST #2	398,308	2.30	559,666.37	65
TEST #3	537,240	1.45	557,008.62	20
TEST #4	189,800	1.82	625,116.24	28
TEST #5	429,200	0.87	4,894,640.96	65
TEST #6	589,509	2.82	494,696.16	16
TEST #7	690,290	0.60	7,168,642.56	65
TEST #8	894,588	1.00	2,876,823.05	16
TEST #9	571,744	0.63	2,190,060.00	28

For the technique validation, the Mentor Graphics Nitro-SoC™ tool [16] and flow [17,18] are used to execute the full PNR flow on all test case experimentation.

C. Results

As the concerned engine is the clock-tree building, the first objective is to have an estimated latency at the pre-CTS stage that is very close to the real latency at the CTS stage. To validate that, we used the below steps:

1. Load the database.
2. Run the latency estimator.
3. Store the latency estimations.
4. Synthesize the clock tree.
5. Query the implemented latencies.
6. Compare the implemented latencies with the estimated ones.

Figure 7 shows how close the estimation is to the implemented latencies for each test case.

Fig. 7. A comparison of estimated and implemented clock latencies.

Note that annotations will increase the arrival times of all ideal clock timing endpoints, and therefore may change the timing picture of a design. This is in comparison to the older mechanism, which used slack margins and was designed to only change the clock-gating paths.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

Consequently, it is expected that the TNS will increase after we insert the annotations due to new violations that appear when estimated latencies are used instead of the ideal latency. Figure 8 shows the TNS comparison at the end of the pre-CTS stage between the old method and the new one. Figure 9 shows the CKG TNS.

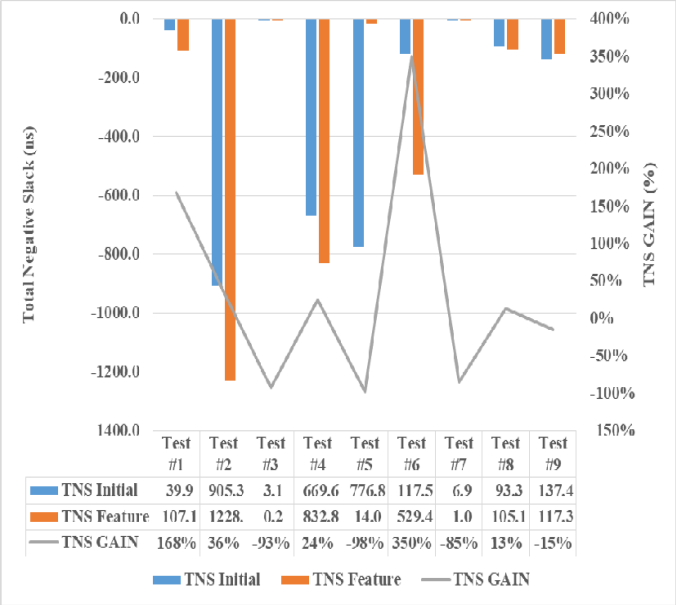

Fig. 8. Impact on timing TNS at the pre-CTS stage.

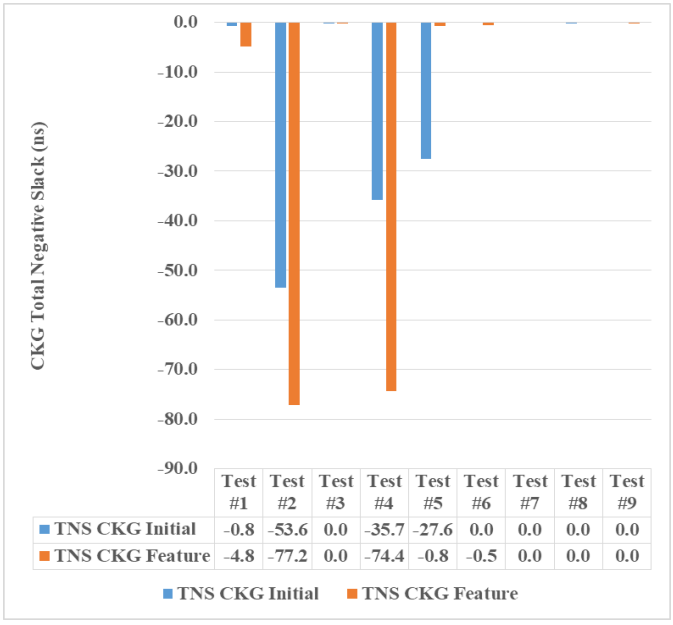

Fig. 9. Impact on timing CKG TNS at the pre-CTS stage.

After the pre-CTS stage, as expected, the gain of this new method is mostly observed at the post-CTS stage. Since the new arrival times come from estimated clock arrival times, the new timing values match more closely with the expected TNS immediately measured after CTS. Figure 10 displays the TNS and in Figure 11 the CKG TNS at the end of the post-CTS stage using both methods. The benefit of the new method is clearly demonstrated as the TNS and CKG TNS are globally reduced in all test-cases by an average of 34% and 65%, respectively.

A reduced TNS at the post-CTS stage leads to faster and better timing closure at the end of the route stage due to fewer violations needing to be fixed. Figures 12 and 13 show how much the timing TNS and CKG TNS are reduced in all test cases when estimated clock latency is used early in the flow from pre-CTS. An improvement of up to 68% of TNS is achieved in case #5 and the average improvement is about 35%. The same goes for the CKG TNS, with a reduction of up to 82% in case #3 and 63% in case #8, and an average reduction of 32%. This proves the benefit of the new method as the global gain is well preserved compared to the post-CTS gain.

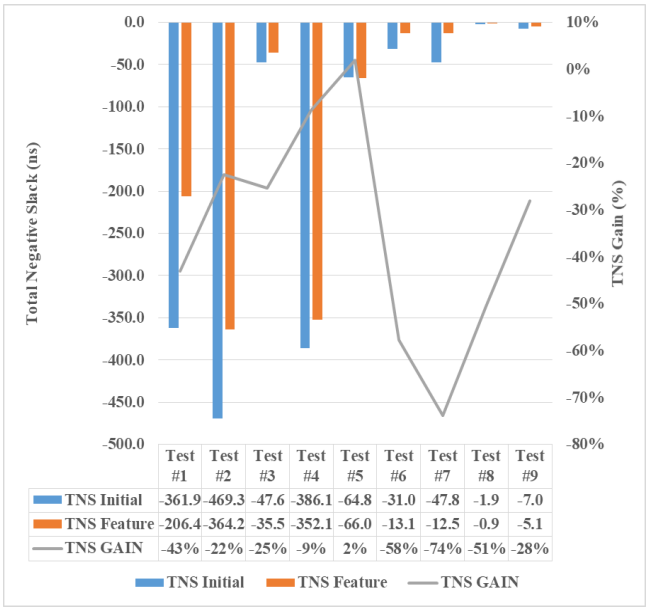

Fig. 10. Impact on timing TNS at the post-CTS stage.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

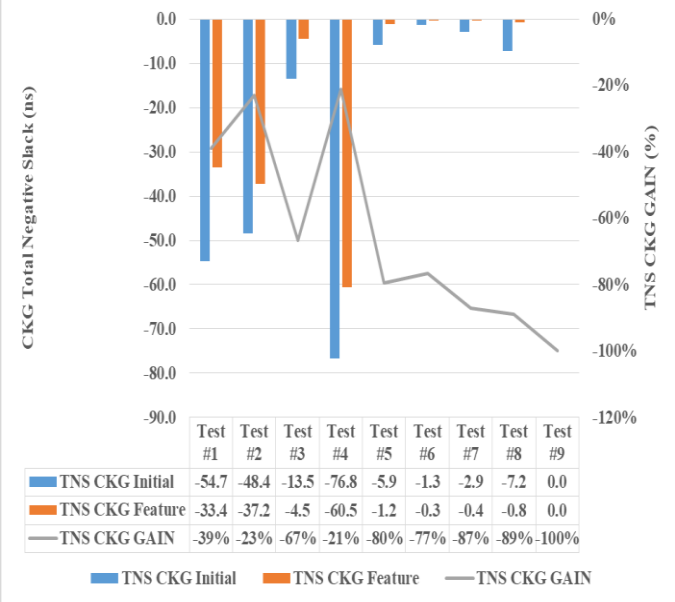

Fig. 11. Impact on timing CKG TNS at the post-CTS stage.

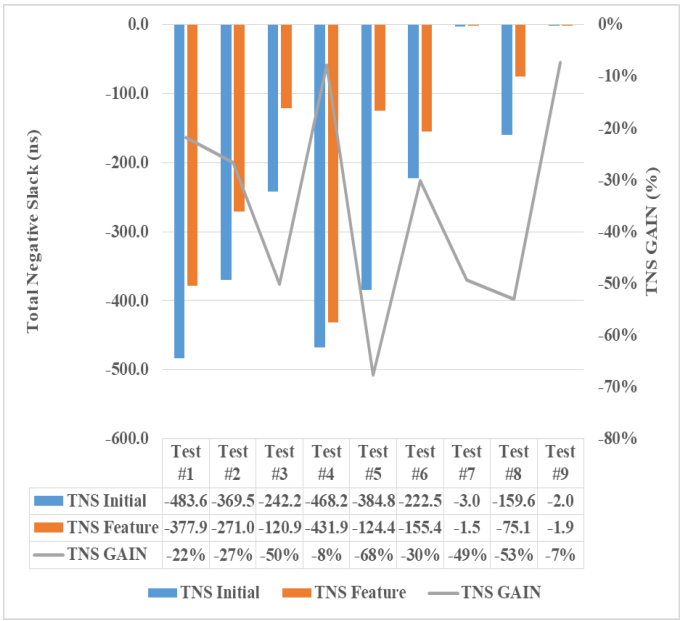

Fig. 12. Impact on timing TNS at the route stage.

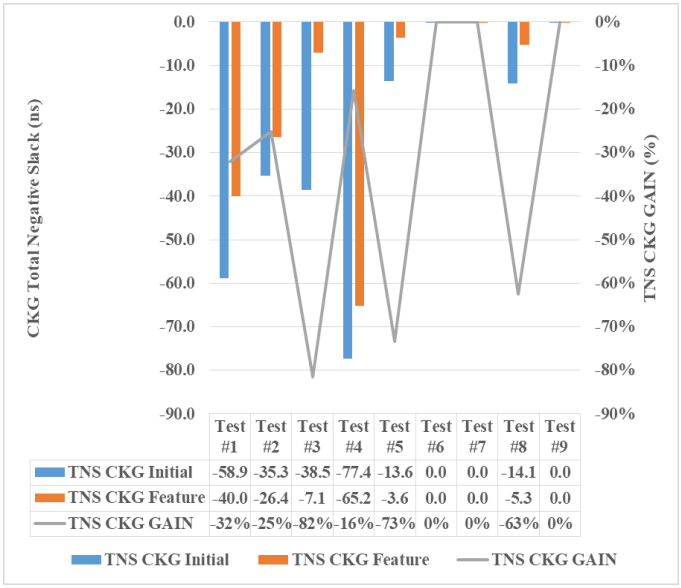

Fig. 13. Impact on timing CKG TNS at the route stage.

Any new method cannot be accepted in a production version of an EDA tool if it impacts the runtime by a lot. It is true that the proposed method adds some runtime during the pre-CTS stage due to additional paths' violations. However, the usual runtime is easily recovered during the next post-CTS and route stages. This is due to fewer remaining timing violated paths seen after building the clock tree. During this exercise, we have achieved a 14% runtime reduction in case #3 and a 3% average run time reduction over the nine test cases. Figure 14 shows the total runtime for each case.

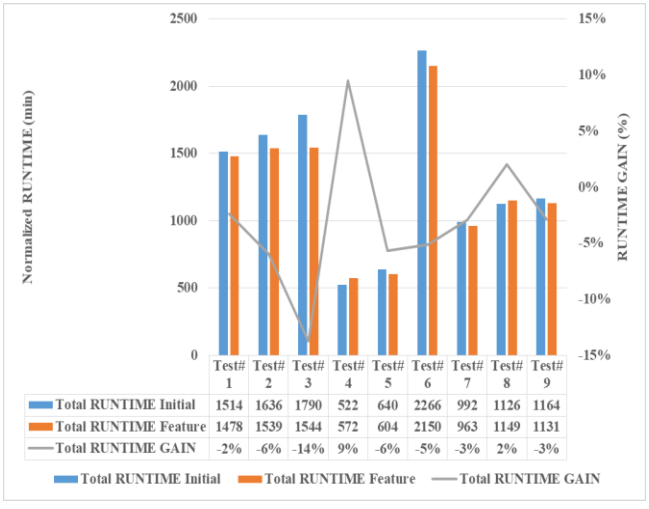

Fig. 14. Impact on full PNR flow total runtime.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 02, February 2022)

V. Conclusion

This work introduced a new method, "estimated clock latencies annotations." The objective is to annotate accurate latencies on each clock pin early in the Place and Route flow during the pre-CTS stage. Accurate clock tree estimation helps us to find timing violations that may appear at the post-CTS stage. The ability to see violations early in the flow is essential for efficient timing closure. The validated technique leads to a TNS reduction of up to 68% with an average of 35% and a CKG TNS improvement of up to 82% with an average of 32%. Another advantage is that there is no runtime degradation. Finally, all techniques having clocking architecture like in ASICs could benefit from the research in this paper by adding an accurate clock latency estimation to their algorithms.

Acknowledgements

This research supported by "Mentor Graphics A Siemens business" Corporation.

REFERENCES

[1] Bhasker J. and Chadha R. Static timing analysis for nanometer designs: a practical approach. New York: Springer, 2011.

[2] Lin T.-L. and Chen S.-J., "A Platform of Resynthesizing a Clock Architecture Into Power-and-Area Effective Clock Trees," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 10, pp. 2475–2488, 2020.

[3] Peng X., Wang H., Wang S., and Du J., "A New Generation of Static Timing Analysis Technology Based on N7 Process—POCV," 2019 IEEE 4th International Conference on Integrated Circuits and Microsystems (ICICM), 2019.

[4] Dietrich M. and Haase J., Process variations and probabilistic integrated circuit design. New York, NY: Springer, 2012.

[5] Shahsavani S. N. and Pedram M., "A Minimum-Skew Clock Tree Synthesis Algorithm for Single Flux Quantum Logic Circuits," IEEE Transactions on Applied Superconductivity, vol. 29, no. 8, pp. 1–13, 2019.

[6] Kahng B., "New game, new goal posts," Proceedings of the 52nd Annual Design Automation Conference on - DAC 15, 2015.

[7] Sengupta D., Mishra V., and Sapatnekar S. S., "Invited - Optimizing device reliability effects at the intersection of physics, circuits, and architecture," Proceedings of the 53rd Annual Design Automation Conference on - DAC 16, 2016.

[8] Thomas L. and Kiran V., "Timing convergence techniques in digital VLSI designs," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017.

[9] Jhao P.-R., Wu D. C.-Y., and Wen C. H.-P., "Skew-Aware Functional Timing Analysis Against Setup Violation for Post-Layout Validation," 2018 IEEE International Test Conference in Asia (ITC-Asia), 2018.

[10] Kao H.-Y., Lee Y., Huang S.-H., Cheng W.-K., and Chou Y.-C., "An industrial design methodology for the synthesis of OCV-aware top-level clock tree," 2017 6th International Symposium on Next Generation Electronics (ISNE), 2017.

[11] Liu W.-H., Li Y.-L., and Chen H.-C., "Minimizing clock latency range in robust clock tree synthesis," 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010.

[12] Farshidi A, Rakai L., and Behjat L., "An efficient optimal clock network buffer sizing with slew consideration," 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 2017.

[13] Lu J. and Taskin B., "Post-CTS Delay Insertion," VLSI Design, vol. 2010, pp. 1–9, 2010.

[14] Mishagli D., Koskin E., and Blokhina E., "Path-Based Statistical Static Timing Analysis for Large Integrated Circuits in a Weak Correlation Approximation," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019.

[15] Nitro-SoC™ Software Version 2019.1, August 2019.

[16] Nitro-SoC™ Software Version 2020.1, April 2020.

[17] Nitro-SoC™ "User's Manual", Software Version 2020.1, August 2020.

[18] Nitro-SoC™ "Advanced Design Flows Guide", Software Version 2020.1, August 2020.