Dynamic Optimization of Skew Balancing through an Innovative Correct-by-Construct Path Delay Query Technique

Tejas Salunkhe
Subhadeep Aich
Abhranil Bose
Ayushi Bapna

Texas Instruments

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Tejas Salunkhe

Tejas has 8+ years of experience in VLSI industry with specialty in physical design and PPAS optimisation domains. He is working as MGTS and synthesis, STA expert at Texas Instruments India for processors class of designs.

Motivation

- In an era of unprecedented technological advancement, the demand for higher performance in System-on-Chip (SoC) designs has become critical across industries like artificial intelligence, 5G communications, and autonomous systems. These domains require SoCs to deliver exceptional computational capabilities, faster data processing, and enhanced efficiency to meet the growing complexity of applications. However, achieving high performance in SoCs introduces multifaceted challenges, including signal integrity, thermal management, and electromigration, which demand innovative approaches to design and optimization.
- **A key area of focus is the management of data path and bus skew specifications**, which have become increasingly stringent due to the rising number of signals with deeper logic levels. Traditional iterative approaches, relying on design, place-and-route (PnR), and static timing analysis (STA) cycles, often struggle to address these requirements efficiently, especially in Mixed Signal SoCs. For example, balancing skew across >200 timing paths with >10 logic depths in a Power Management Unit (PMU) interface poses significant challenges, emphasizing the need for more precise and reliable methods.
- **Prior approaches** and limitations in skew balancing of large number of signals with significant logic depth:
  - **Using set_min_delay, set_data_check constraints**: A fixed reference point such as fixed target delay numbers using min and max delay constraints and a fixed reference signal in set_data_check command, are used across all PD stages to optimize the paths
  - **Custom placement** such as placing flops near analog interface boundary: This method is not suitable for data path with larger logic depths and it also imposes restrictions on place and route. This doesn't guarantee optimal PPA and skew balancing
  - **Manhattan length based custom routing and buffering**: This method is not suitable for notches in floorplan and is also sensitive to floorplan & hard IP pin placements. This method is not a correct by construct through the flow approach thus limiting portability
  - **Building clock tree on data lines**: This approach increases constraints complexity due to large number of clock definitions and imposes overhead of false clock gating checks cleanup. Also adds up extra area & power due to clock tree cell usage on data paths
- To address these challenges, **a correct-by-construct approach that dynamically monitors path delay changes and adjusts constraints offers a pathway to predictable design closure and reduced design cycle times**. This approach not only mitigates bottlenecks but also ensures robust functionality and accelerates time-to-market, aligning with the increasing expectations for speed and performance in modern electronic devices.

Main Idea (1/3)

- Prior approach (Fig 1) uses fixed reference signal or fixed target delay value for skew optimization based on user discretion resulting in suboptimal PPA
- The proposed approach (Fig 2) enables a dynamic selection process for the optimal reference signal for balancing skew around it at every backend optimization step

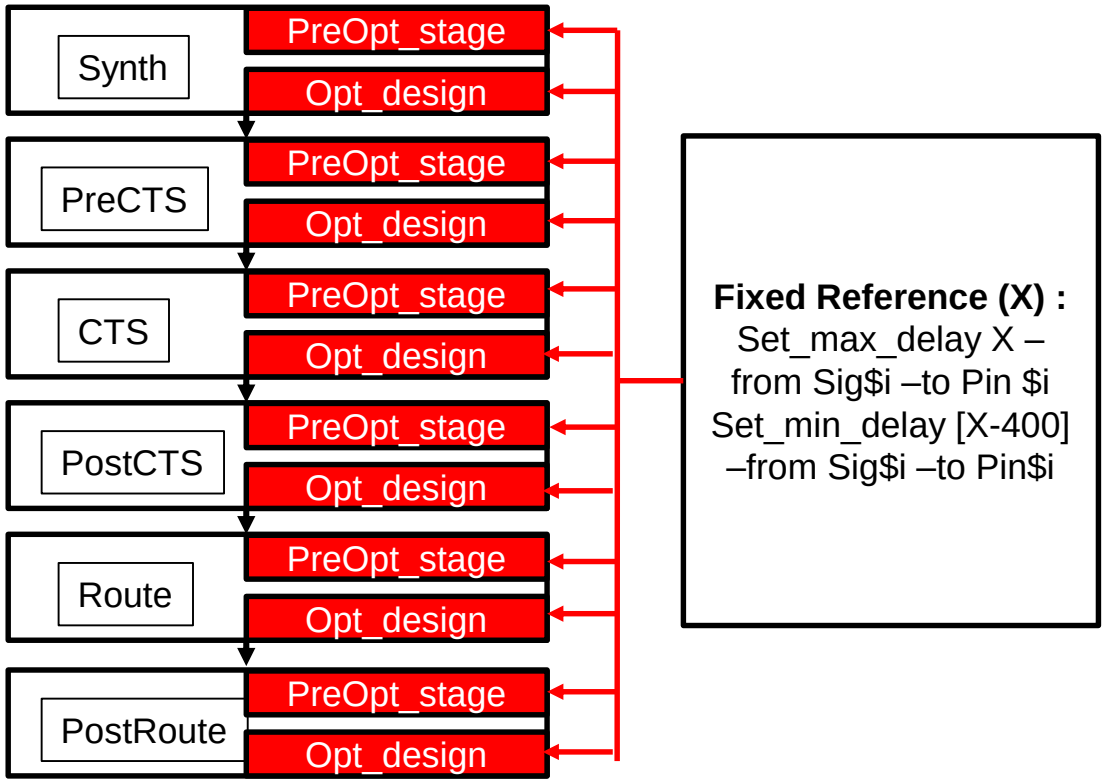

Fig.1 : Prior Approach

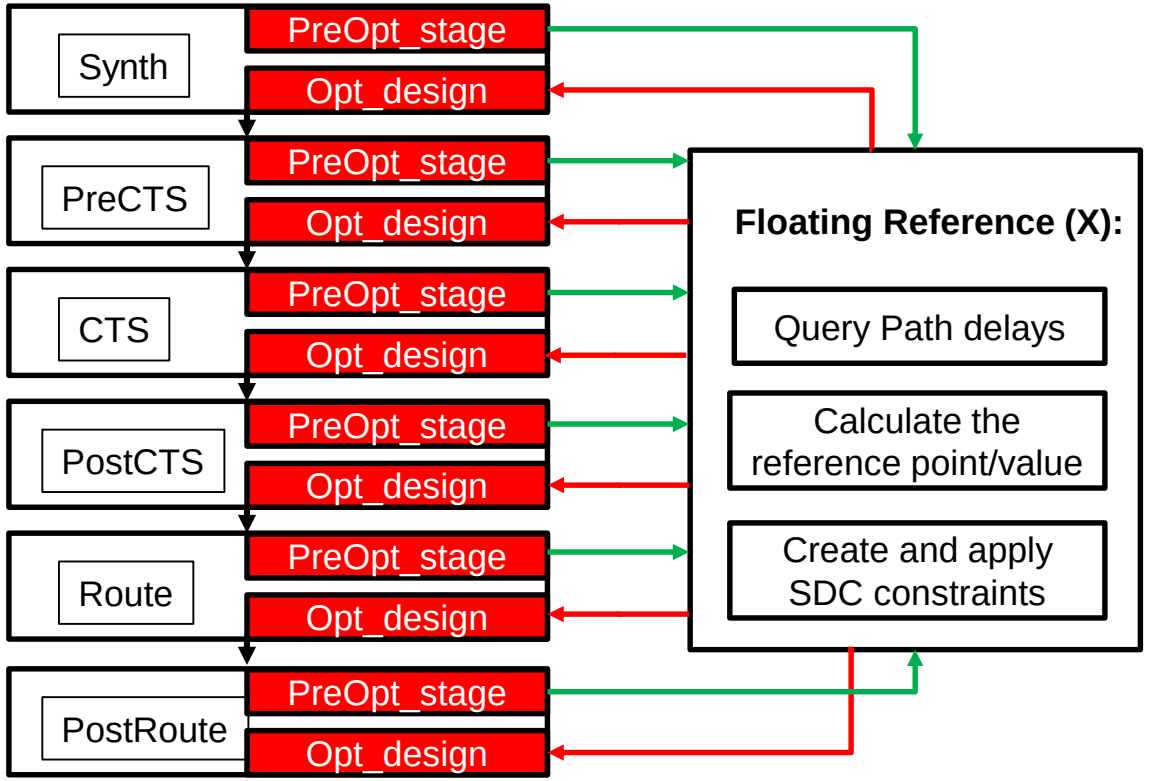

Fig.2 : Proposed Solution

Main Idea (2/3)

- The algorithm shown in Fig. 3 is used to compute the floating reference point/value to be used for skew balancing before each optimization step in backend execution
- Constraints generated using computed floating reference point in SDC format and are fed to subsequent optimization step
- Reference selected based on user inputs such as mean, max, min path delays, % of paths to decide on min, max (default 70%)

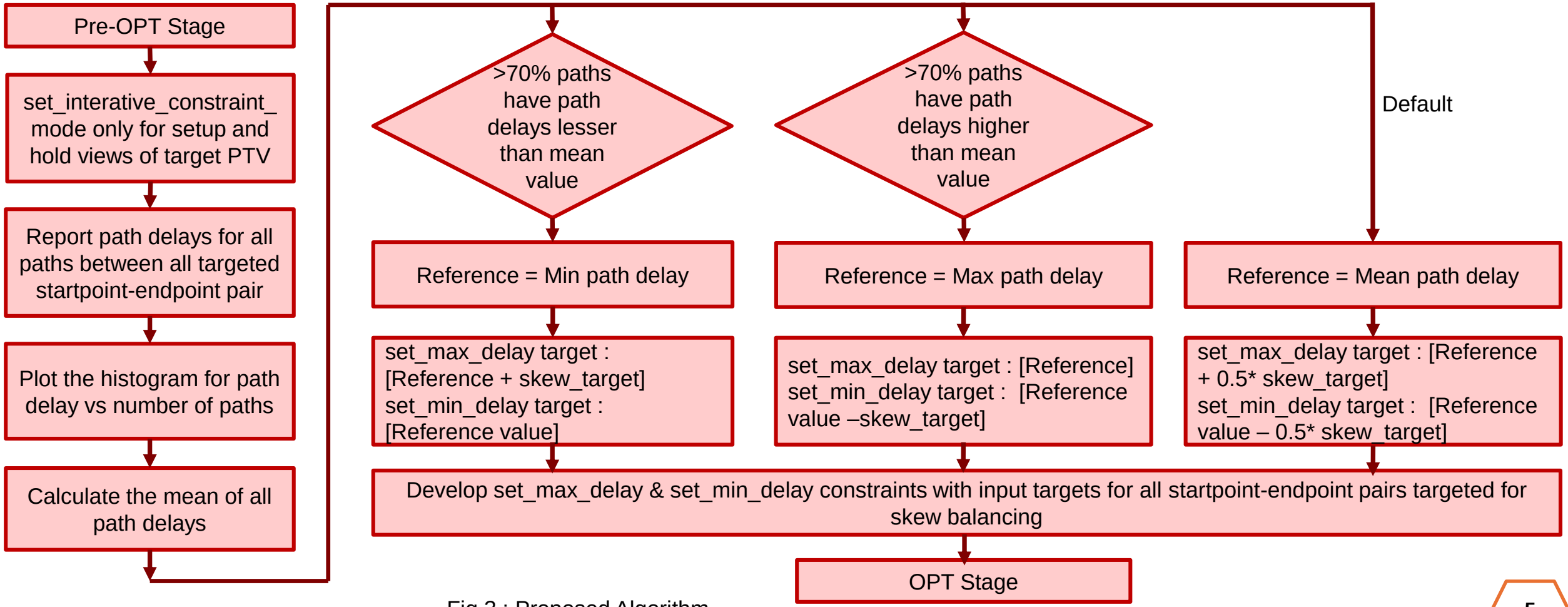

Fig.3 : Proposed Algorithm

Main Idea (3/3)

- To make this solution generic, user friendly and easy to use through EDA tool feature enhancement, it is wrapped into below construct with an intention of industry wide adoption.
- ***set_skew_check -min/max/mean -skew_required_value <skew_value> -skew_window_adj <skew window adjustment> -combinational_from_to -view <view_name> -paths { -from $sp1 -to $eps1 } {-from_rise $sp2 -to_rise $eps2} {-from_fall $sp3 -to_fall $eps3} {-from_rise $sp4 -to_fall $eps4} {-from_fall $sp5 -to_rise $eps5} {-from $sp6} {-to $sp7} {-from_rise $sp6} {-to_rise $sp7} {-from_fall $sp6} {-to_fall $sp7} –help –tag <tag_name>***
    - -min/max/mean: This provide flexibility to chose between 3 algorithms used to calculate floating reference point.
        - Min/Max/Mean option will decide the value to be considered as reference out of given paths delays.
        - Min will consider minimum path delay as reference. Max will maximum path delay as reference. While mean will consider average of all path delays as reference.
        - Default is mean.
    - -skew_required_value : Specifies the skew balancing requirement value/skew target.
    - -skew_window_adj : Provide flexibility to adjust the skew window around reference point
        - The command will generate SDC constraints given below to fed to optimization step.
            - set_max_delay [reference value + skew_window_adj*skew_required_value] -from <Sig1> -to <Pin1>
            - set_min_delay [reference_value – (1-skew_window_adj)*skew_required_value] -from <Sig1> -to <Pin1>
        - The default values for skew_window_adj are based on –min/max/mean option as given below
            - -min : 0.66 , -max : 0.33, -mean: 0.5
    - -combinational_from_to : Ignore clock latency and consider path delay only between provided points
    - -view: Help in selecting analysis view in which optimization is targeted.
        - Traditional fixed reference-based approach is iterative and time consuming to scale across PTV. This proposed solution overcomes this challenge by providing control of PTV through –view option.
    - -paths: Paths between which skew need to be balanced can be selected using multiple options and variation of –from* and –to* as supported by report_timing
    - -tag : tag_name is used in output reports and constraint file name.

Results/Evidence (1/2)

- Table 1 shows how the target values for set_min_delay and set_max_delay constraints changes in fixed reference vs floating reference approach.
- As evident from below table, tracking the path delay, calculating floating reference and adjusting set_min_delay and set_max_delay targets for each optimization step would help in avoiding unnecessary optimization on interface paths.
- For large number of signal set_data_check approach will also pose similar challenges as set_min/max_delay based approach due to fixed reference point usage.
- Set_data_check require signal to be clocked by synchronous clocks if skew need to be balanced. Proposed set_skew_check approach overcomes this challenge by considering path delays.

S.No	PD Flow Stage	Design Data (Path Delay in ps)				Fixed Reference (X)		Floating Reference (X)								
								Mean			Max			Min		
		Td1	Td2	Td3	Td4	set_max_delay	set_min_delay	Reference	set_min_delay	set_max_delay	Reference	set_min_delay	set_max_delay	Reference	set_min_delay	set_max_delay
1	Synth	2000	2200	3000	3500	3500	3100	2675	2475	2875	3500	3100	3500	2000	2000	2400
2	PreCTS	2200	2500	3500	3700	3500	3100	2975	2775	3175	3700	3300	3700	2200	2200	2600
3	CTS	3000	3200	4500	4000	3500	3100	3675	3475	3875	4500	4100	4500	3000	3000	3400
4	PostCTS	3000	3800	4550	4100	3500	3100	3862.5	3662.5	4062.5	4550	4150	4550	3000	3000	3400
5	Route	3100	4200	4600	4200	3500	3100	4025	3825	4225	4600	4200	4600	3100	3100	3500
6	PostRoute	3600	4800	4600	4200	3500	3100	4300	4100	4500	4800	4400	4800	3600	3600	4000

Table 1 : Prior art vs proposed solution. Reference comparison

Results/Evidence (2/2)

- As can be seen in figures 4,5,6 & 7, the usage of buffers and the number of buffers used have been optimized with proposed solution.
- First pass STA skew timing closure enabling **15% signoff cycle time reduction** was achieved.
- **53% area reduction** enabled in Digital-Analog interface channel targeted for skew balancing using proposed approach.

S.No	Category	Prior Approach (set_min/max_delay with fixed reference)	Proposed Solution (set_skew_check with floating reference)
1	Analog-Digital interface channel area (Normalized)	100	47 (**53% less**)
2	Design closure complexity	Manual ECO fixes required as skew requirements not met across PTVs through flow	No manual ECO fixes required due to 1st pass STA success
3	Signoff closure time reduction	5 ECO iterations required	**0 ECO iterations required. 15% signoff closure time reduction**
4	Scalability across PTVs	Not scalable	Scalable across PTVs through –view option
5	Scalability & Reusability across netlists and designs	Not Scalable	Scalable and reusable without any tweaks
5	Core timing QoR distortion risk during signoff	Poses higher risk	No risk
6	Susceptible to late design/architecture/spec changes	Yes. Spec relaxation was required	Could meet spec in first pass
7	Suitable for tighter skew requirement	No	Yes
8	Suitable for skew balancing across large number of signals	No	Easily scalable

Table 2: : Prior vs Proposed Approach

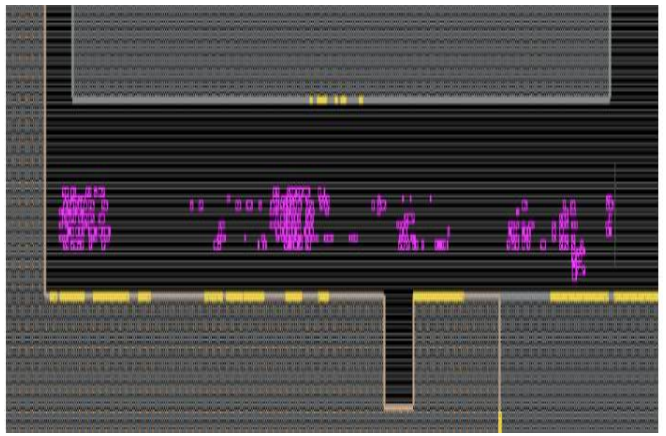

Fig 4: Prior Approach: Cells used for skew balancing

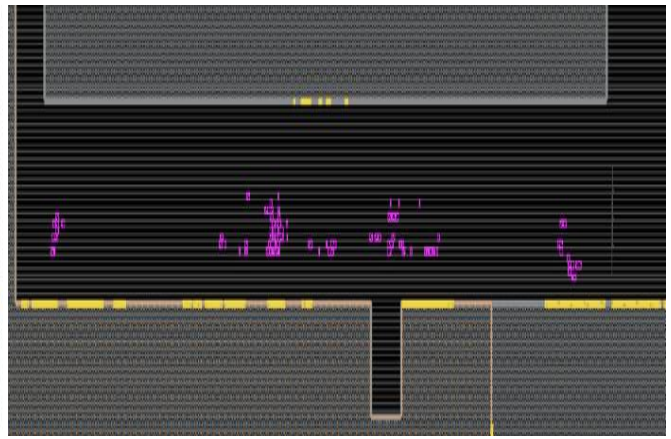

Fig 5 : Proposed Solution: Cells used for skew balancing

Fig 6: Prior Approach: Buffers used on the path

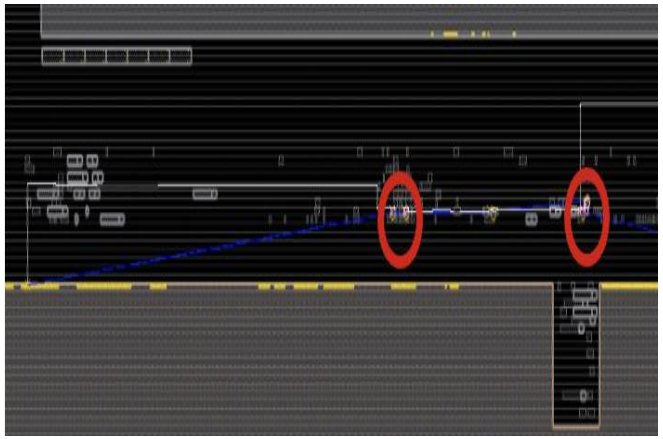

Fig 7: Proposed Solution: Buffers used on the path

Summary

- There is no existing prior approach which provides optimization constraints for tool which dedicatedly target skew balancing. All earlier prior approaches try to constrain path delays with fixed reference value or with respect to fixed signal.
- Skew balancing is about making sure all interested signals arrive in targeted skew window and it doesn't require signals to arrive at specific time.
- Prior approaches lack in providing predictable performance and in providing through the flow correct by construct solution enabling optimal PPA.
- The proposed approach reevaluates the reference point before each optimization, accounting for path changes caused by different backend stages. It generates constraints for optimization, enabling dynamic skew optimization throughout the PD flow. Additionally, the proposed SDC-like construct facilitates industry-wide adoption of this solution.
- Proposed approach enabled **53% of area reduction** in the interface and **optimized 15% of the STA closure cycle time**.

S.No.	Approach	Advantages	Disadvantages
1	Prior Art: Building clock tree on data lines	1) Floorplan and placement independent	1) Constraints complexity increases due to large number of clocks and their relationships. 2) Higher area due to clock tree cells usage. 3) Overhead of false clock gating checks cleanup
2	Prior Art: Defining min/max delay	1) Easy to implement	1) Sensitive to changes in floorplan and placement 2) Sensitive to design changes 3) Sensitive to PD optimization stage 4) More of a trial and error-based approach for tight skew balancing
3	Prior Art: Defining set_data_check	1) Easy to implement 2) Suitable for skew balancing between small number of signals with known fixed reference signal such as clock	1) Doesn't guarantee optimal PPA 2) Fixed reference signal usage can lead to congestion and challenges in final signoff closure
4	Prior Art: Placing flops near analog boundary	1) Predictable approach 2) Suitable for interface which doesn't have much logic between flop and analog hard IP pins	1) Not suitable for implementation where significant digital logic is implemented between flop in digital domain and analog hard IP pins. 2) Imposes floorplan restrictions
5	Prior Art: Pre-route and buffering based on Manhattan length	1) Predictable implementation	1) Not suitable for notches in floorplan 2) Not a correct by construct through the flow implementation 3) Sensitive to floorplan, hard IP pin placement changes
6	**Proposed Solution**	**1) Immune to floorplan , design changes and PD optimization stages** **2) Through the flow optimization technique** **3) Reusable across designs and industry**	1) Custom scripting need to be added in the flow to decode set_skew_check construct.

Table 3: Prior vs Proposed Approaches Comparison Summary