Improving RTL quality and reducing Backend cycles by using Synopsys RTL-Architect at the Chip top and Subsystem

Micro-Architecture Exploration and Refinement for Optimal PPA

Mohammad Javed (ST Microelectronics)
Penugonda MAHESH (ST Microelectronics)
Ashi Goyal (Synopsys)

Challenges of Automotive SoC Implementation

Trends in Automotive Industry

Rapid Market Evolution:

- The urgency of meeting market release schedules.
- The competitive edge gained through timely product launches.

Architectural Complexity:

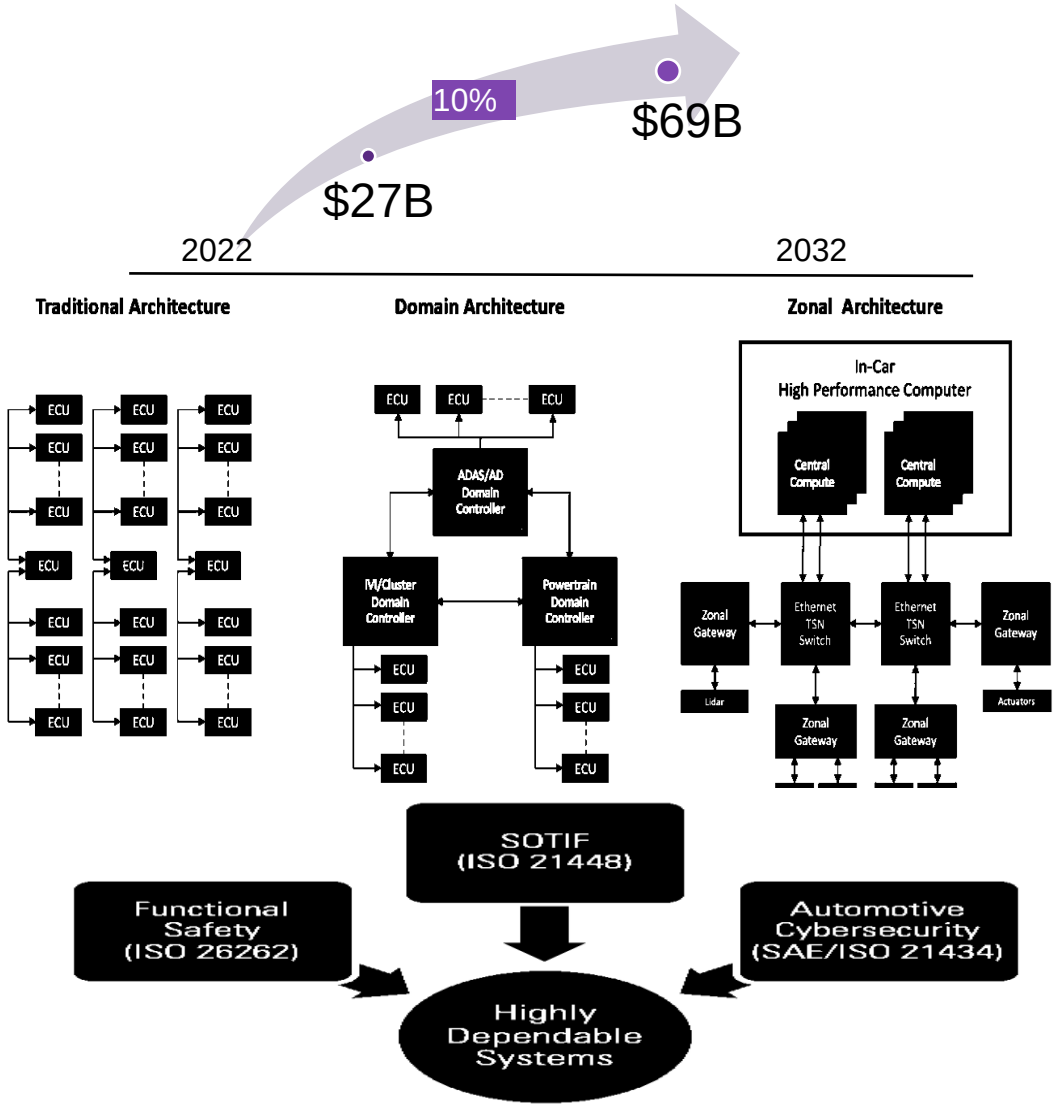
- The shift from traditional to domain-based and to zonal vehicle architectures.
- The trend of MCU aggregation to support diverse vehicle functions in lesser MCU.

Performance, Power and Area Optimization:

- The imperative to balance power, performance, and area (PPA).
- The integration of real-time applications demanding high-speed processing.

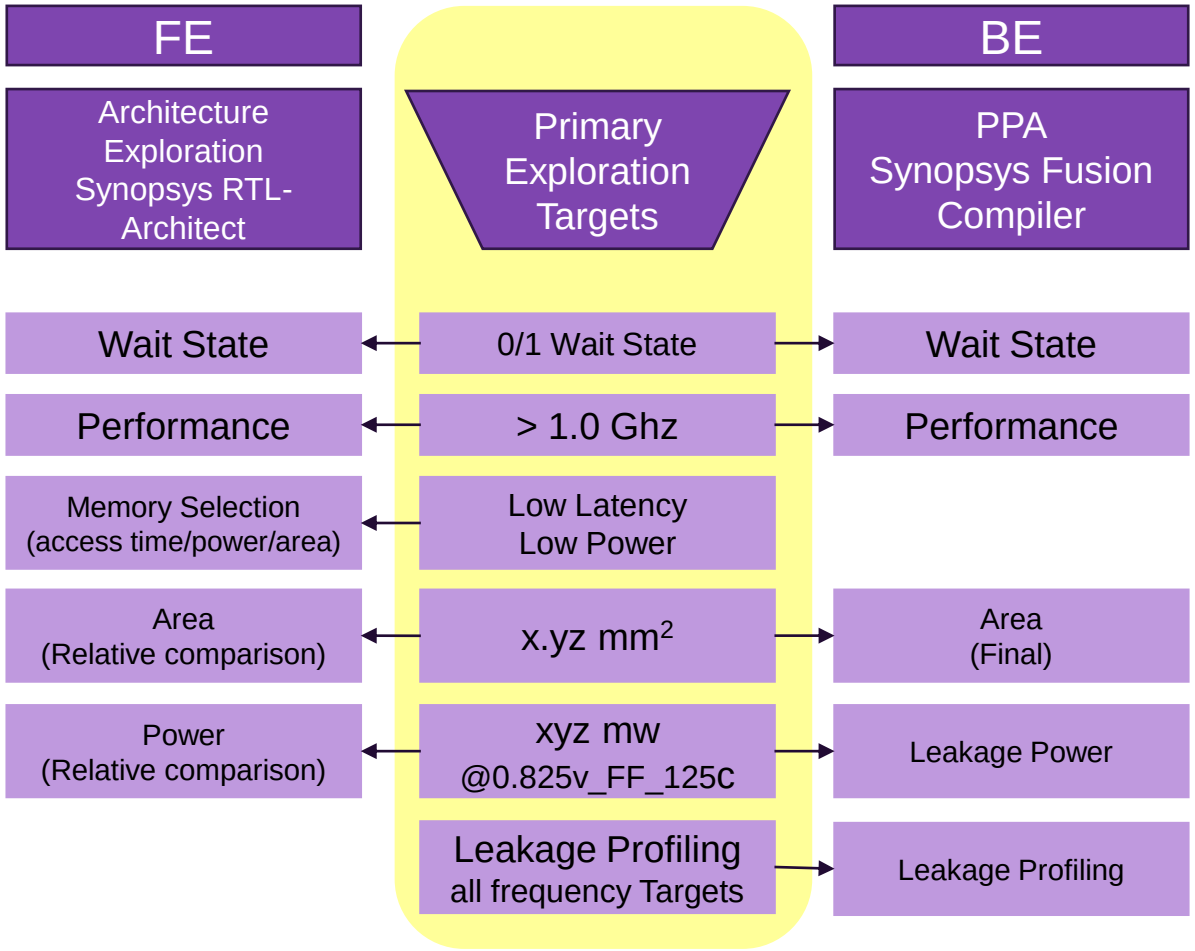
Regulatory and Safety Standards:

- The stringent safety and security requirements for automotive SoCs.
- The challenge of adhering to these standards amidst increasing system complexity.



Problem Statement

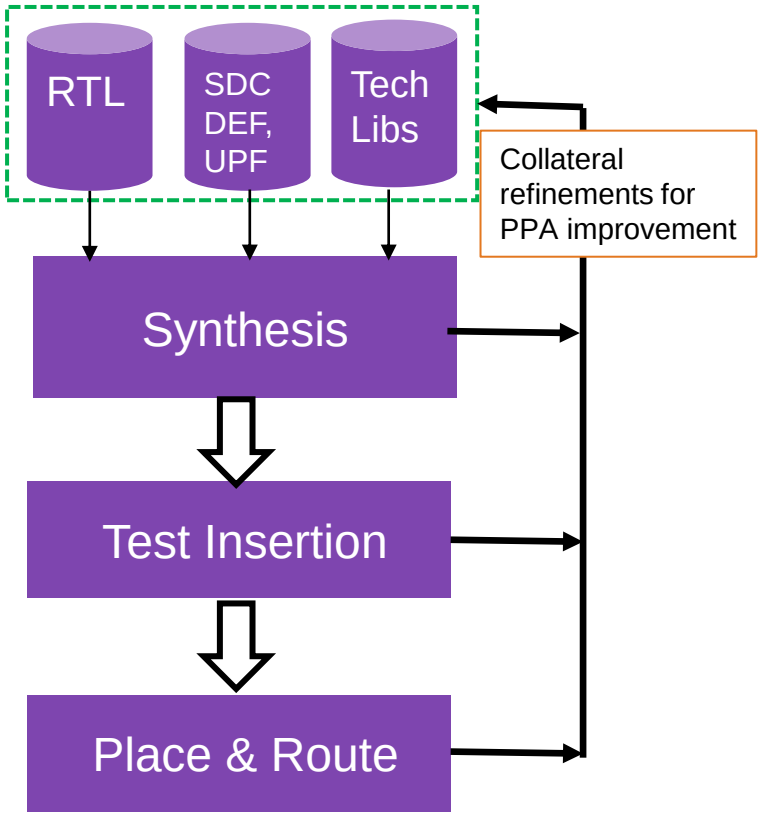
Design Challenge



Design Challenges		
Performance	Target Frequency	> 1Ghz
	Low Memory Latency	0/1 Wait State > 1 Ghz Clock
	Level 2 Cache	To reduce miss latency of DDR4 access
Functional Safety	ASIL-D/ASIL-B Support	Replication and well as ECC/Parity support
Power	Power-Budget of 500mW	Select Low-Power SRAM put Clock-Gating
Area	x.yzum2	Area Optimized SRAM Selection
Security	Native Encryption/Decryption	For inter-chip communication and
Technology	TSMC-N7	Safety Architecture for TSMC Memories and validation

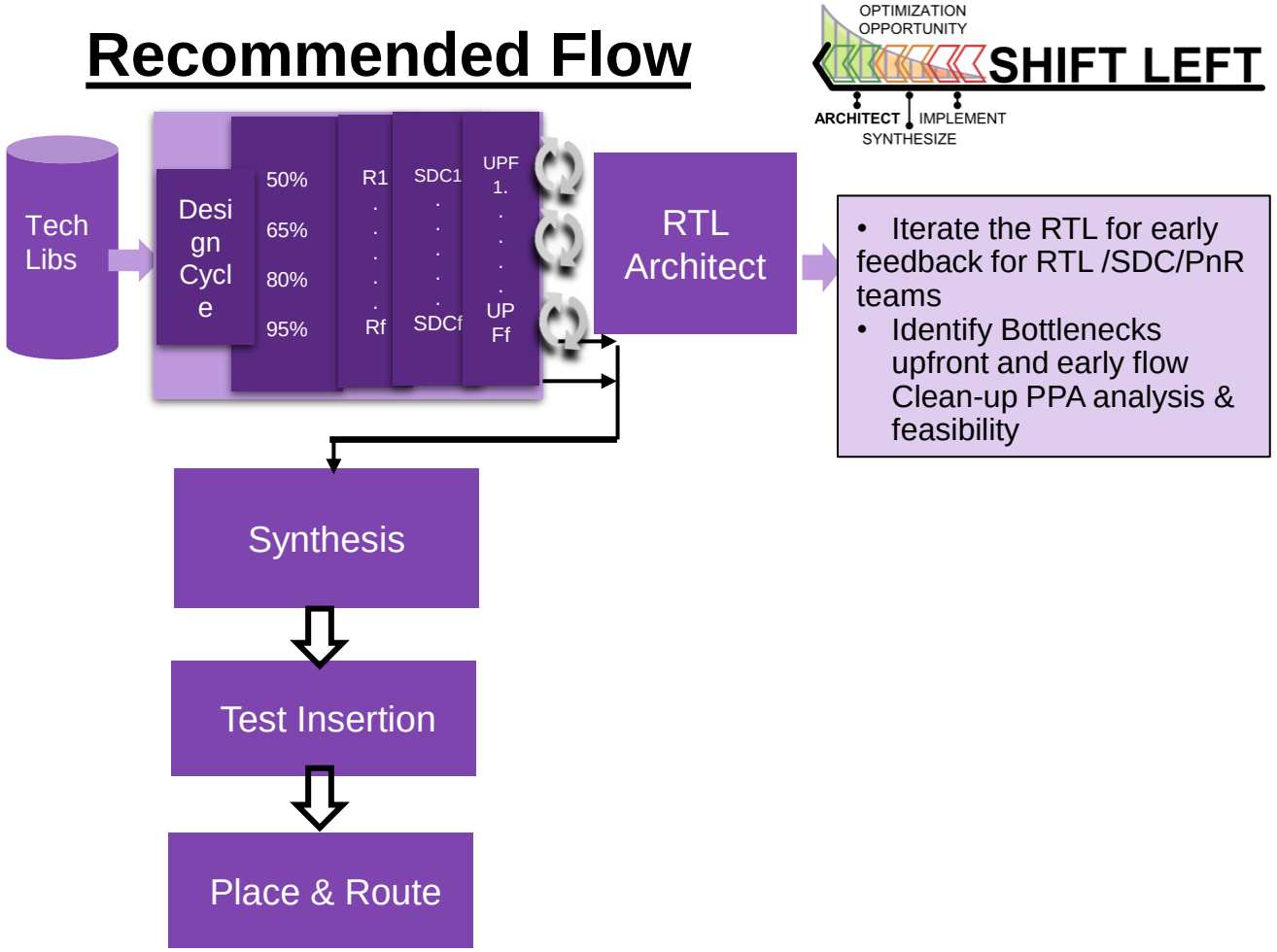
Traditional Flow→ Recommended Flow

Traditional Flow



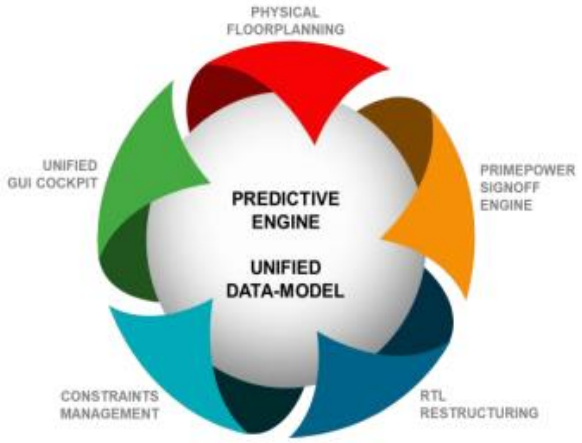
Iterative, Risking Schedule & Quality..

Recommended Flow

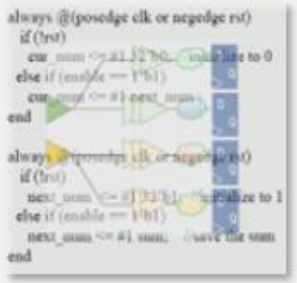


Predictable, Convergent & Scalable !!

Synopsys RTL-Architect Features

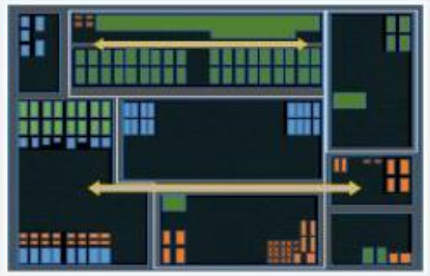


Predictive Engine



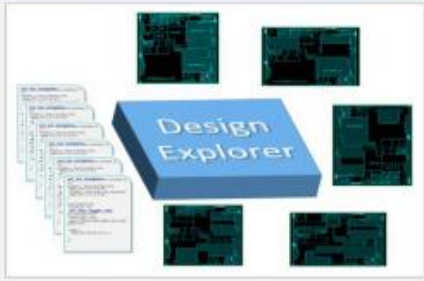
Fast gate prediction
Scalable from block to SoC

Design Planning



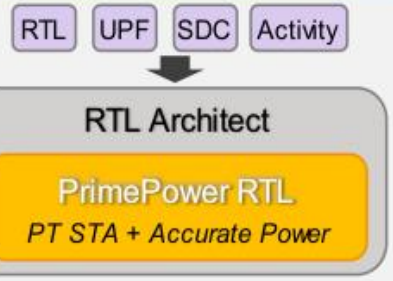
Automatic & comprehensive floor planning features

Parallel Exploration



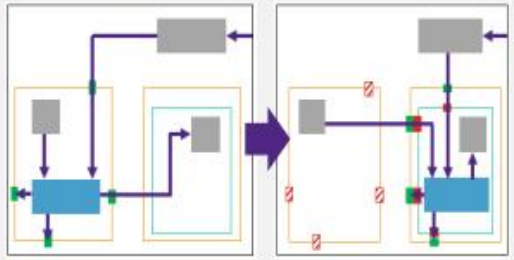
Sweep and compare multiple design parameters

Power Analysis



PrimePower-driven avg, peak, and glitch with timing & activity

Restructuring



Quickly optimize RTL hierarchy for better PPA

Constraints Management

UPF
SDC
Promotion
Demotion
Checks

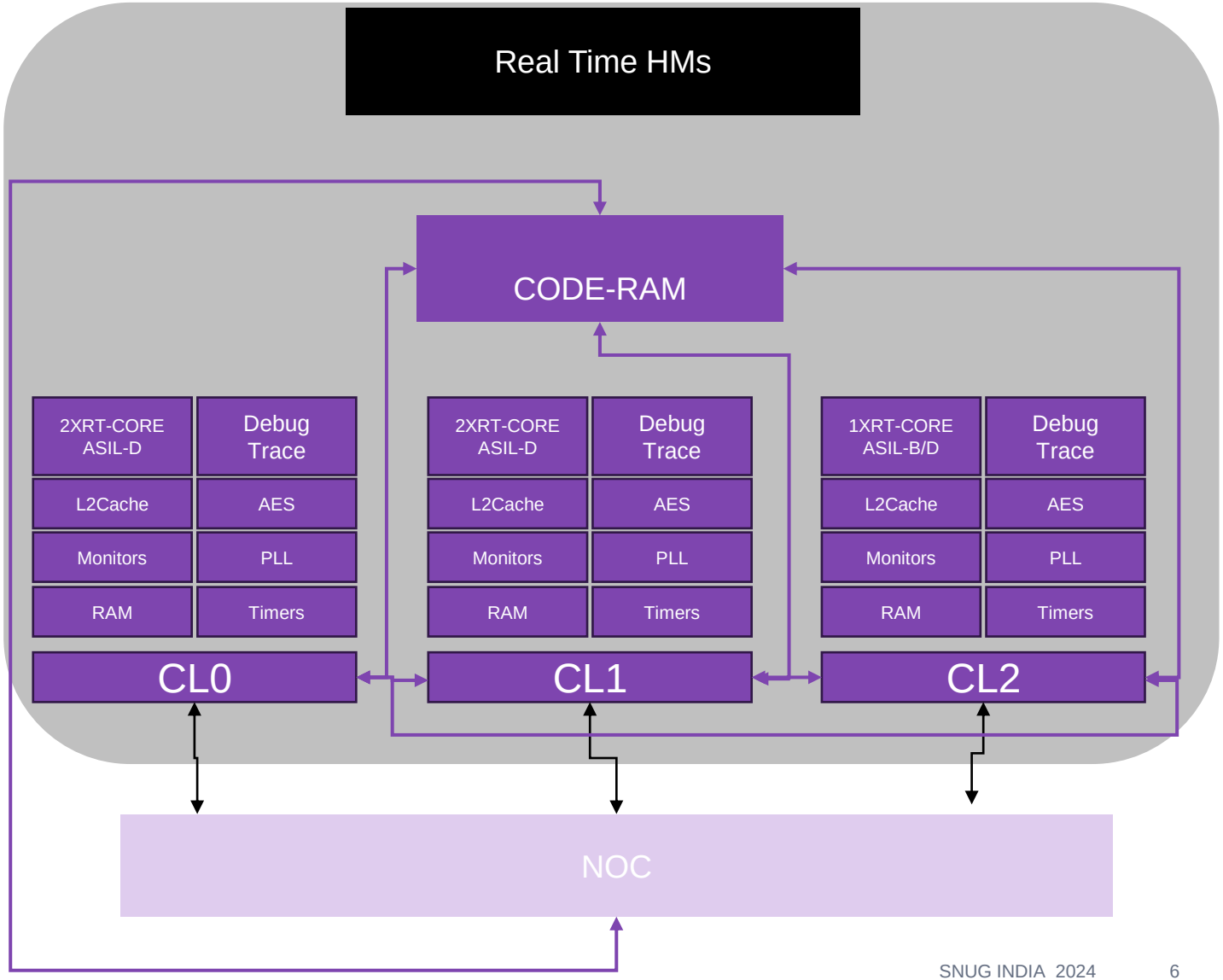
Track and maintain timing and power constraints

Unified GUI

Single GUI to analyze, debug, and pinpoint issues with RTL

Subsystem Overview

- 10 Real Time Core
  - 5XLockStep / 4XLockstep + 2-Split
  - Clock Frequency more than 1.0 Ghz
  - Low Latency SRAM.
  - Level 2 Cache.
  - Dedicated AES Engine.
  - Process/Clock Monitors
  - Shared SRAM for Real-Time Code.
  - Functional Safety



Real Time Subsystem

Challenges and Opportunity

- SoC Partitioning to Realizable Hard Macros for parallel execution and reduce TAT
- Shift-Left Methodology to perform the essential design implementation steps early
- Parallel Exploration for various design parameters and updated design options
- Power Analysis using RTL-Architect and Prime Power for Static and Dynamic Power

Activities Performed

Memory Selection and Validation – **Done using Ad hoc Excel**

Micro-Architecture Refinement of Interconnect and Other Critical IP(s) – **More than 50 iterations Tried**

Fusion Compiler/RTL-Architect File-Set and Design Constraint Validation – **No Issue reported from BE**

Timing Constraint and Exception Validation – **No Issue at BE for Constraints Issues**

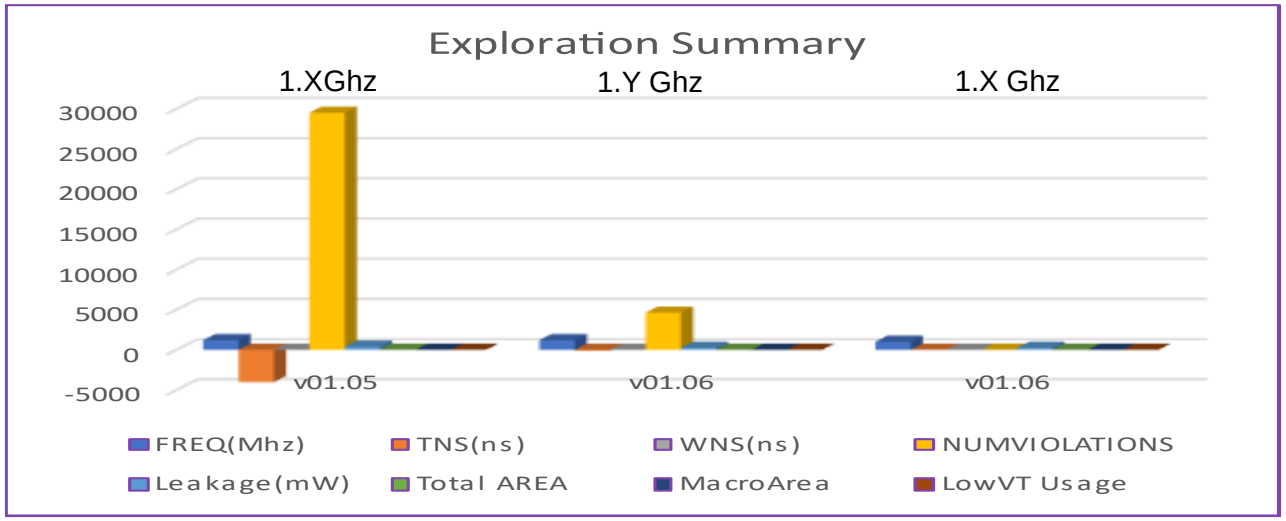
Hold Fix analysis with Latch Insertion on critical hold path(s)

Internal IP Design update for Hold Fix.

Result…

FE PPA Phase: 16 WW

Design Version	TNS (ns)	WNS (ns)	NUM	Memory Leakage* (mW)	Total AREA (um2)	Macro Area (um2)	Low VT Usage
v01.05-1200 2022-WW47-1,x Ghz	-4020	-0.707	29486	393	2.57	2.28	4.7%
v01.06-1200 2023-WW12-1. Ghz	-126	-0.174	4609	245	2.65	2.29	4.6%
v01.06-1000 2023-WW12-1.x Ghz	-0.04	-0.04	1	245	2.62	2.29	4.1%



Implementation Phase(Fusion Compiler): Status

- The BE implementation is Ongoing with target frequency more than 1 Ghz
- There is no show-stopper found in BE for the Real Time Cluster that need major design update.

Clusters	Placeable Instances	Memory Leakage* (mW)	Total AREA (um2)	Macro Area (um2)	Low VT Usage
Cluster0(1.x Ghz)	4.5M	294	2.63	1.98	4.8%
Cluster1(1.x Ghz)	4.5M	294	2.61	1.98	4.8%
Cluster2(1.x Ghz)	3.5M	294	2.50	1.98	4.74%
Cluster3(1.x Ghz)	1.2M	325	4.08	3.92	4.90%

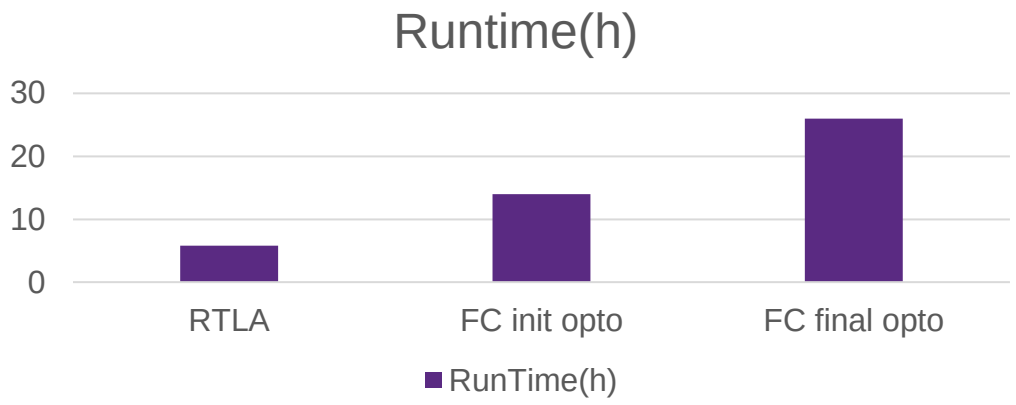
* Leakage power are from .libs and without uplift factor

10 Correlation(Fusion Compiler : RTL-Architect) at PPA Phase

RTL-Architect Max Timing Summary[1 violations]

	Total	Reg->Reg	In->Reg	Reg->Out	In->Out
WNS	-0.04	-0.04	0.00	0.00	0.00
TNS	-0.04	-0.04	0.00	0.00	0.00
NUM	1	1	0	0	0

RTL-Architect Runtime is almost 3-time faster than Fusion Compiler init opto and 5-times faster than Fusion Compiler final opto



Fusion Compiler Init Opto Max Timing Summary[98 violations]

	Total	Reg->Reg	In->Reg	Reg->Out	In->Out
WNS	-0.14	-0.08	-0.14	-0.06	0.00
TNS	-1.45	-0.93	-0.41	-0.11	0.00
NUM	98	84	12	2	0

Fusion Compiler Final Opto Max Timing Summary[211 violations]

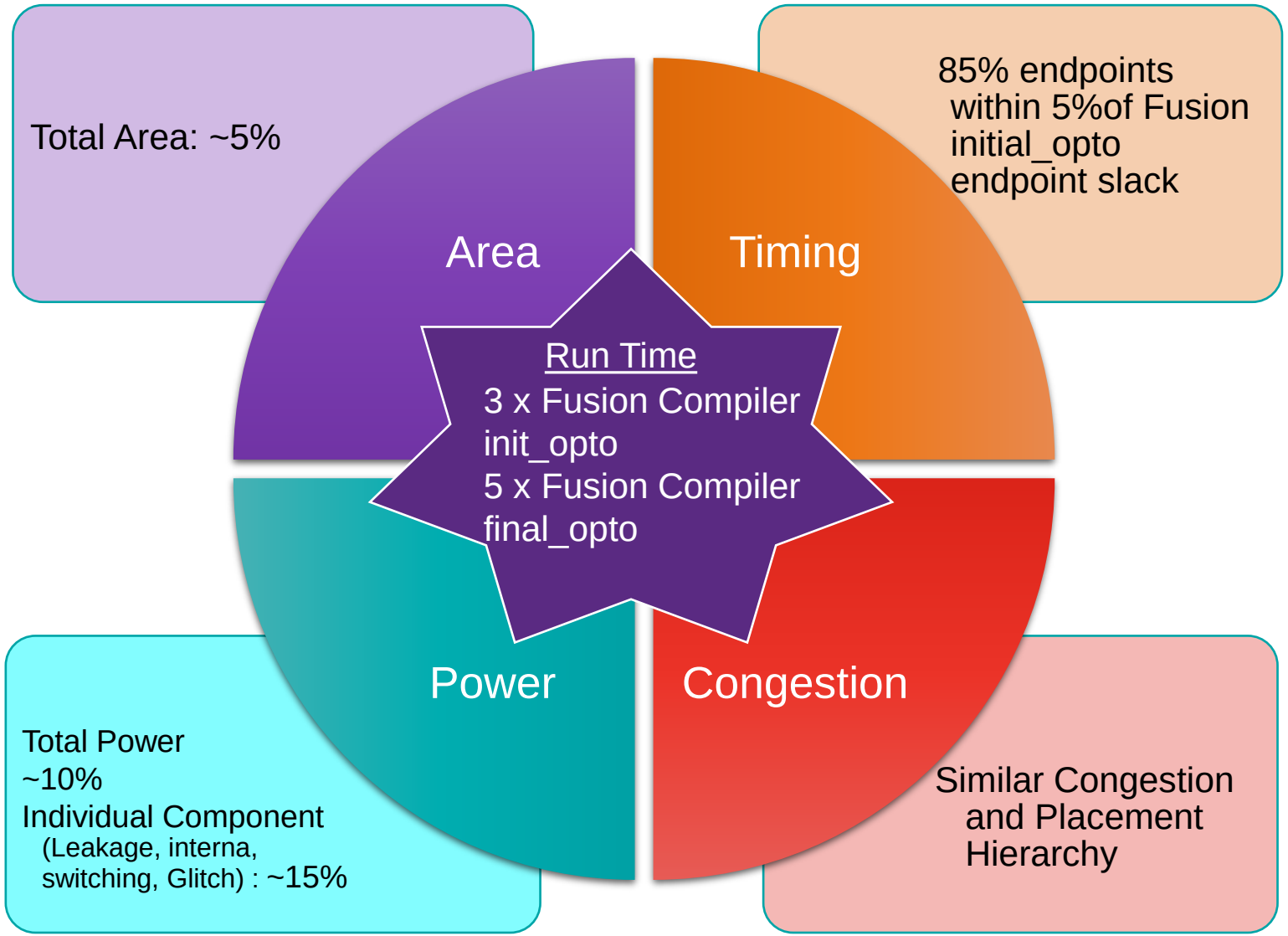
	Total	Reg->Reg	In->Reg	Reg->Out	In->Out
WNS	-0.12	-0.12	-0.07	-0.09	0.00
TNS	-7.41	-6.82	-0.43	-0.16	0.00
NUM	211	183	25	3	0

Timing Summary[RTL-Architect-Fusion Compiler Final Opto] (211 violations)

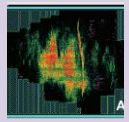
Range	Num Violations
-0.01 – 0.00	121
-0.120 – -0.010	90

- Out of 211 violations only 90 violations are between -0.120 ns to -0.010ns, 121 are less than 0.010 ns

Correlation(Fusion Compiler: RTL-Architect) at PPA Phase

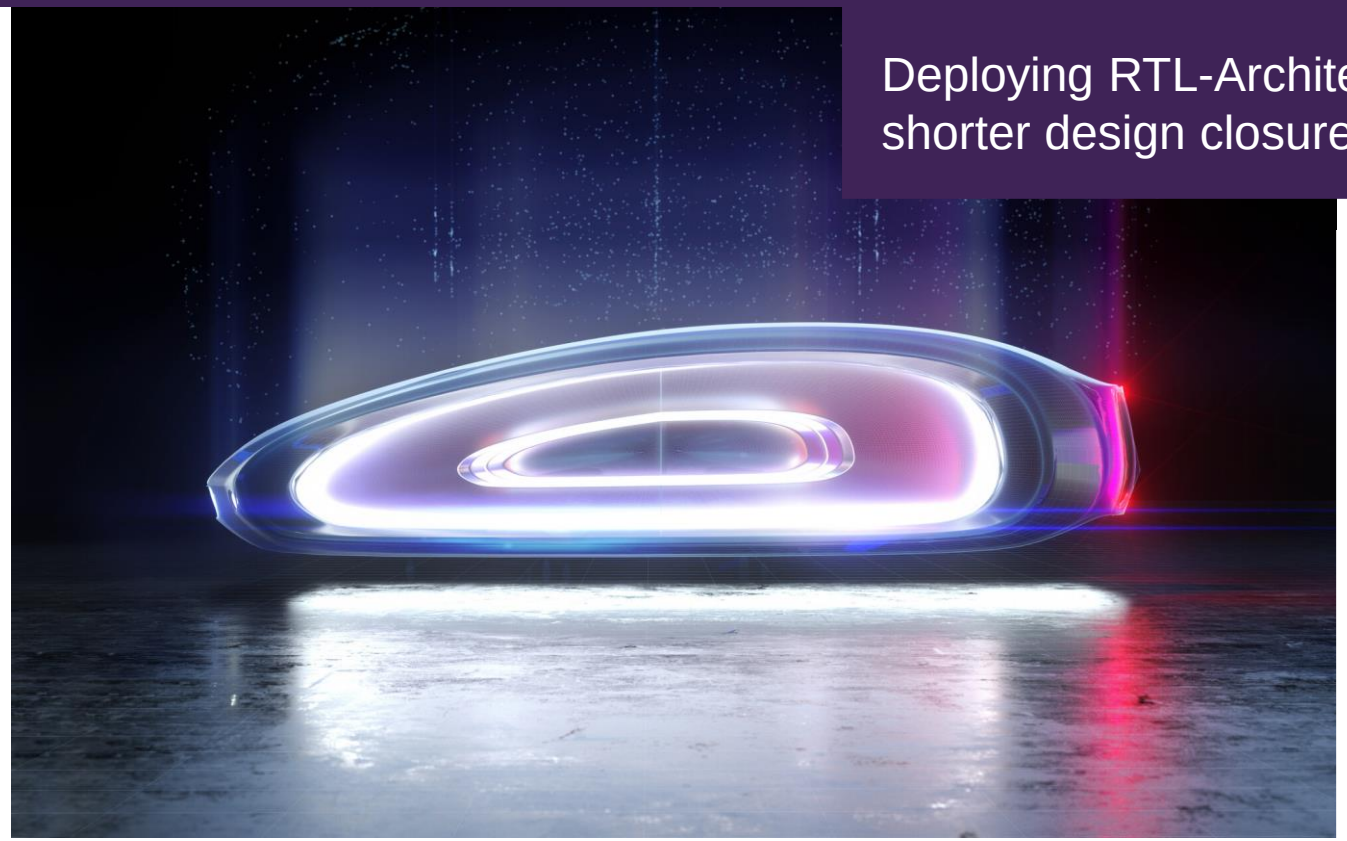


Future-Work

Power	Static Power Calculation and Mitigation Dynamic Power Calculation and Mitigation
Memory Exploration	TSMC/Synopsys Memory Selection and Generation of Mux Logic (Timing/Area/MPW/Power)
VC-Spyglass Static – RTL-Architect	Validation of Optimization of registers by Fusion Compiler/RTL-Architect using VC-Spyglass and RTL-Architect/Fusion Compiler Optimized register list.
 Congestion	Congestion Analysis in RTL-Architect to fix design congestion(s)

Conclusion

Synopsys RTL-Architect Tool provides key technology to designer for improving RTL quality and other collaterals for BE implementation.



Deploying RTL-Architect in design flow improves efficiency and promote shorter design closure.

- Adopting the RTL-Architect tool has improved the design process by identifying challenges early and providing solutions, resulting in streamlined design processes, improved efficiency, and timely delivery of high-quality design collaterals.

- Additionally, the RTL-Architect timing/area/power results are well correlated with the Fusion Compiler, indicating the tool's effectiveness in improving the design process.


THANK YOU
Our
Technology,
Your
Innovation™