Divergence Engine: Early prediction of clock-tree divergence at Logic-synthesis stage

Sanjana Sundaresh , Murali Mohan Thota , Atul Garg

Texas Instruments India Pvt Ltd ,Bangalore,India

sanjana@ti.com,muralit@ti.com,atulgarg@ti.com

Abstract— **Timing convergence, while reducing area and power, is one of the hardest challenges in the physical design of nanometer VLSI chips which push the limits of frequency entitlement. If there is clock tree divergence (CTD) due to architecture in a high frequency design, then margins are required during optimization stage. These margins will account for divergence at the early stages that can significantly impact the frequency entitlement. However, each timing path can have different divergence values and applying the same margins across all paths is not optimal. To address CTD, applying flat margins can lead to incorrect frequency entitlement, impacting area and power. If accurate path specific margins based on CTD are used, it will help in early optimization stages i.e. at logic - synthesis and placement to achieve the right power, performance, area and schedule (PPAS). Early identification of critical paths which are highly clock tree divergent at logic-synthesis stage and in-time architecture feedback will go a long way to reduce design or project cycle time.**

Keywords— clock tree divergence, early feedback, frequency entitlement, clock skew

I. Introduction

Timing closure of a VLSI design is an iterative process. At each stage of the backend flow, i.e. logic synthesis, placement, clock-tree synthesis and routing, different challenges are exposed to the task of timing closure. At the logic synthesis stage, the basic netlist and connections are available. At placement stage, floorplan related variations will start to take shape. Post clock-tree synthesis (CTS), clock latencies will be applicable. This results in clock skew, CTD impacting timing. Further, post routing, the real net delay will come to play. At each of these stages, timing will degrade.

Especially for a high frequency design with considerable CTD, appropriate selection of design-ware components is of paramount importance. This will largely impact to what extent synthesis tool can perceive the criticality of the design. To achieve this, traditionally flat margins are used across the flow. Calculation of flat margins is shown in Table I. The tool natively supports the application of flat margins as part of clock uncertainly for any given clock. However, this option offers no flexibility and penalizes all paths equally. As we attempt to push frequencies higher, the choice of these flat margins becomes crucial. Lower margins can lead to incorrect frequency entitlement at signoff stage as paths may not be optimized sufficiently. Higher margins can penalize area in certain parts of design, since not all paths are timing critical in a design. In extreme cases, high margins can cause problems in the timing optimization itself and result in failure of timing closure. This current estimation of margins is very unscientific and is derived from previous design knowledge.

For any given clock, timing paths in the design can be roughly divided into 4 categories

1) *Low Divergence + Low path depth*
   a. Optimize for best area and power
2) *Low divergence + High path depth*
   a. Optimize to meet timing without area and power impact
3) *High divergence + Low path depth*
   a. Optimize to meet timing smartly to avoid iteration from synthesis.
4) *High divergence + High path depth*
   a. Identify such critical paths early which are frequency entitlement bottlenecks and provide RTL feedback.

As the nature of each category is different, a single flat margin cannot address all of them. Hence to get the best of each of these categories, applying path specific real margins will give us the best frequency, area and power entitlement.

II. Traditional Margins

This section explains the current methodology of flat margins. In this case, a certain amount of clock latency and CTD is assumed for the design based on previous design experience as shown in Table I. Margins are calculated based on this assumption and applied as part of uncertainty across the optimization stage (till pre-CTS). They are removed post CTS.

TABLE I. Flat Margin Calculation

Predicted Latency	7000ps
Assumed CTD	30%
Lauch_clock derate	1.113
Capture_clock derate	0.947
Skew	250ps

Divergence margins = 7000 * 30 (1.113 – 0.947) + 250 = 602ps

At post CTS stage, if a violating path has balanced clock tree and no useful skew is employed, then the violation is most likely due to CTD. So the divergence that was initially assumed to calculate margins is incorrect or not sufficient for these paths. This had led to under optimization of these paths from logic-synthesis stage. In such cases, by the existing methodology, this is taken as feedback and synthesis is repeated with extra uncertainty for these paths . Then placement and CTS is redone to check if the path is meeting frequency. If the path fails even after applying high margins, architectural feedback is provided to the front end team.

Drawbacks of current methodology can be summarized as:

1) Inability to predict frequency entitlement at logic synthesis stage.

2) Requirement to complete CTS in order to find critical paths and hence delayed architectural feedback.

3) Unscientific iterative margin process impacting PPAS.

For a high frequency design with considerable CTD, these drawbacks impact significantly. A better methodology is required to bridge the gap between actual CTD and margins applied at logic synthesis stage.

III. Divergence Engine

To address CTD margins, an algorithm is developed which predicts and dumps path specific divergence values. This algorithm is called the Divergence Engine (DE). To understand the divergence calculation intricacies, take the case of a typical timing path between launch flop FF1 and capture flop FF2 shown in Fig. 1.As shown in Table II estimated divergence is 44%.

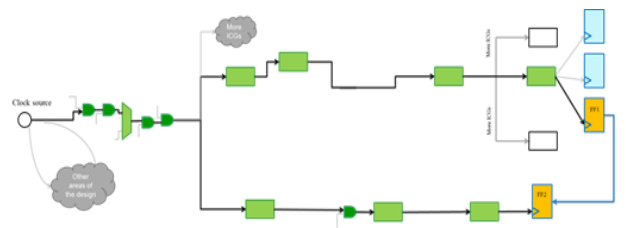

Fig. 1 Typical timing path at pre-CTS stage

TABLE II. Actual calculation of clock divergence pre-CTS stage

Common clock path	5 stages
Uncommon launch clock path	4 stages
Uncommon capture clock path	4 stages

Divergence = Worst Uncommon clock path/Total clock path = 4/9 = 44%

Assume after post CTS expansion shown in Fig. 2

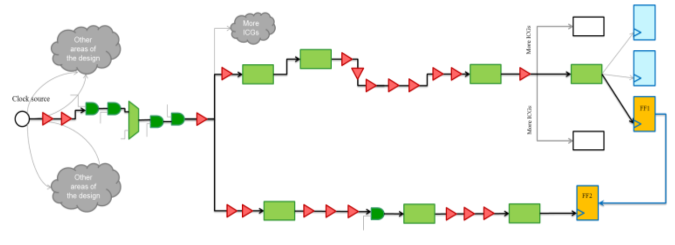

Fig. 2 Timing path at post clock tree expansion

TABLE III. Clock divergence post CTS stage

Common clock path	8 stages
Uncommon launch clock path	13 stages
Uncommon capture clock path	12 stages

Divergence @ post CTS stage = 13/21 = 62%

There is a huge difference between pre-CTS and post CTS divergence percentages. In order to fill this gap, the major challenge is to predict the buffer insertion points done by the CTS tool. Different approaches to achieve this will be discussed below. For illustration purposes, launch path is assumed to have worst uncommon clock path.

A. *Fan-out based weight estimation*

One basic reason for buffer insertion at CTS stage is to meet transition for high fan-out nets. Therefore, if the fan-out of an RTL instantiated clock tree element (CTE) is high then it is reasonable to assume that the CTS tool would insert buffers here. For example, if fan-out of a CTE was 5, then CTS could insert 1 buffer hence a weight of 1 in assigned to this CTE. The total count of CTE would be weight (1) + the element itself = 2. In this manner, more the fan-out more the weight assigned. This is done for every RTL instantiated CTE as shown in Fig. 3

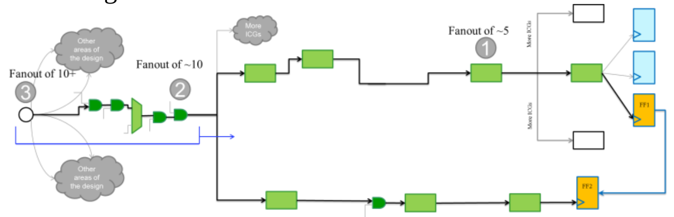

Fig.3 Timing path with fan-out based weight estimation

TABLE IV . Predicted Divergence using this methodology

Common clock path	5 stages + 5 weight
Uncommon launch clock path	4 stages + 1 weight

Divergence = 5/15 = 33%

There is a huge gap in predicted (33%) vs actual divergence (62%) in this methodology. There was a miscorrelation because floorplan based inputs were not considered. Some key findings were :

1) The source of the clock tree example PLL or oscillator is placed further away in the floorplan from its branch and hence tool will always buffer the path to meet a given transition during CTS.

2) This is also the case when the clock path traces through different modules. As a single module tends to be clustered together and could be placed far away from another module, CTS tool will buffer to meet transition.

B. *Mixed Bag Approach*

From previous experiment, it was concluded that adding weights is the right approach but it has to be fine-tuned to add weights appropriately by making it placement aware. Further, as the percentage of clock tree divergence is used to calculate the margins, the exact number of buffers is not required; rather just the correct ratio of worst uncommon clock path and total clock path is needed.

After doing extensive heuristic analysis on a post CTS database, the following key points were identified to enable weight insertion.

- Add weights for every RTL instantiated ICG and MUX.
- Add weights if the clock path changed modules.
- Add weights to clock source.

With this approach predicted divergence is shown in Fig. 4

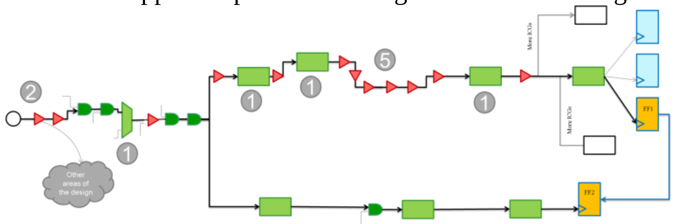

Fig 4. Timing path with weight estimation with mixed bag approach

TABLE V. Predicted divergence using mixed bag approach.

Common clock path	5 stages + 3 weight = 8
Uncommon launch clock path	4 stages + 8 weight = 12

Divergence = 12/20 = 60%

With this an excellent correlation between post CTS divergence data (62%) and predicted data (60%) is obtained.

The exact values of weights to be used for each of the above 3 key points for divergence prediction algorithm is based purely on heuristically analysis done on existing post CTS DB for a platform. In an experiment, weights were configured for a already existing digital design based on its post CTS database. Using these weights, on comparing the post CTS CTD percentage to predicted divergence percentage there was very good correlation with accuracy of +/-10%, if the timing paths were balanced. With the same weights in place DE was run on another design belonging to the same platform and similar extent of high accuracy was achieved. Hence, if weights are configured for DE for a platform design, then they can be reused across different derivatives from the platform. If a design is the first of its kind to a new platform, DE can be run by assuming certain weights to begin with. Paths which are predicted to be divergent at pre-CTS will remain divergent even after post CTS. But the predicted divergence percentage vs actual post CTS based divergence percentage could vary some extent. The weights will have to be fine-tuned based on post CTS feedback as shown in Fig 5.

IV. DIVERGENCE PREDICTION ALGORITHM AND RUN TIME IMPROVEMENTS

The aim of the divergence prediction is to predict, with sufficient accuracy, the actual divergence percentage of each of the timing paths in the design before any clock tree is actually built. The divergence engine (DE) then can be called at various stages to achieve corresponding aims (Fig 5)

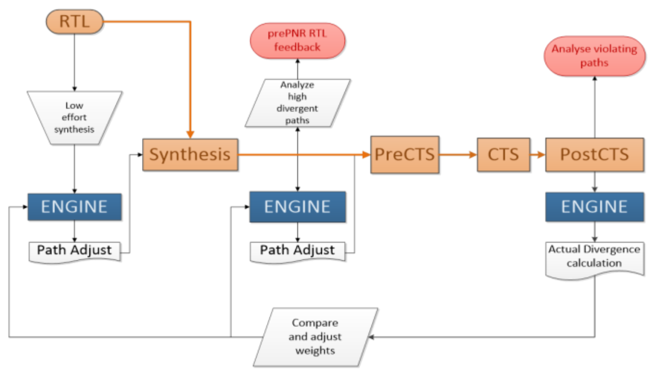

Fig.5 Different stages where DE can be used in the design

DE needs only the netlist and constraints as inputs to calculate divergence. The engine is run in the sign-off tool. Since only instances and their connections are analyzed, accurate delay calculations are not required. Once the divergence values are estimated, the DE will require values of derate, estimated skew and clock insertion latency to calculate the final uncertainty values. Ideally divergence for each and every interacting flop pair needs to be calculated. If there is a design which has 100K flops, the timing path interactions can run into 20Million. Hence calculation of divergence, based on every timing path would be time consuming. To solve this problem, the flop's immediate fan-in level-1 element is used, that is a RTL instantiated CTE. If the divergence is calculated for one flop pair under two interacting level-1 CTE, then all the flops under them will have the same divergence. Logic-synthesis inserted clock gating cells are skipped as they can change with every synthesis run.

DE is composed of 3 major parts. The first step is to find the leaf level RTL instantiated CTE for each sequential element. Divergence will be calculated for each of these CTE pair. The list of CTE (Fig. 6) is written out to a file to be used by the next stage.

Procedure: create_cteList.tcl
Inputs: timing database of the design under process Outputs: 1. CTE – fan-out flop mapping for all registers. 2. CTE – RTL clock tree element mapping.
1. Obtain a list of all the leaf sequential elements in the design. 2. For each of the flops, obtain the fan-in element.   a. If such an element exists   i. Look into the fan-in to this element to obtain the RTL CTE element.   ii. If found, assign the CTE to the RTL CTE.   iii. Append the flop to the CTE fan-out list.   b. Write the fan-out list of CTE to the fan-out file. 3. Close the file handles and return the output files.

Fig. 6 Creating the CTE list

The next stage predicts the total clock path based on the weight assumption given for single timing path for a given CTE pair (Fig. 7).

```
Procedure: predict_clock_path
Inputs: List of pre-CTS elements in path
Outputs: Number of elements expected post CTS expansion
1. For each element i in the path,
    a. Compare the module name with the previous element's module name
        i. If base module names are different, add a weight of 5
        ii. If the second module names are different, add a weight of 1
    b. Check the cell name of the element
        i. If it is an ICG, add a weight of 2
        ii. If it a CTMUX, add a weight of 2
2. Return the total number of elements + weights
```

Fig. 7 Heuristic for estimating path depth

This process is iterated for single CTE w.r.t. all other CTEs. If any valid path is found, the output is written the file. Since each CTE has to be analyzed with all other CTEs, this step is time intensive. To optimize the run time during this step, the algorithm has been designed such that the calculation of divergence for multiple CTE pairs can happen in parallel..

In the last step, the divergence output is parsed through a Perl script to generate uncertainty command files that can be read by optimization pre-CTS tool.

V. Experimental Results

A. Test-Case Details

The test case that considered to check out DE is a digital design which is area and power critical. . The design had been closed using flat margins assuming 50% clock divergence. Using DE , divergence was calculated for every timing path and then mapped to a pie chart details in Fig. 9.

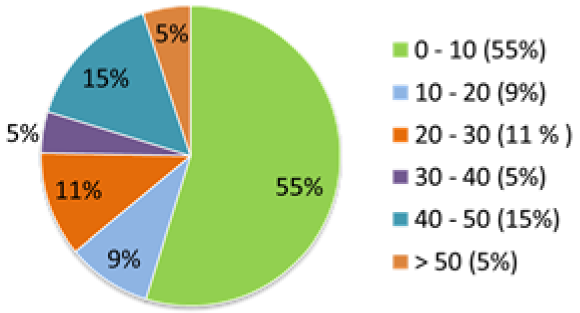

Fig. 9 CTD pie chart of digital design

From the CTD pie chart following are some key take-away.

1) Not many paths with high divergence exist in design.
2) If 50% of the design is assumed to be divergent for margin calculation, it would be valid only for ~5% of the design.
3) The margins were pessimistic for ~95% of paths
4) With the use of margins based on divergence engine, it's expected to see area benefit for design.

B. Test-Run Details

In order to check out the divergence aware margin methodology on Design, 4 runs were given till post CTS setup optimization.

1. Traditional Methodology with flat margins

A flat margin of 50% was assumed.

Clock Latency	= 8000ps
Launch derate	= 1.0632
Capture derate	= 0.945
Skew	= 250ps
Flat margin	= 50%

Divergence margins = 8000 * 50% (1.0632 – 0.945) + 250 = 714ps

Results are presented in Fig. 10

2. Reduced flat margin flow

In this case, 12% of design is assumed to be divergent based on the CTD pie chart.

Divergence margins = 8000 * 12% (1.0632 – 0.945) + 250 = 363ps

Results are presented in Fig. 11

3. Divergence aware methodology (DAM)

This run is with the CTD based divergence aware methodology. Details are captured in section C (Divergence aware methodology) .

Results are presented in Fig.12

4. No margin flow

Just to make sure design was not over penalized w.r.t margins and to see how much area gain can be done, design was run with no margins. Results are presented in Fig.13

C. Divergence aware methodology (DAM)

In this section explains DAM calculations.

1) DE is run on synthesized netlist in signoff timing tool to calculate "Divergence Percentage" for all possible paths for a given clock w.r.t. all RTL inserted CTE. This helps to study the overall divergence percentage of design. And pick a flat divergence % number which represents most of the paths and add that as flat uncertainty. In this case it was 12%. Some portion of flat margins is included as the optimization tool is unable to take all the path based uncertainty given.
2) To calculate this flat margin, below formula is used and added that as clock uncertainty in addition to clock jitter.

Divergence margins = 8000 * 12 (1.063 – 0.945) + 250 = 363ps

3) For paths which have high divergence (> 12 %), extra uncertainty is added in addition to flat margins.

For example,

a) Assume paths between (Launch CT element) ICG1 and (Capture CT element) ICG2 have a predicted divergence of 40%.

b) Overall flat divergence is assumed to be 12%. Hence clock to clock uncertainty applied for say CLK1 will be 363ps

c) Rest 28% of divergence is specific to paths between ICG1 and ICG2. This is applied as uncertainty at logic-synthesis stage.

d) Flops driven by ICG1 and ICG2 are grouped. Uncertainty is applied to only specific interacting paths .

	Traditional FLAT margins only (714ps)			
	Logic Area in mm2	WNS/TNS.FEP	Leakage power	Run time
preCTS	3.16	-625ps/-647ns/2919	51.73mW	13H
postCTS setup opt	3.092	-188ps/-16ns/384	32.561mW	6H

Fig 10 Results of Traditional flat margin flow

	Reduced FLAT margins only (363ps)			
	Logic Area in mm2	WNS/TNS.FEP	Leakage power	Run time
preCTS	2.99	-271ps/-156ns/1517	41.73mW	13H
postCTS setup opt	2.99	-307ps/-173ns/1535	36.29mW	6H

Fig 11 Results of Reduced flat margins flow

	363ps FLAT + Divergence aware margins			
	Logic Area in mm2	WNS/TNS.FEP	Leakage power	Run time
preCTS	3	-184ps/-112ns/1416	49.1mW	24H
postCTS setup opt	2.99	-188ps/-20ns/547	32.8mW	6H

Fig 12 Results of Divergence aware margin flow

	NO MARGIN FLOW			
	Logic Area in mm2	WNS/TNS.FEP	Leakage power	Run time
preCTS	2.82	-180ps/-27ns/567	36.7mW	13H
postCTS setup opt	2.92	-289ps/-330ns/3006	43mW	6H

Fig 13 Results of No margin flow

D. Analysis of the experiment on Design.

1) Traditional Margin flow.
   a) Good timing and leakage picture but area is high.
2) No margin flow.
   a) ~5.8% area saved w.r.t traditional margin flow.
   b) Huge impact on timing with unrecoverable 330ns (20X) as total negative slack (TNS).
   c) 33% impact on leakage w.r.t traditional margin flow
3) Reduced flat margin flow.
   a) ~3.5% of std cell logic area saving w.r.t traditional margin.
   b) 8X degradation in TNS post CTS which is unrecoverable.
   c) 10% increase in Leakage.
4) Divergence aware methodology
   a) 3.5% std cell logic area saving
   b) Predictable timing picture at pre-CTS stage when compared to traditional flow.
   c) Good leakage.
   d) High run time.

Hold optimizations do not affect results as CTS is not influenced but just data path optimization

VI. CONCLUSION

We have developed a custom-built divergence prediction engine that generates path-specific margins based on clock tree divergence with the following advantages.

1) Complete and accurate prediction of expanded clock tree divergence at logic synthesis level without any physical information like floorplan.
2) Very early architectural feedback to the RTL team on critical paths based on clock tree divergence.
3) With realistic margins, we can predict target frequency at pre-CTS.
4) Engine once configured (weights configured) for a platform works with same consistency across different designs.
5) On a post CTS database, the engine can derive weights and hence can calculate the real margins which can be fed back to improve optimization.
6) Divergence Engine can be run on any netlist with a quick turnaround time.

This algorithm can also be extended to dynamically model the local clock skews, useful skew scenarios.