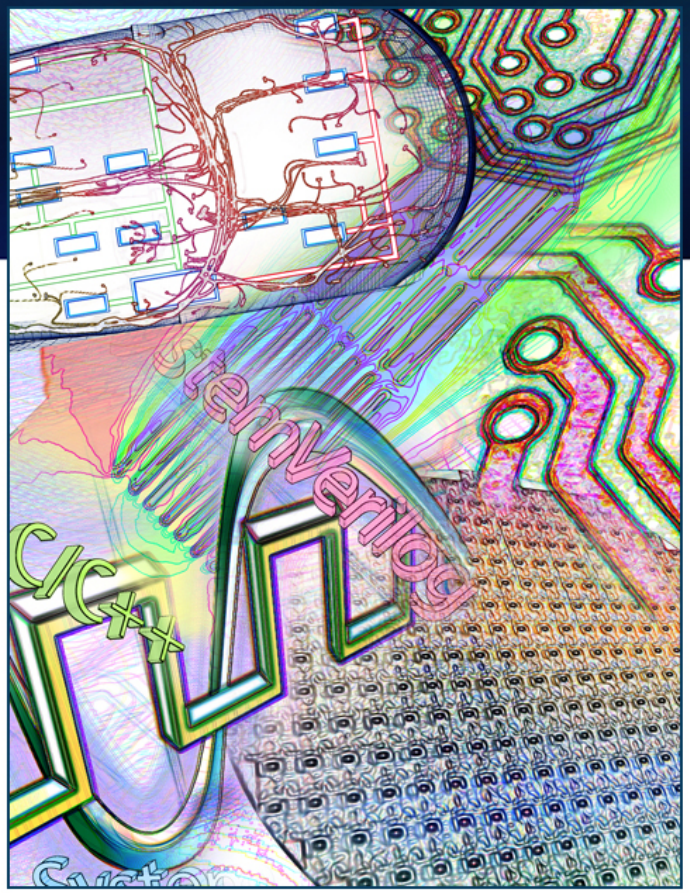


Functional Skew-Aware Clock Tree Synthesis

Venky Ramachandran

P&R Architect

Place and Route Divsion

Mentor Graphics®

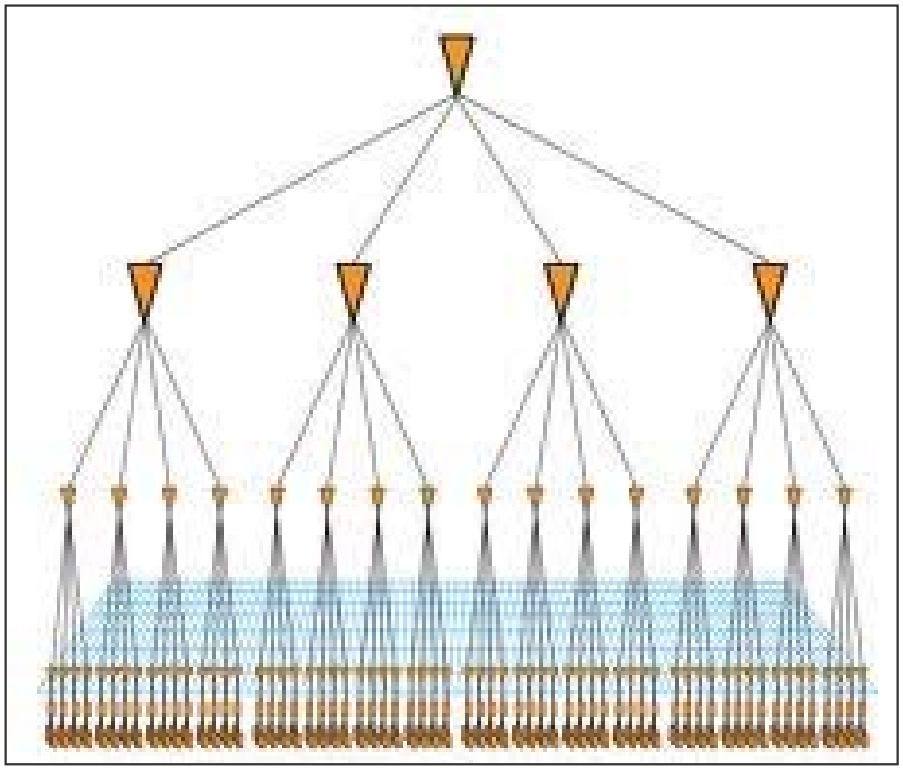
Outline

- CTS Problem Statement & Challenges
- Functional Skew Driven CTS Methodology
- Results & Conclusion

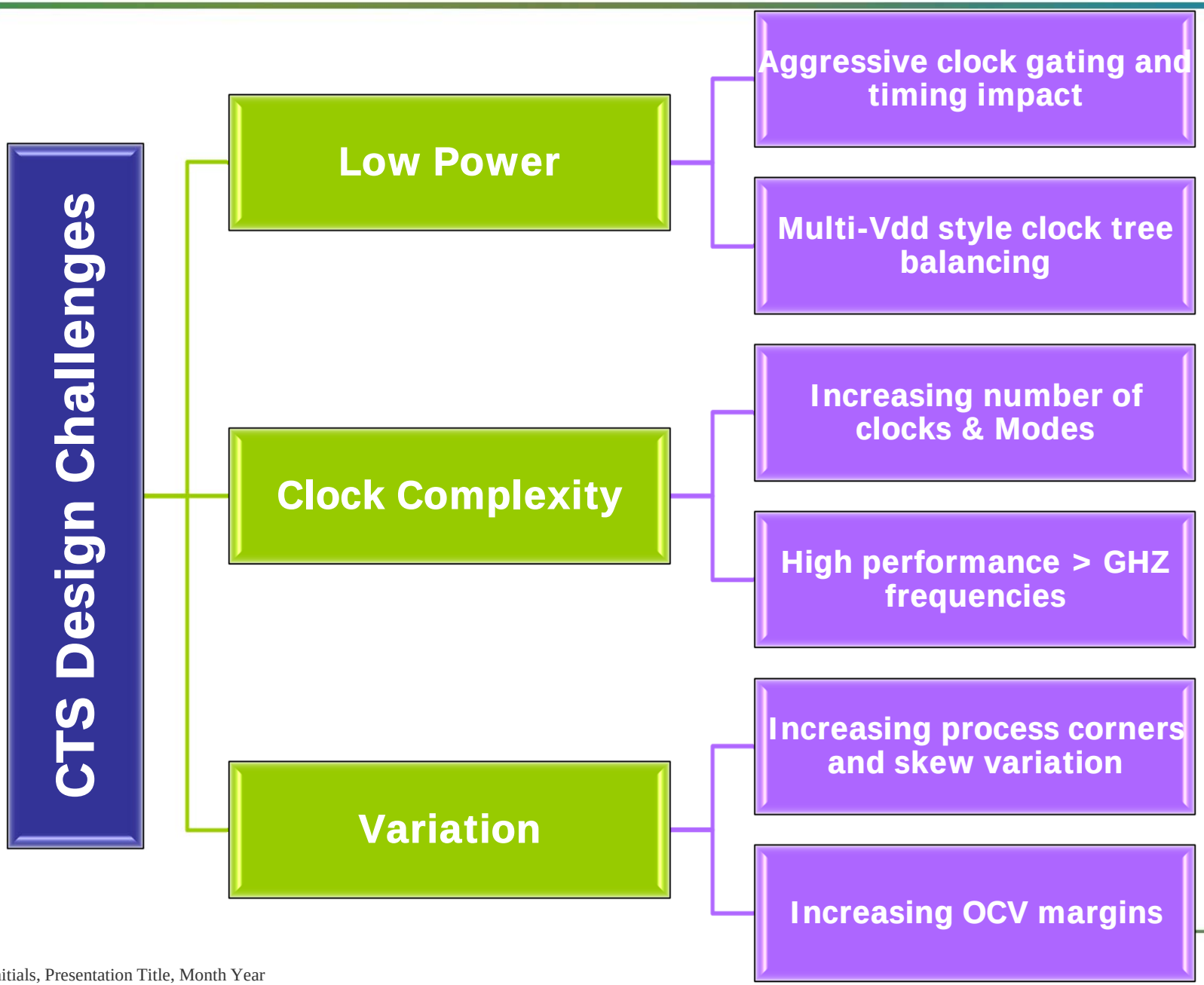
CTS - Problem Statement

- Building a clock tree network with a prescribed set of buffers and inverters,
- Synchronizing every sequential element in the design
- Achieving Smallest buffer and routing resources & best performance (skew, insertion delay)

- TYPICAL ABSTRACTION: Single net buffering problem

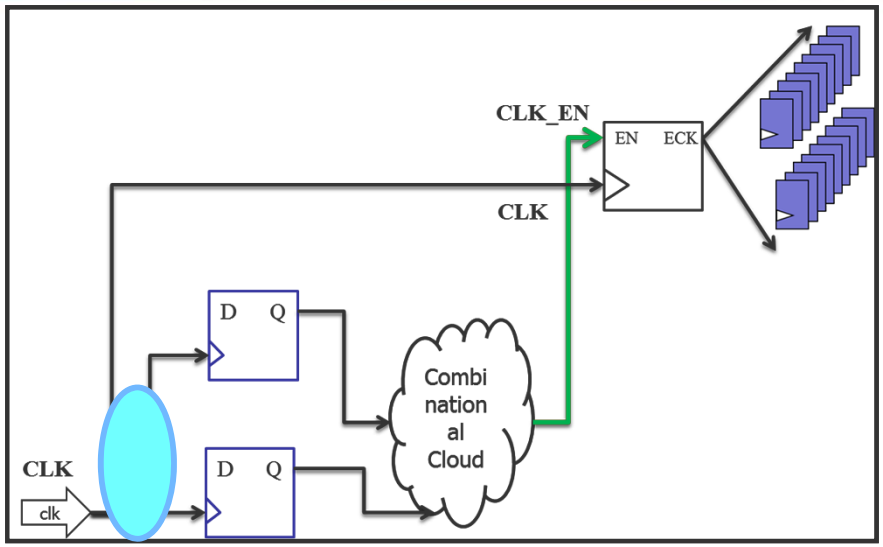


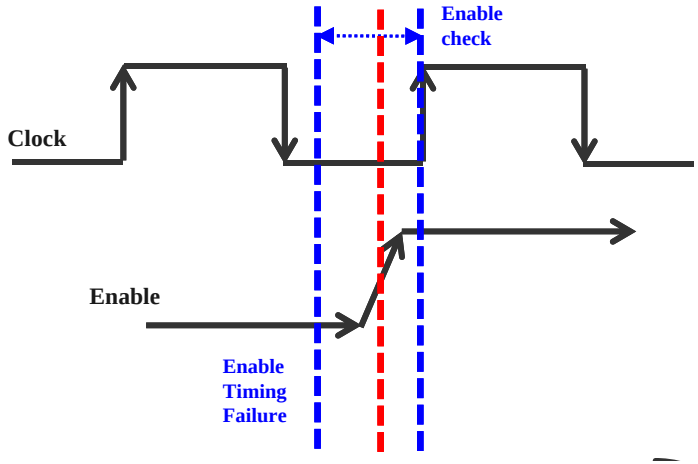
CTS Challenges



Mentor Graphics

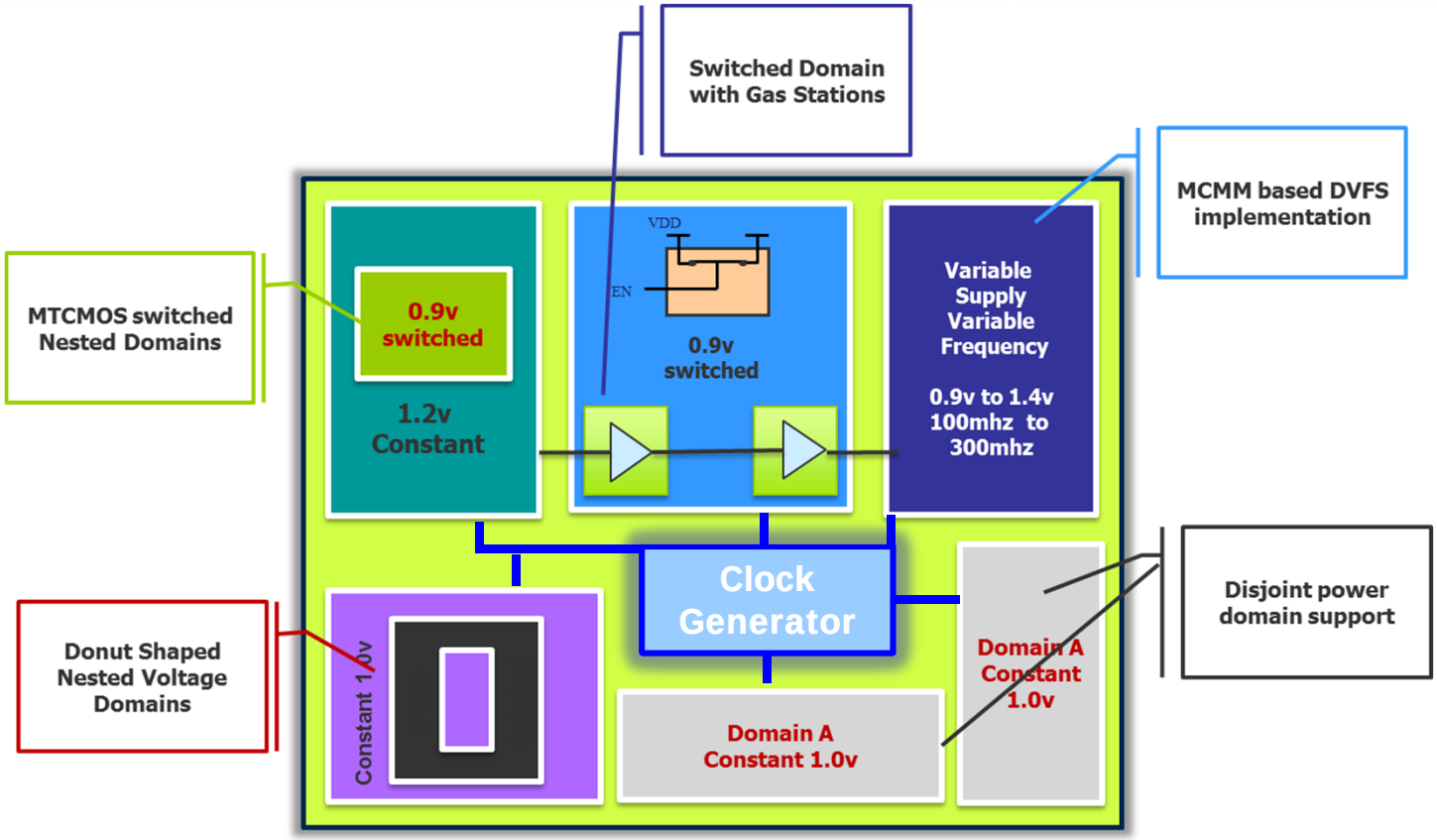
Controlling Clock Power through Gating





- Aggressive and custom clock gating schemes required
- Too many gates leads to lots of small and/or unbalanced buffer trees
- Meeting Enable timing is a challenge
- Impact on OCV as branch point is moved up

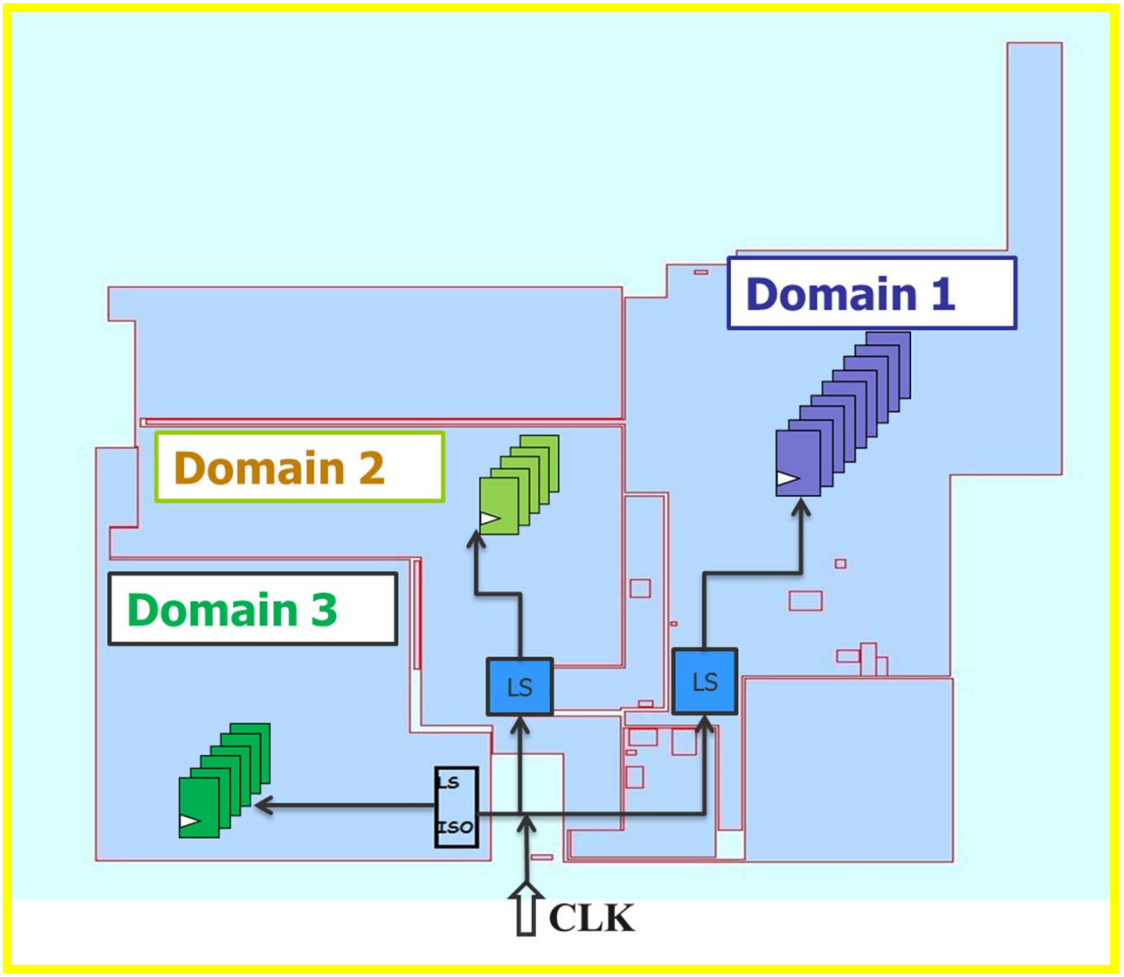
Impact of Multi Voltage Design Styles



- Balancing complexity due to multiple power domains
- Level Shifters and Isolation cells add to latency and complexity
- Multiple libraries characterized for different voltage levels needed

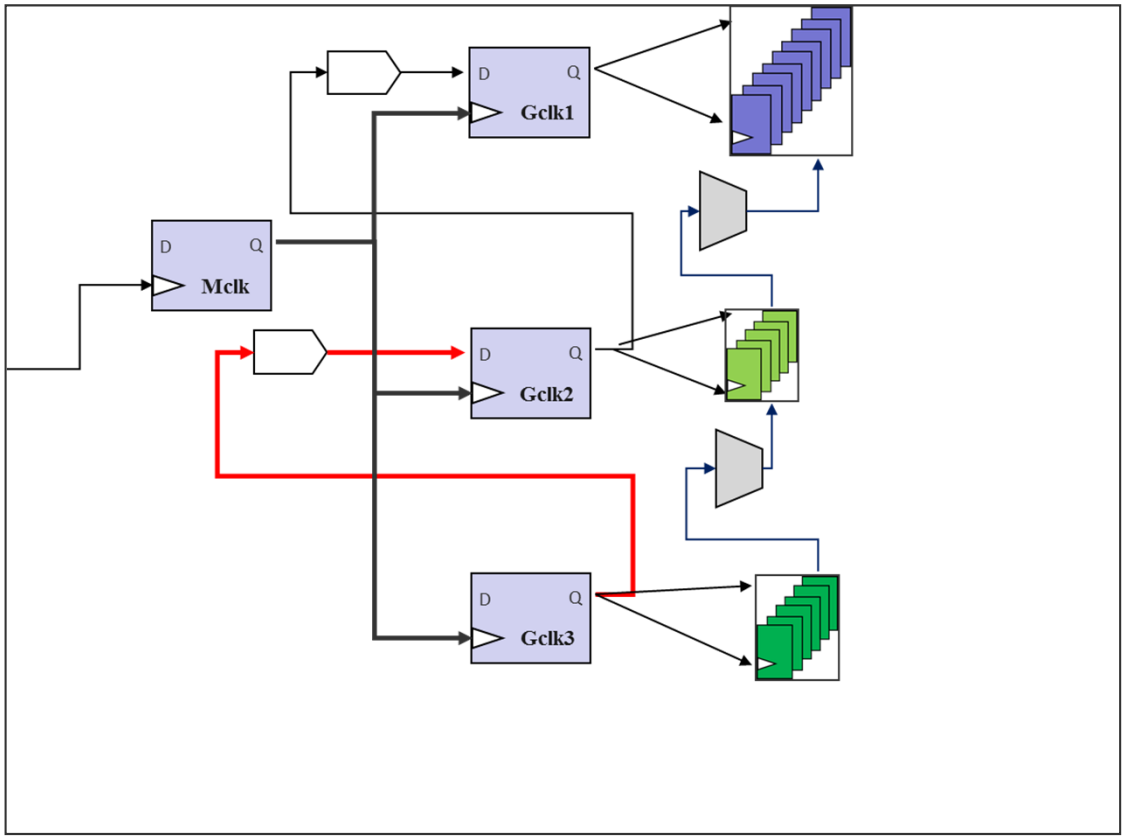
Impact of Power Domains on Balancing

- MV domain complexity lead to non-uniform floorplans
- Balancing across non-uniform domains is a challenge



Increasing Modes & Clock Complexity

- Multiple modes driven by architecture choices
- Large number of clocks & generated clocks
- Clock balancing with multiple modes becomes a challenge

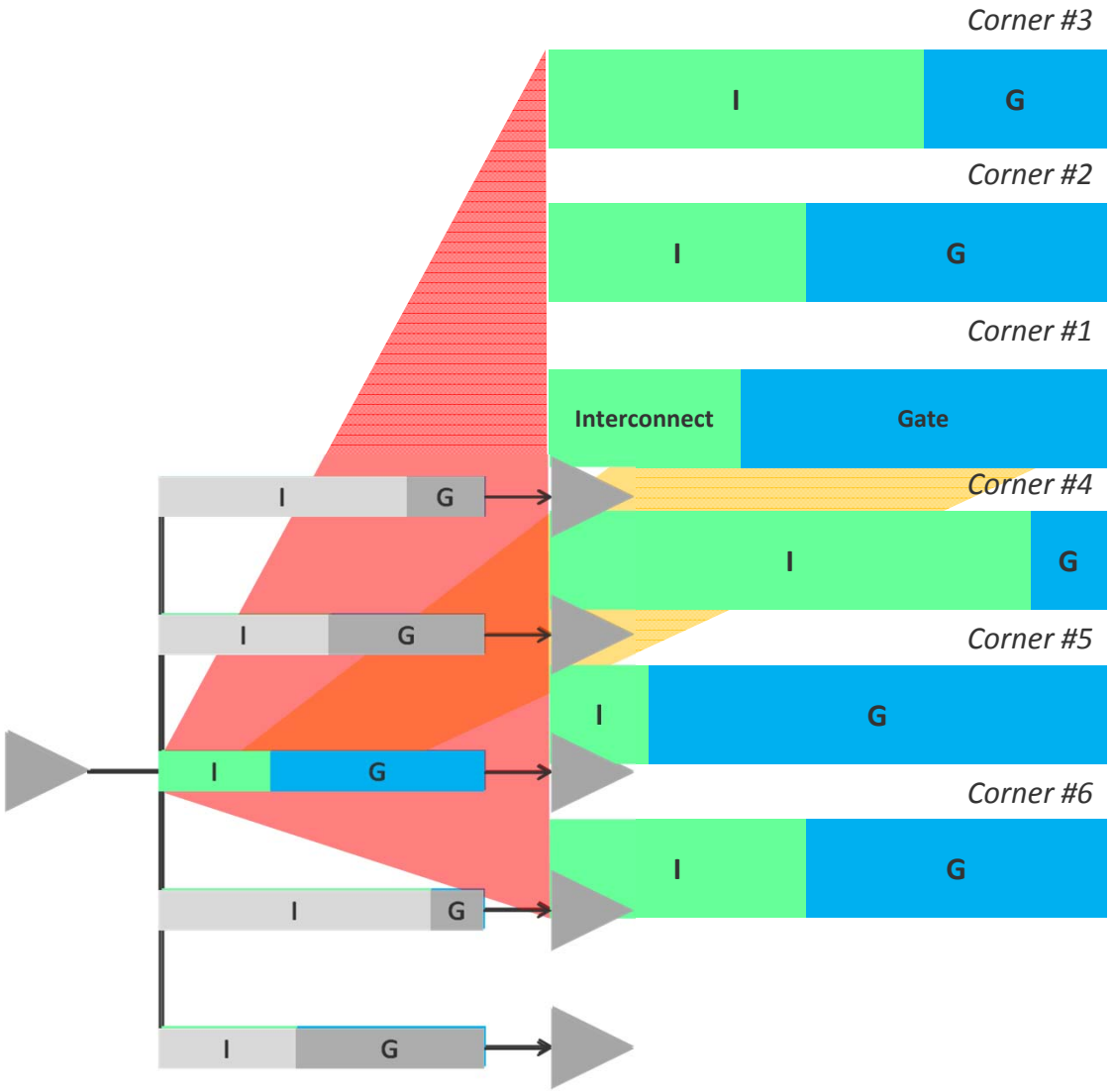


Functional Mode - One flop group DOES NOT talk to the other group

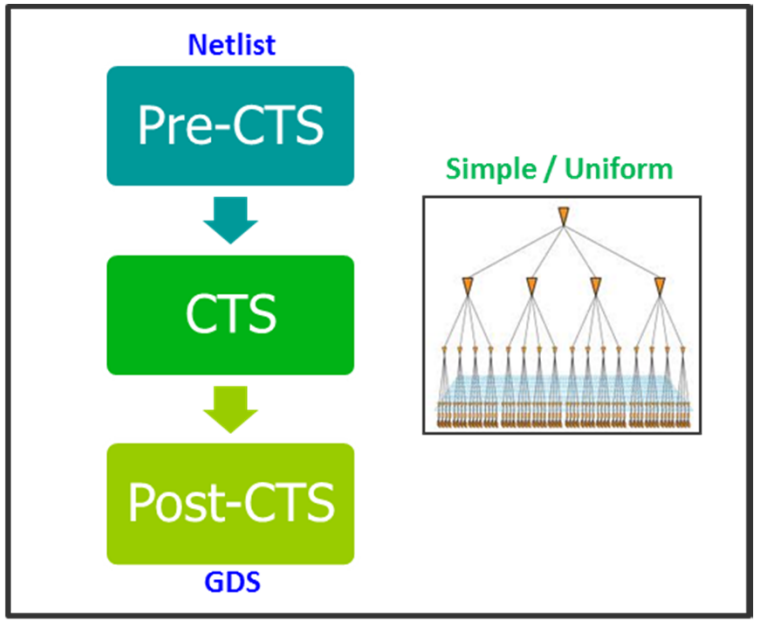
Scan Mode - Each flop group talks to other through the mux

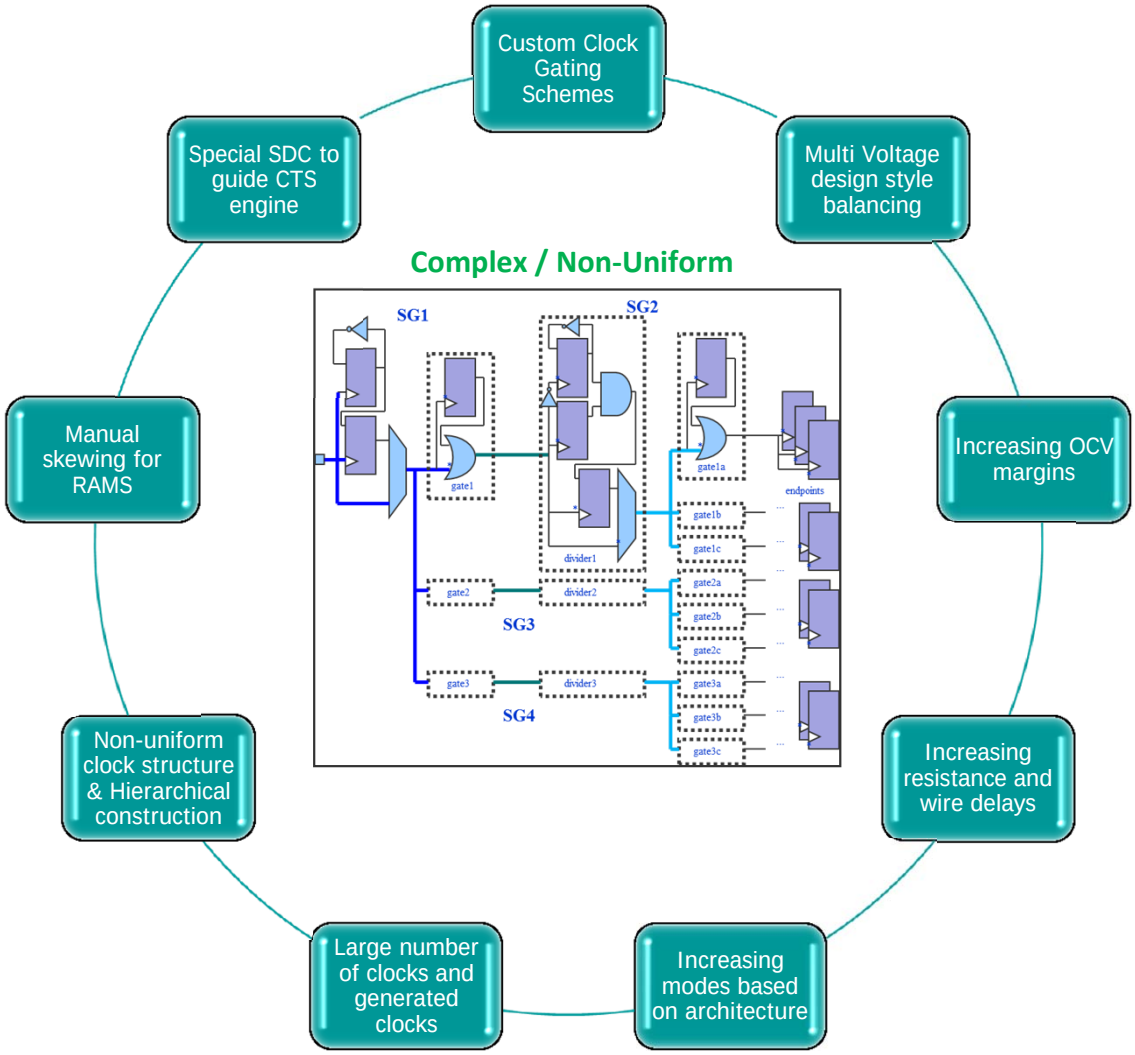
Variation Effect on Clock Trees

- Clock tree variation across process corners has significant impact on skew
- Increasing OCV margins causes timing closure challenges
- Wire delay dominates path delay due to increasing resistance
- Increases iterations for timing convergence



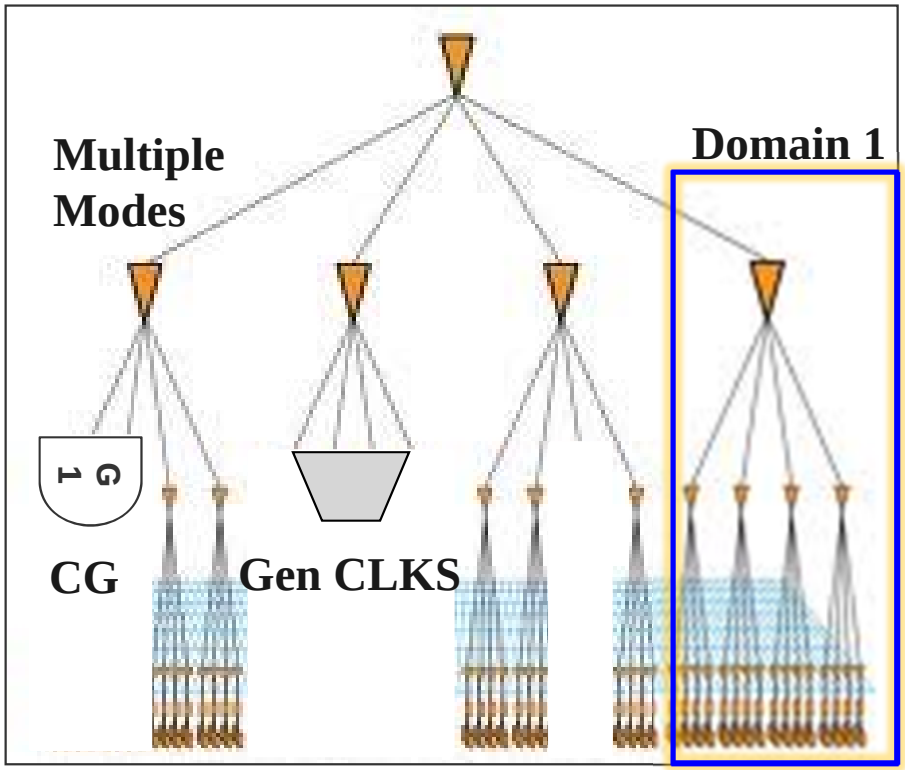
Increasing Complexity of Clock Tree Synthesis





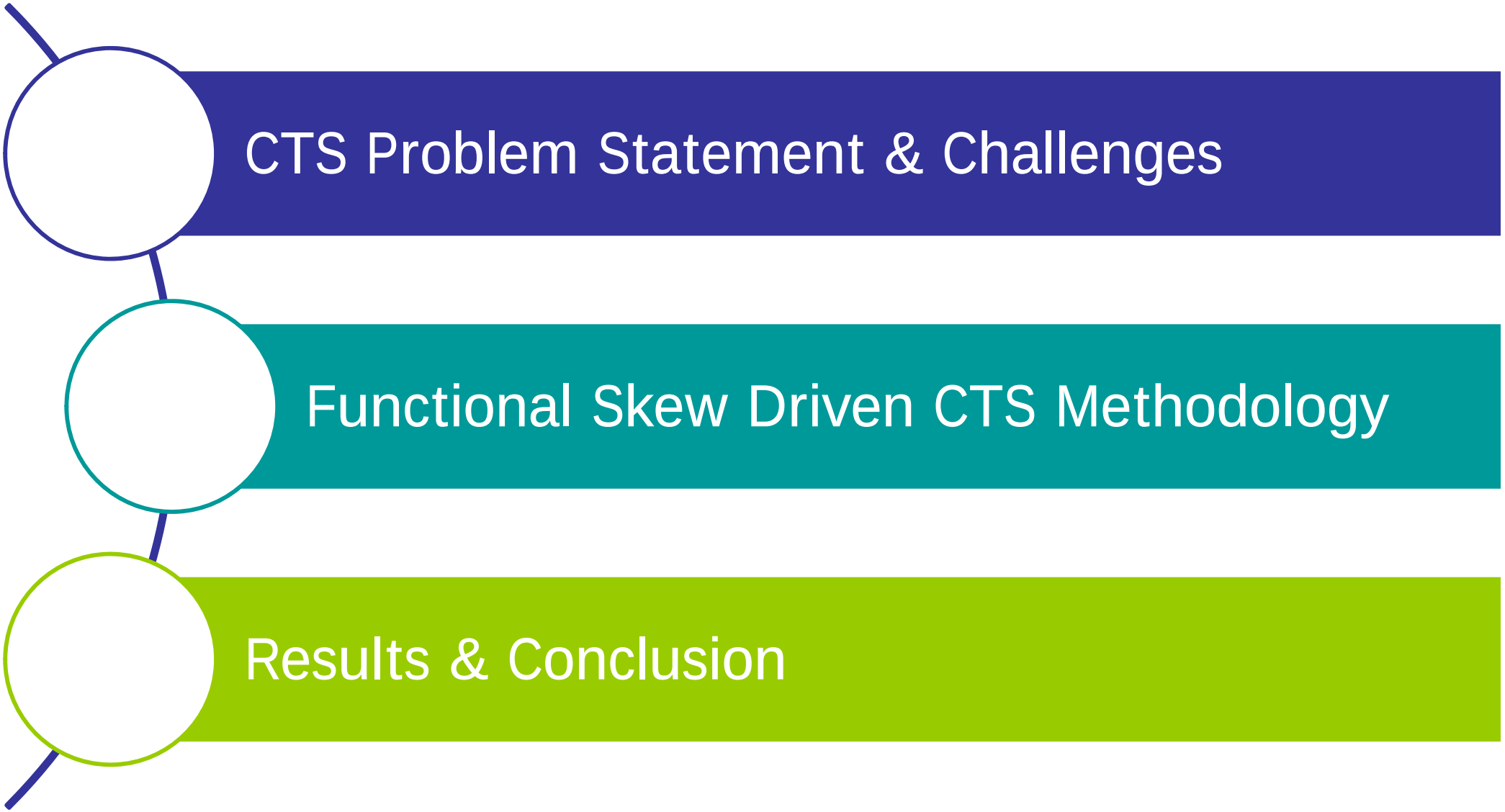
CTS Problem Statement Revisited

- Identifying proper balancing requirements across multiple sub-trees
  - Accounting for multiple power domains and modes
- Achieving Smallest buffer and routing resources & best performance
  - (enable timing, MCMM timing closure, overall post-CTS design TNS/THS)



NO LONGER AN IDEALIZED SINGLE NET ZERO-SKEW BUFFERING PROBLEM!

Outline



Functional Skew Driven CTS - Concept

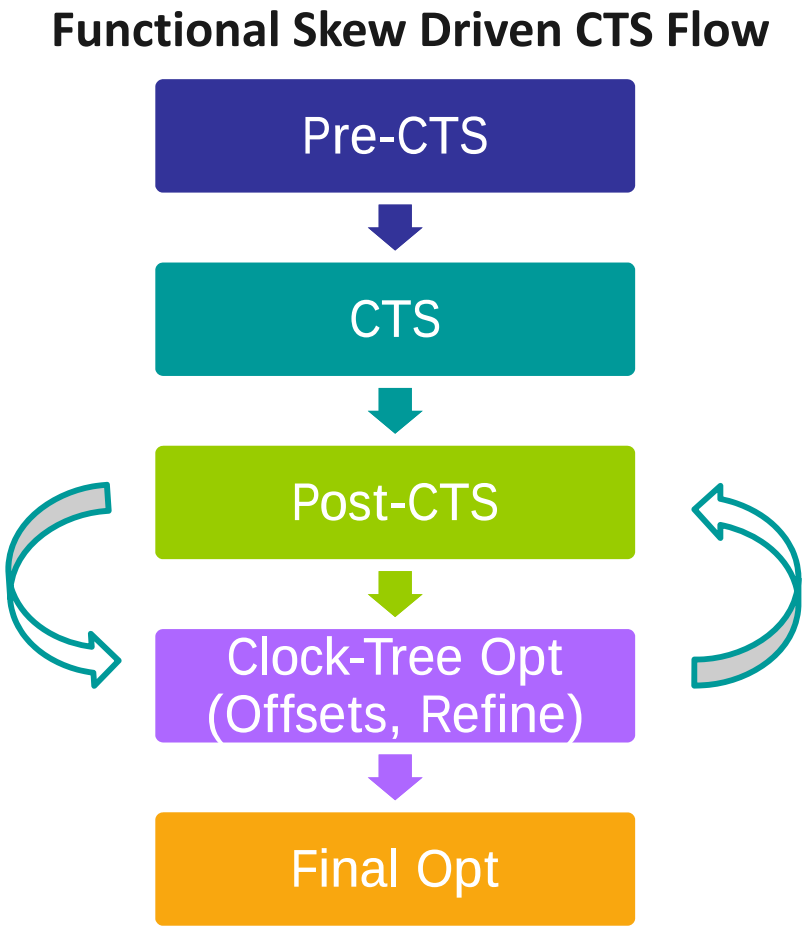
- Traditional CTS flow
  - CTS constrained by only skew, slew and latency targets

  - Despite meeting CTS targets, huge jump in design TNS and THS post-CTS
  - Significant power impact due to higher buffer count
  - Requirement for a new methodology

- Proposed Flow – Functional Skew Driven CTS
  - Identify sub-tree balancing requirements (manual or automatic)
  - MCMM & OCV aware optimization to help with overall design closure problem

Functional Skew Driven CTS - Methodology

1. Improve TNS/THS across all modes/corners by selective speedup/slowdown of portions of the current tree
   - Speedup based on current clock path delay
   - Slowdowns based on impact to tree latency
2. Refine the current clock tree
   - Repeat timing optimization on data-paths
   - If design timing is not met, loop back to Step 1



Performing Timing Optimization In CTS

- For clock-tree optimization need to consider entire (or large) portions of design in one shot
  - Analyze all functional timing paths in all active modes and corners
  - Use existing tree to identify OCV-timing improvement opportunities
- LP formulation can be used as a solver
  - Restrict WNS/WHS fixing to otherwise hard-to-meet paths
  - Focus on TNS/THS improvements

LP Construction

- Modeling setup/hold timing constraints
  - ***(Setup) Tl + MaxPathDelay <= Tc + Tp – RT***
  - ***(Hold) Tl + MinPathDelay >= Tc – RT***
    - Tl → clock arrival at launch
    - Tc → clock arrival at capture
    - Tp → clock cycle shift adjustment
    - RT → Includes all required time adjusts (incl margins, pessimism etc)

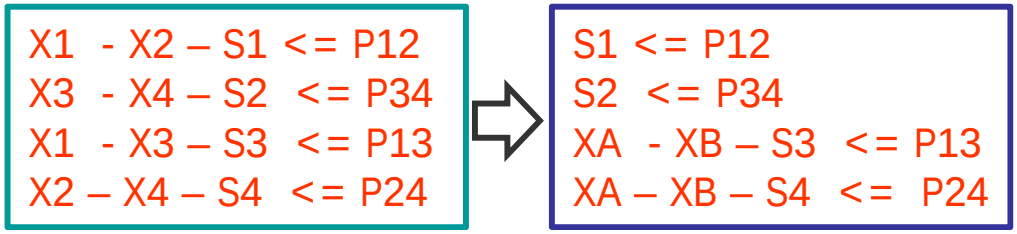
- Adding delay offset variables and rearranging
  - ***(Setup) Xl - Xc <= PathSlack_lc***
  - ***(Hold) Xc – Xl <= PathSlack_ec***
    - Xl → Incremental clock arrival offset at launch
    - Xc → Incremental clock arrival at capture

LP Construction

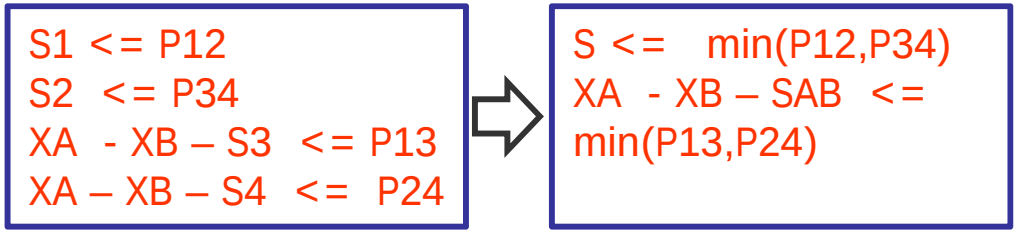
- Slack variables
  - ***(Setup) c1*Xl - c2*Xc – S1 <= PathSlack_lc; S1>=0***
  - ***(Hold) c3*Xc – c4*Xl - H1<= PathSlack_ec; H1 >=0***
  - S1, H1 → New LP variables representing setup/hold slacks
  - c1 .. c4 → Constants used to model corner scaling, derates, etc
- WNS objective: min(slack vars)
  - ***min (S_i) or min (H_i)***
- TNS objective: min(sum_of_slacks)
  - ***min(∑Si) or min(∑Hi)***
- Area objective:
  - ***min(∑Xi)***

Curbing LP Complexity

- Clustering Delay Variables
  - Use initial tree to define ‘timing’ clusters
  - All sinks belonging to the same sub-tree can be assigned the same delay variable

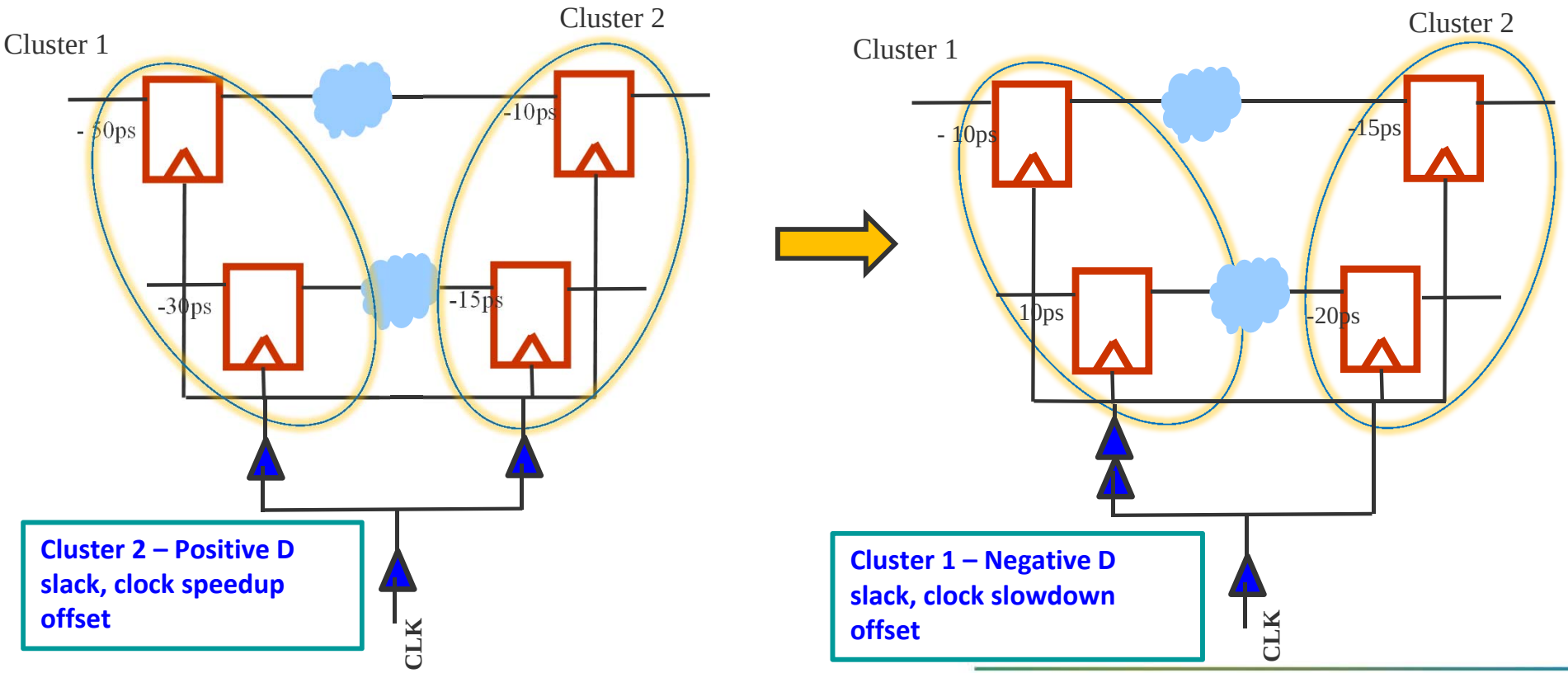


- Clustering Slack Variables
  - Use same slack variable between same set of delay variables

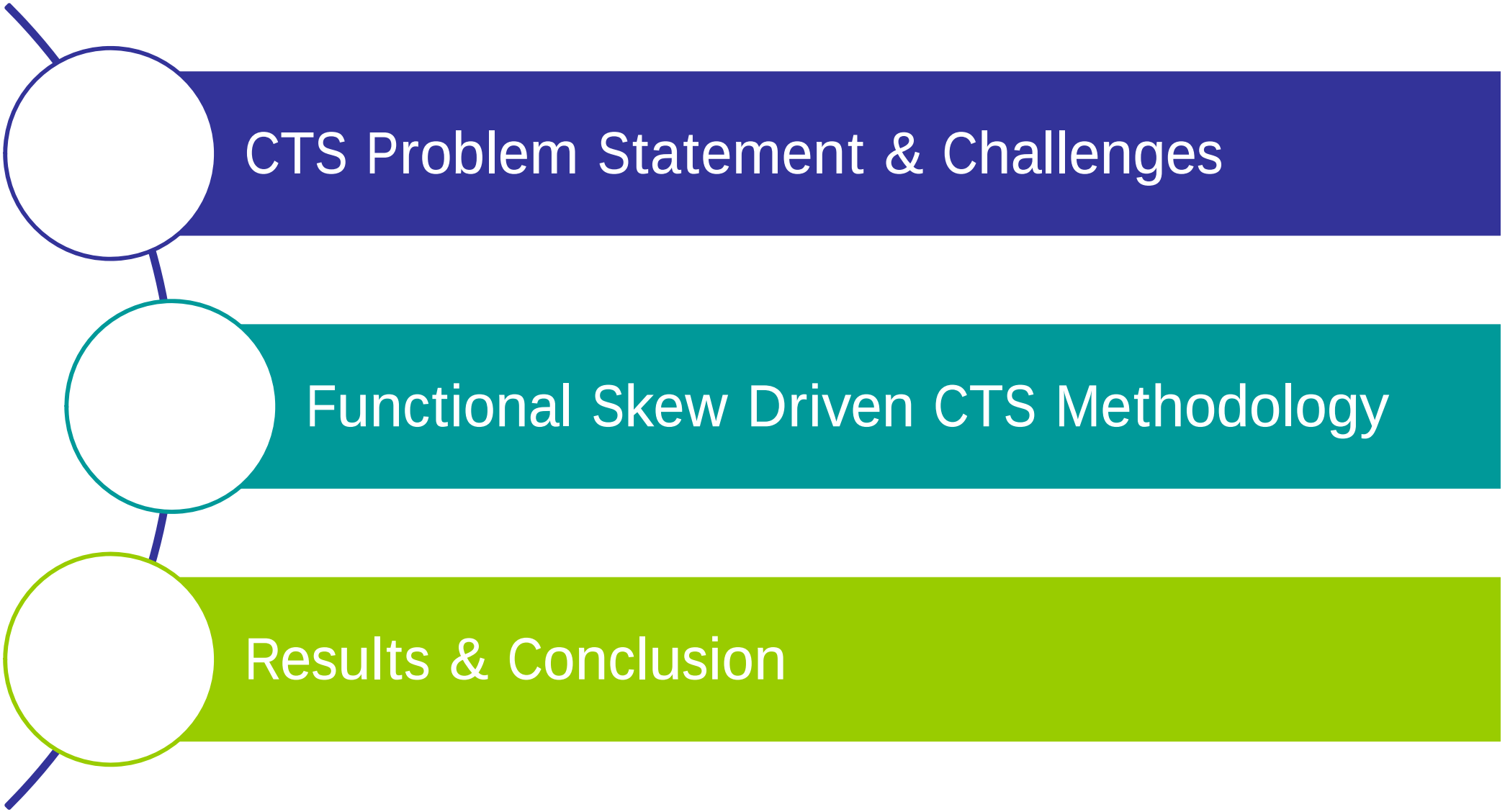


Refining The Clock Tree

- Specified Delay Buffering (SDBP)
  - Given an existing buffer tree B1 with initial path delays of pi for each sink I of this tree:
    - Construct modified buffer tree B2 with path delays of (pi + xi) for each sink I



Outline



Functional Skew Driven CTS – Case 1

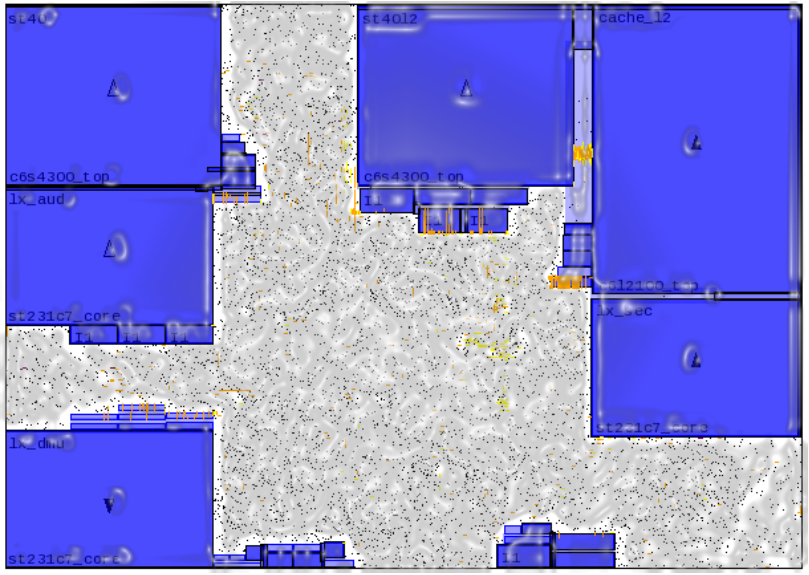
15 Million Gates

Technology 45nm

Metric		Setup (ps)			Hold (ps)		
	Corner	WNS	TNS	FEP	WHS	THS	FEP
CTS	Worst	-12,065	-149,720	278	-1,765	-7,877,176	23,164
	Best	-na-	-na-	-na-	-1,186	-2,465,929	22,007

Functional Skew Driven CTS – Case 2

- First trial – WNS Without skew driven CTS:
  - -330ps post cts
  - -311ps post route
- Second trial – WNS Functional Skew-driven flow
  - -80ps post cts,
  - -97ps post route



- 40nm Block, 1.1M Instances

Functional Skew Driven CTS – Case 3

- 3M instances
- 5 Partitions
- 40nm Technology
- Could not close top level timing
- Employed skew driven CTS
  - 98% TNS and 18% THS Reduction

Instances	Partitions	TNS reduction	THS reduction
3 Million	5	98%	18%

Functional Skew Driven CTS – Case 4

- Tested on three 28nm blocks
- Significant reduction in TNS & THS

Blocks	Inst	Initial TNS	Final TNS	% Imp	Initial THS	Final THS	% Imp
B1	14M	-577,179	-461,597	**20%**	-108,064	-89,262	**21%**
B2	23M	-4,668,221	-4,177,748	**11%**	-234,998	-145,386	**38%**
B4	6M	-3,350,400	-748,816	**78%**	-1,635,945	-1,059,284	**35%**

Conclusion

- Functional skewing is necessary to meet design timing in complex scenarios
- Identifying these is the most time-consuming and challenging problem
- Functional skewing can help with significant timing improvement
- Clock tree construction and optimization needs to be considered holistically, esp. for low power SoCs

www.mentor.com