Design Convergence

Shantanu Ganguly

Intel Corporation

shantanu.ganguly@intel.com

Agenda

- Problem introduction
- Design by Metrics
- Data abstraction and refinement
- Convergence
- Future Trends, solution ideas
- Conclusion

Design Convergence

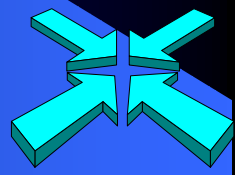
- High Level Convergence Flow
  - Identify key metrics that define success
  - Start design with a budgets for success metrics
  - Iteratively refinement, analysis to measure metrics
  - Find design point that meets specification on all metrics
- Obvious metrics: area, frequency, power, etc
  - Easily quantifiable, can be derived from product targets
- Not so obvious: correctness, reusability, etc
  - Depends on market, use model, price, etc

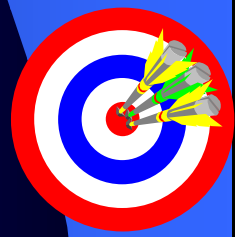
Need for Metrics

- Metrics are an enabler of convergence
- Generation of good metrics does not necessarily lead to convergence!
- Key is using metrics to explore contour of solution space
- Good starting point helps a lot!
- Occasional backtracking necessary...

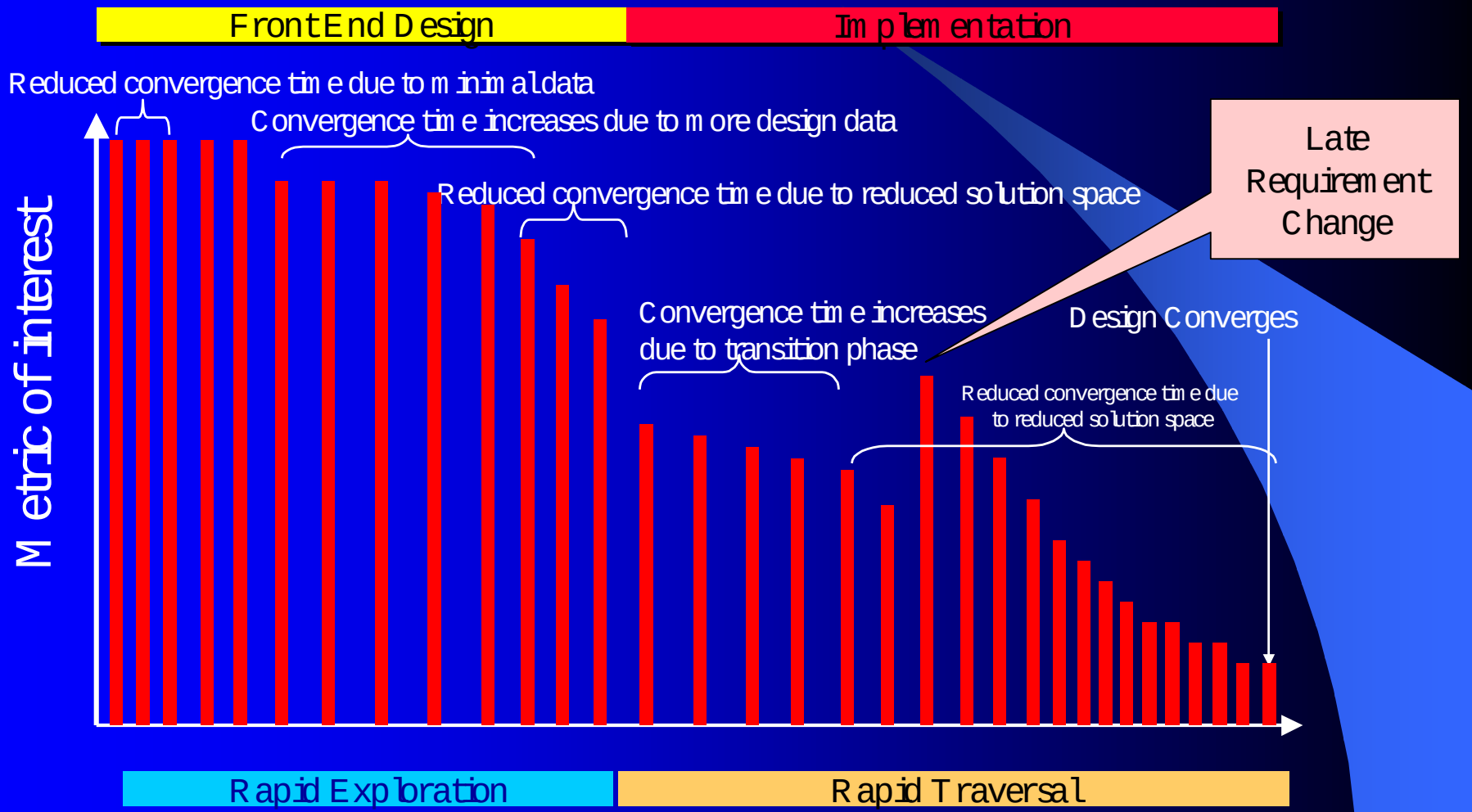
Example: uP design flow

- Historical uP Metrics
  - Area, timing, power, timing, noise
- Example flow around above metrics
  - Chip planning/integration with timing, noise and power analysis.
  - Initially based on estimates and budgets, firms up over time as design gathers detail.
  - Make incremental advances on all fronts simultaneously, occasional drive to push lagging metrics.
  - A successful set of metrics indicate design completion.

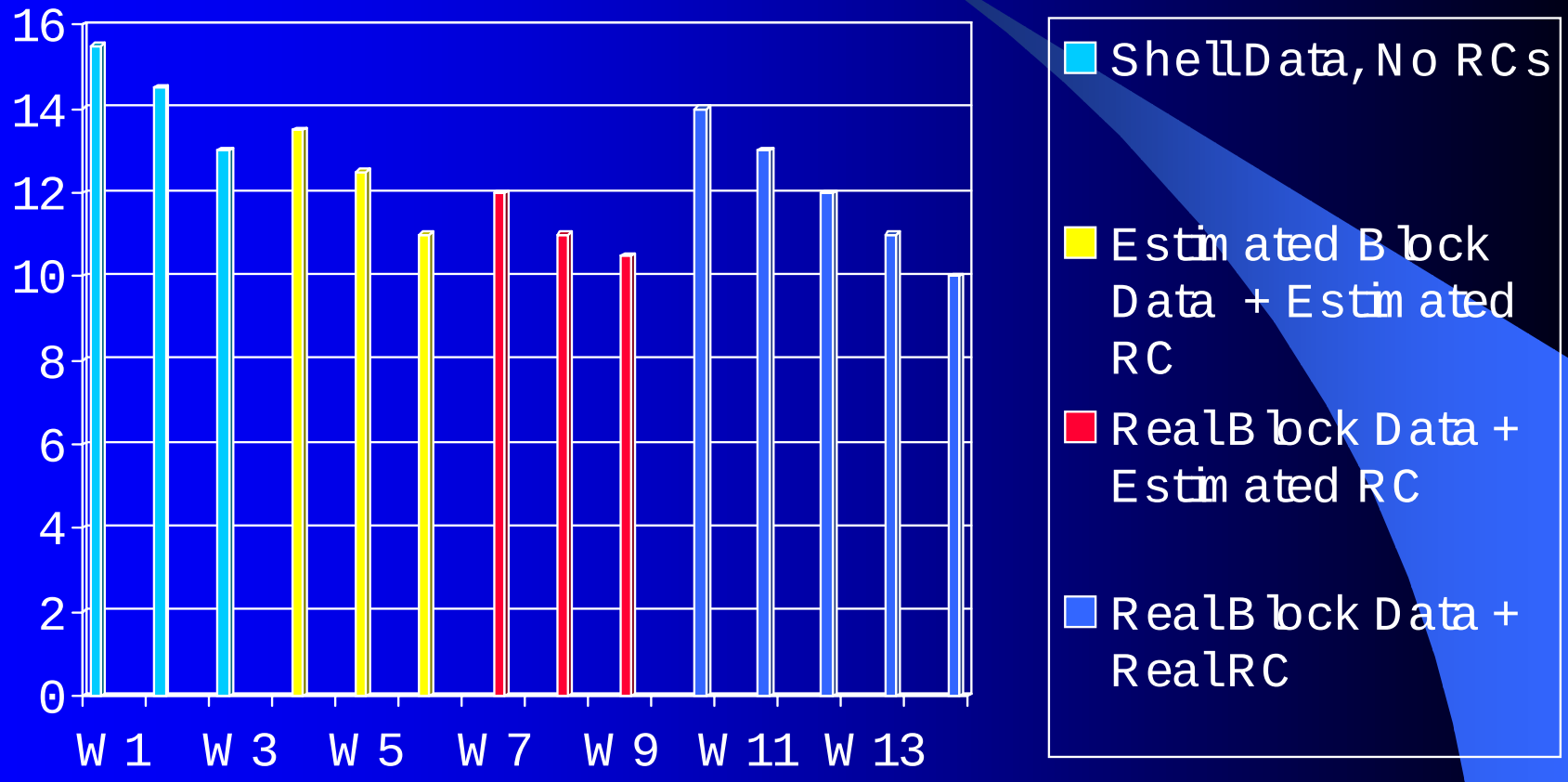




Generic Convergence Profile



Example: Timing Profile



What are relevant metrics?

- Methodology and metrics should revolve around critical path schedule items.
- Example: microprocessor design methodology
  - Most current uP methodologies are RTL driven
  - Circuit & Layout implementation is driven by logic entry
  - This assumes that critical path is creation and validation of a functional description of the product...
- One must pay careful thought to determine the right set of metrics
  - Wrong set of metrics will hinder convergence

When Metrics mislead

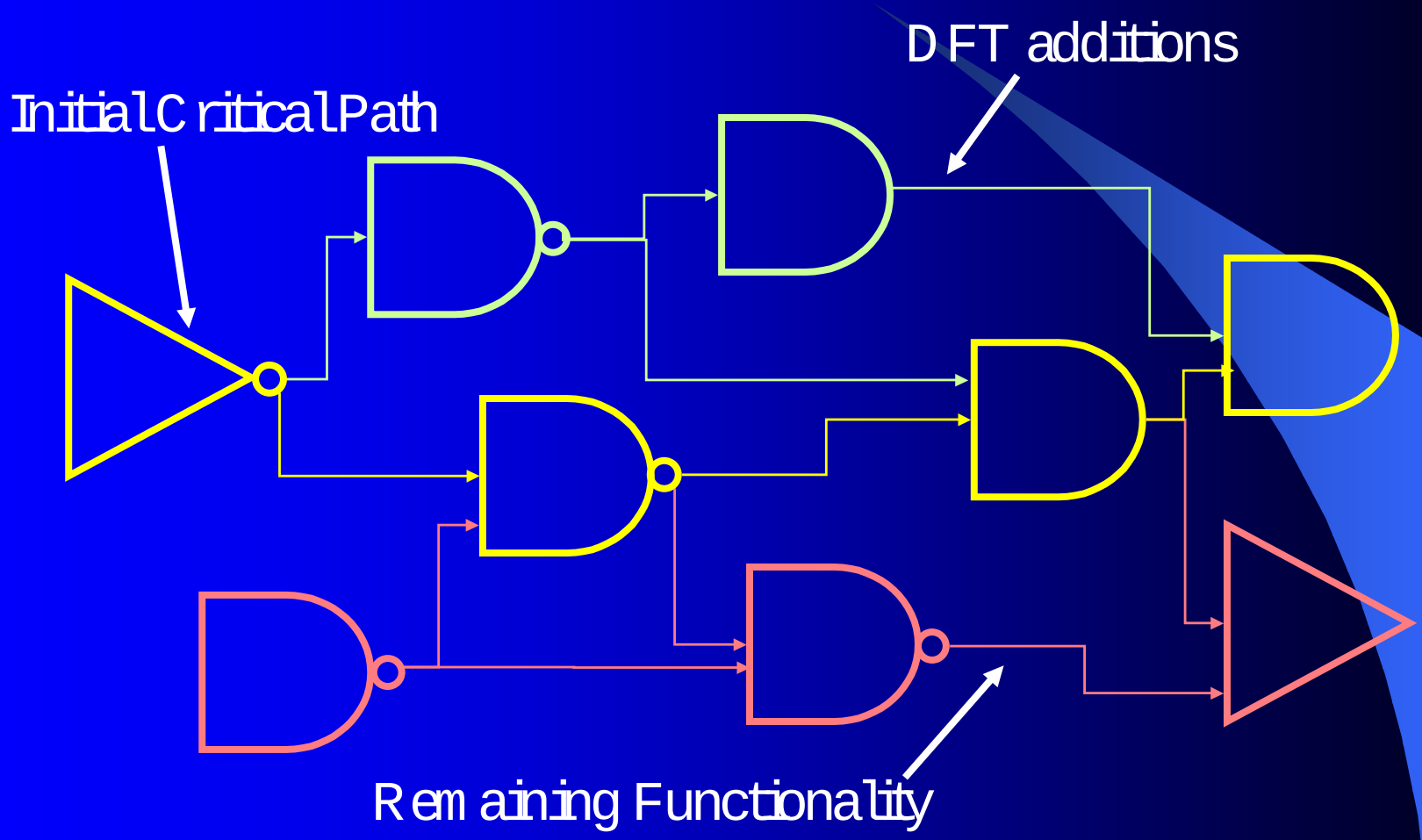
- Analysis engines needed to predict, track metrics
  - Simplifying assumptions used by engines
  - Assumptions may be invalid due to process, design, or product changes

- Consequences
  - Leads to unpredictable behavior of design
  - Causes design oscillation instead of convergence

Convergence using metrics

- Identify metrics that dictate critical path
- Perform up-front analysis to define bounds on critical metrics that must be maintained
- Gradually add functionality to design
  - Such that main metrics are not invalidated
  - Achieve closure on other metrics
- Steal slack in metrics from upstream/downstream to converge

Example: Path based design



Key enablers of convergence

Communication

- Efficient data sharing among tools and people
- Flow re-engineered around tool cockpit

Shorter iterations, fewer people

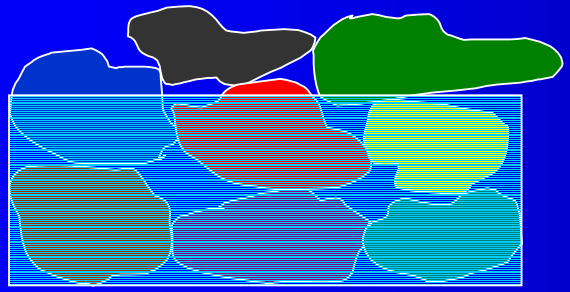
Abstraction

- Exploration with detailed data is costly; use abstract data & estimators
- Refine specs and detail concurrently

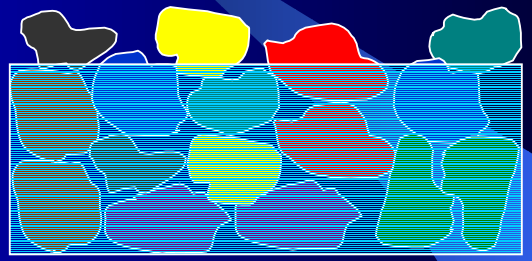
Frequent iterations, higher confidence

Working with abstract data

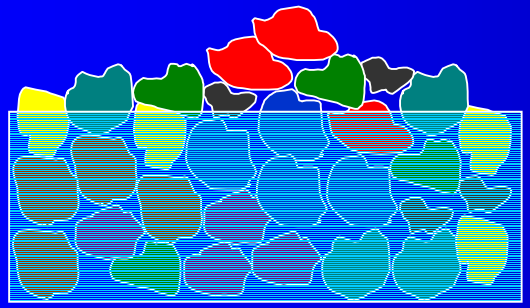
Early Abstracts



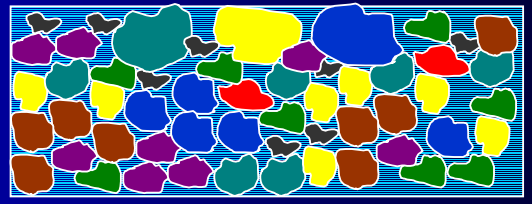
Refined Abstracts



Mixed-Mode

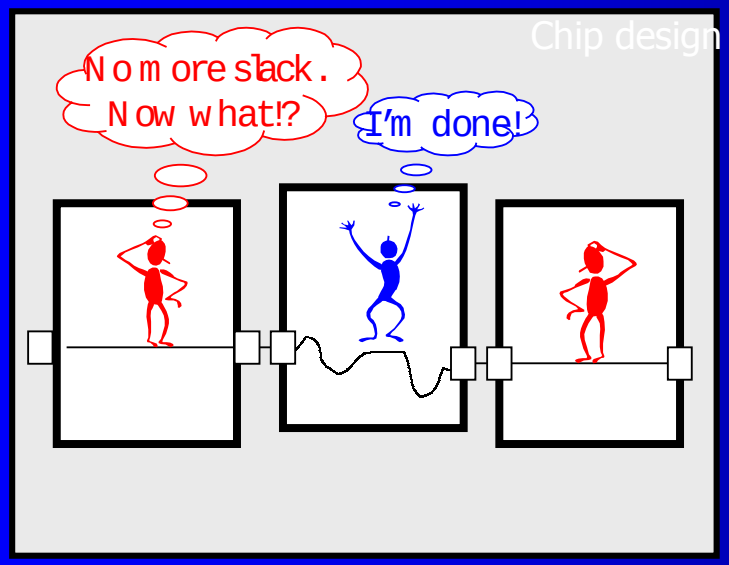


Convergence w Late Change

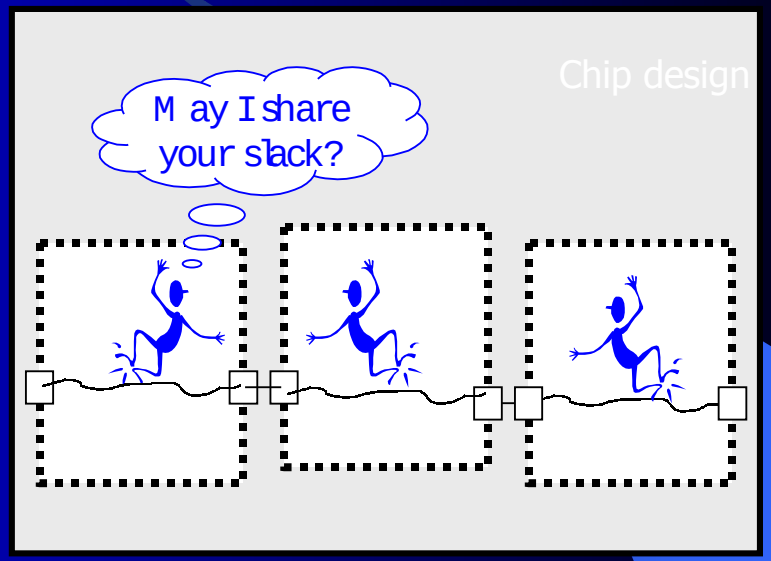


Communication

Traditional



Convergence



- Shared database & data model - fast tool communication
- Incremental design & analysis tools - fast response to changes
- Efficient team structure - fast resolution of conflicts
- Cockpit-oriented methodology - concurrent optimization of multiple metrics

Key Trends

- Abstraction difficult, complicated metrics
  - Deep sub-micron effects
  - Increasingly complex micro-architectures
  - Harder to determine a good starting point
- Communication getting poorer
  - Larger team sizes, geographical separation
  - Attrition over life of project

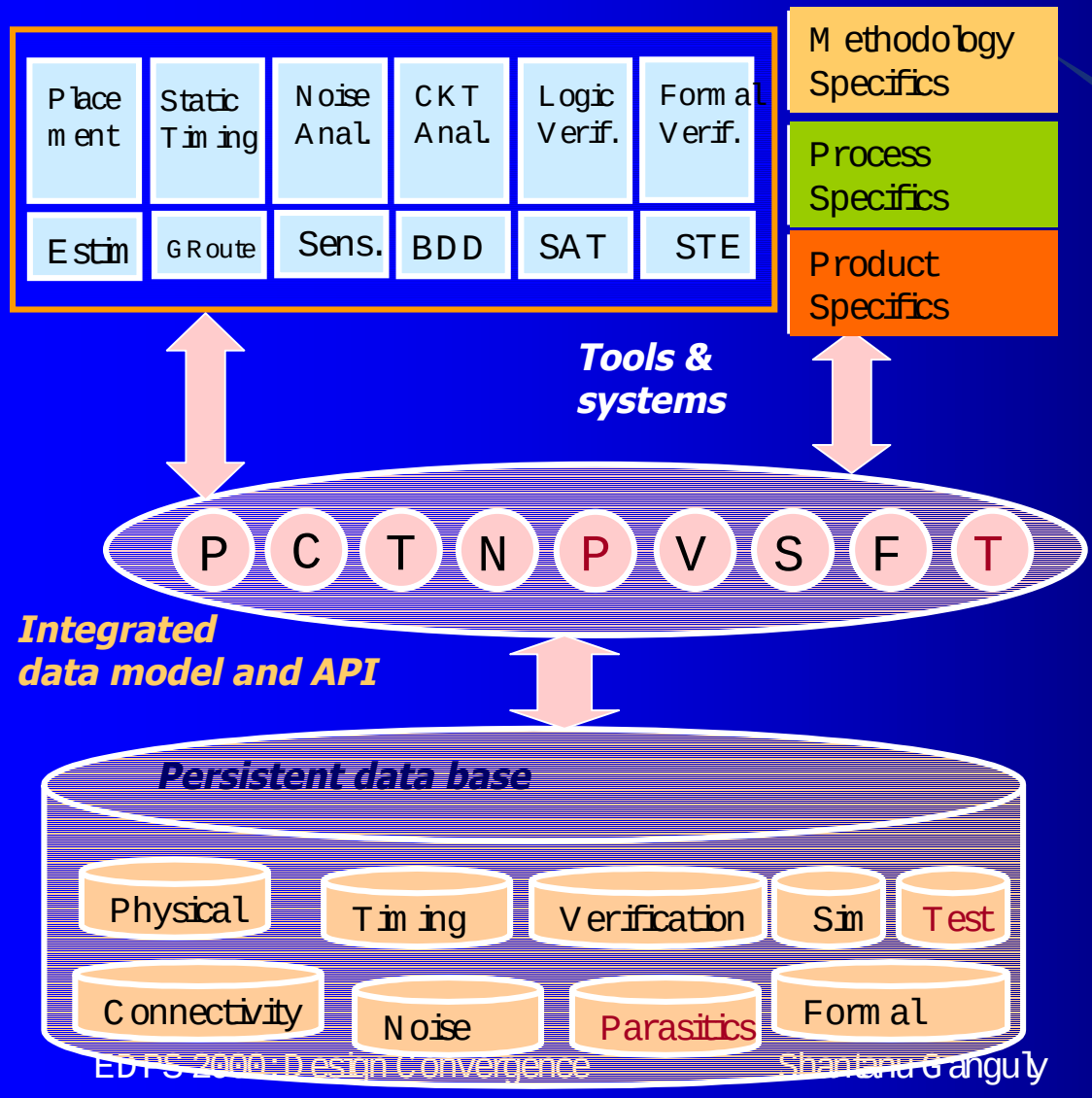
Solution ideas

- Examine, categorize metrics
  - First order metrics: explicitly tracked (area, timing, power, etc)
  - Second order metrics: implicitly tracked by assuming worst case values while measuring first order metrics
- Confirm first, second order metrics at project start:
  - This has implications on design tools
  - Can be mitigated by smart methodology
- EDA architecture to analyze metrics, dependencies

Example of metric complexity

- Second order metrics are becoming first order
  - Example: Worst case power supply droop
- New second order metrics are appearing
  - Assumption of interconnect environment during characterization
  - Locality effects due to process variations
- Interaction between first order metrics
  - Traditionally, decoupled treatment of first order metrics
  - Noise analysis-timing analysis interaction
  - Boolean equivalency: affected by temporal effects

EDA Software Architecture



- Early Data Models drive timing and placement
- Need for parasitics, estimation and global route capabilities
- Noise effects on timing leading to integrated timing and noise flows
- Circuit Analysis tools for handling non-static topologies
- BDD and sensitization capabilities needed for determining functionality and generating models
- Integration of Logic Verification, Formal Verification and Simulation capabilities in future to address dynamic effects in performance analysis

Summary

- Use metrics to drive design convergence
- Identify appropriate metrics at start
- Ensure assumptions do not compromise the goodness of metrics
- Define a good starting point: budgets
- Refine metrics through life of project
  - More iterations typically lead to better solution
  - Use abstractions to speed convergence