Early and Accurate Analysis of SoCs: Oxymoron or Real?

Reinaldo Bergamaschi
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
berga@us.ibm.com

Acknowledgements:

Indira Nair, Nagu Dhanwada, Subhrajit Bhattacharya, Youngsoo Shin, William Dougherty

Contents

- The Need for Early Estimation and Analysis in SoCs
  - How early and how accurate should / can it be?
  - Customer perspective
  - Methodology framework
- SEAS – A system for Early Analysis of SoCs
  - Performance
  - Power
  - Area
  - Timing
- Results
- Recipe for Early Estimation

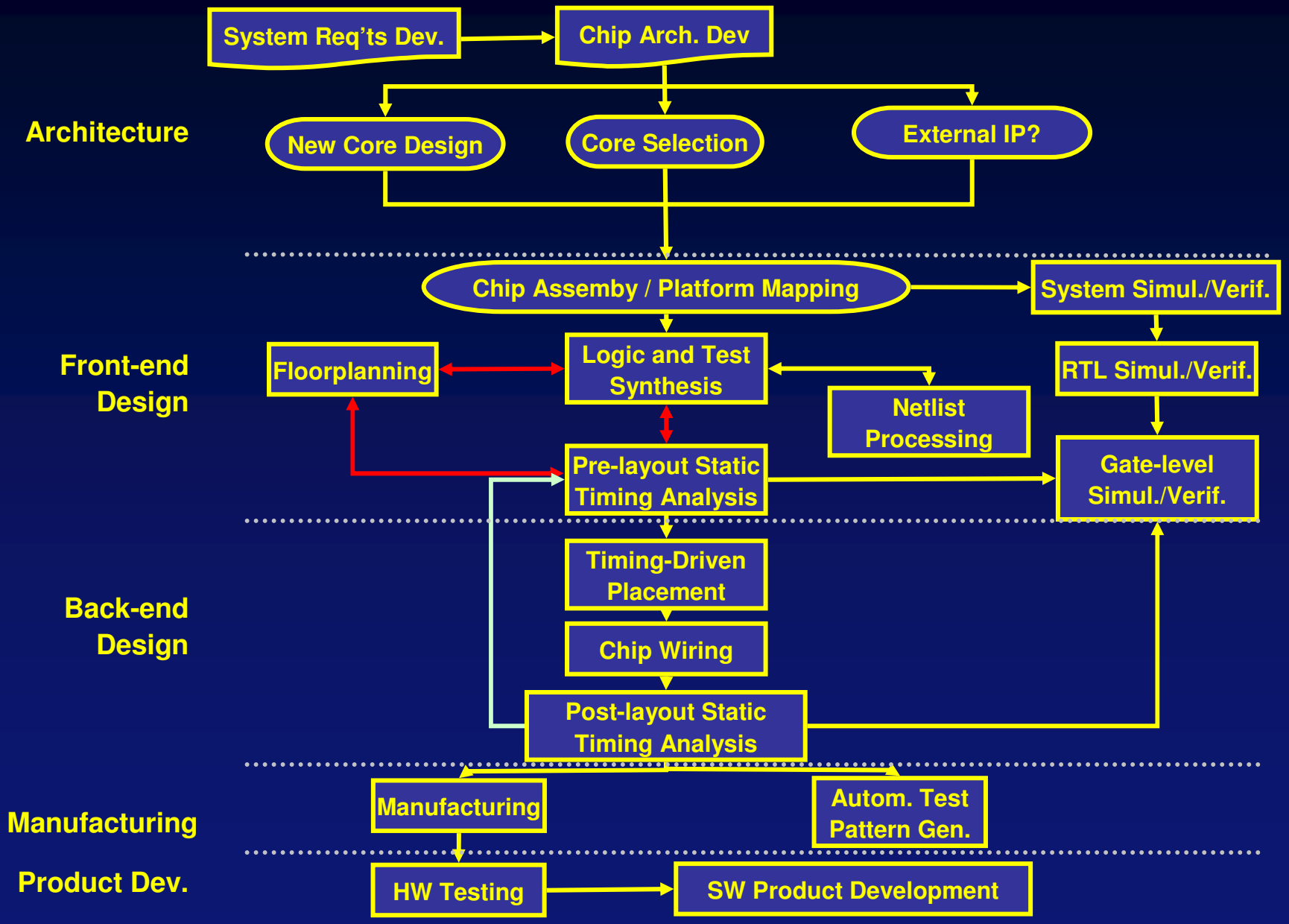
The Need for Early Estimation / Analysis

- Many SoC design problems are directly related to early decisions (which were impossible to fulfill):
  - Pessimistic die size estimation
    - Effect: more costly chip
  - Architectural modifications for performance
    - Effect: more complex final placement, problems for timing closure
  - Inaccurate Floorplanning
    - Effect: timing problems were not detected early, leading to lengthy timing closure iterations

SoC Customer Questions

- Intellectual Property
  - Do you have the functions that I need to build my SoC?
  - IP Libraries, Third-party library support
- How much does it cost to develop it?
  - Development time, methodology, tools
- Will it meet my requirements on performance, power, timing, area, etc.?
  - How can this be estimated early in the design process?
- How much does it cost?
  - Choice of technology, packaging, die size
- How soon can I make it into a product?
  - Product development time (hw + sw)
- Can I get a 2nd manufacturer to make it?
  - Is the process technology supported by other foundries?

SoC Methodology



Reinaldo Bergamaschi

SoC Methodology and Early Estimation

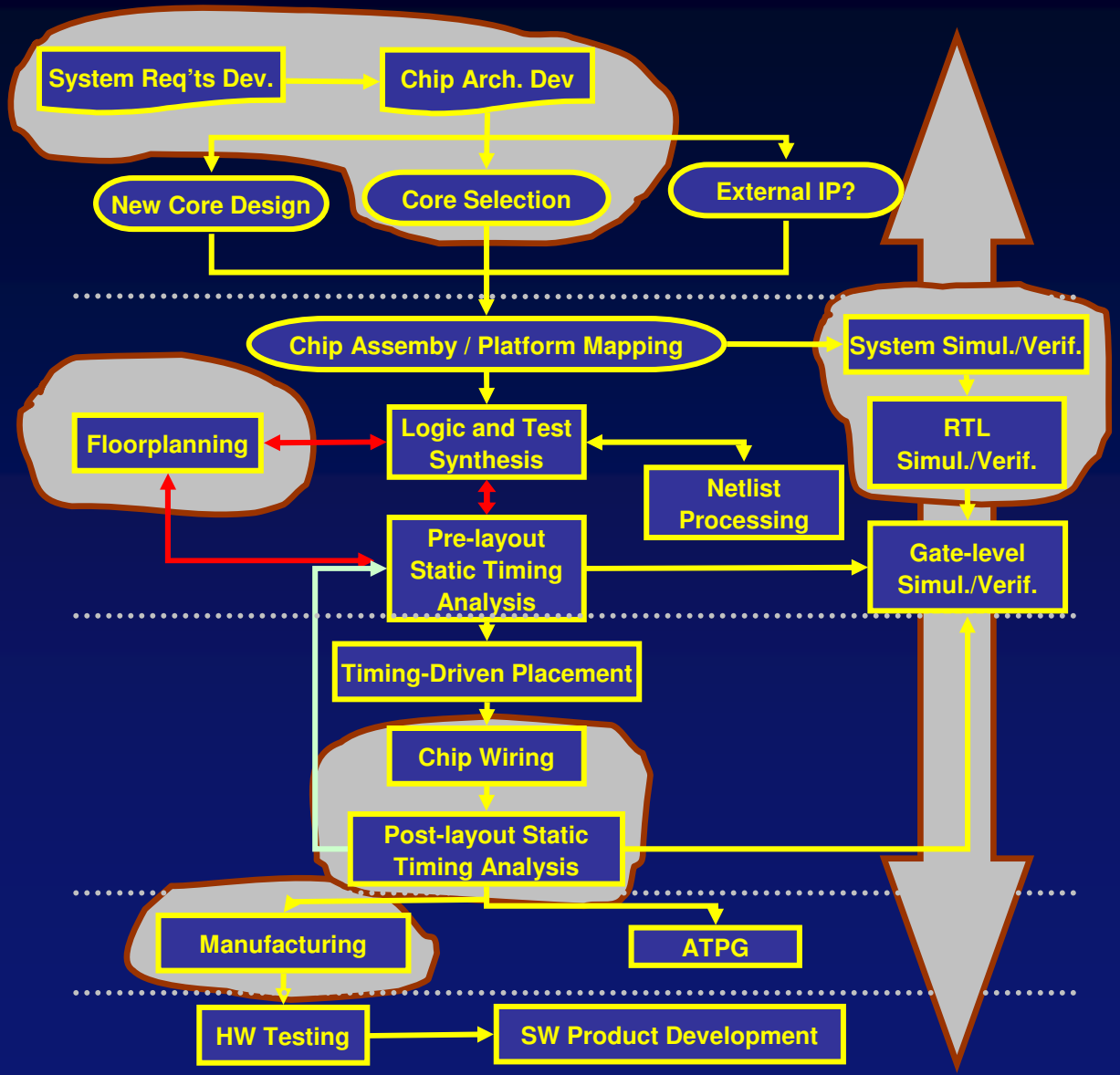
"Do you have the functions that I need?"

"What is the die size"

"Will it meet my requirements on performance and power?

"How much does it cost to fab?"

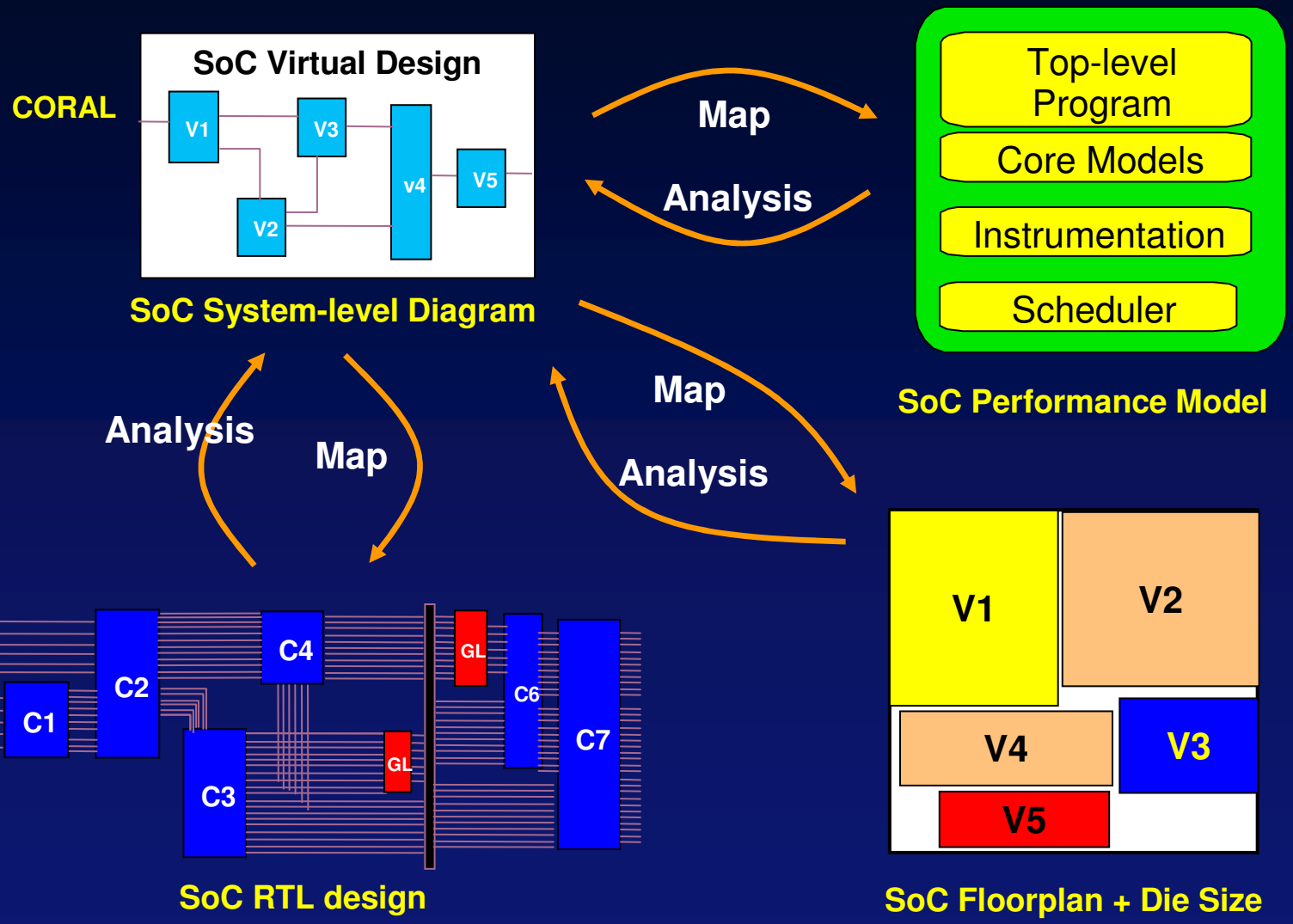
"How much does it cost to develop it?"



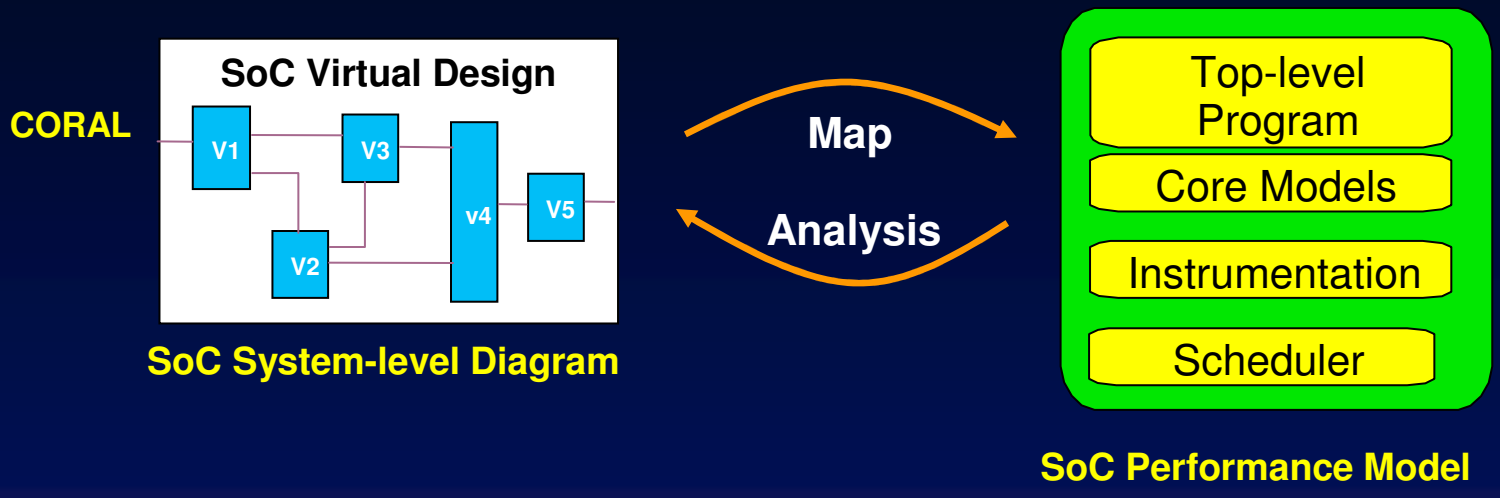
SEAS Motivation and Goals

- An environment to allow early analysis of SoCs
- How early and how quickly?
  - Prior to the existence of any detailed spec (beh, rtl)
  - One day turnaround from creating the design spec to generating analyses results
  - A couple of hours turnaround in evaluating changes
- Accurate enough to allow decisions to be made on:
  - Architecture
  - Components
  - Floorplan (Area + timing)
  - Chip/Die Size and Cost
- Direct links to implementation (e.g., detailed synthesis, P&R)

SEAS Environment



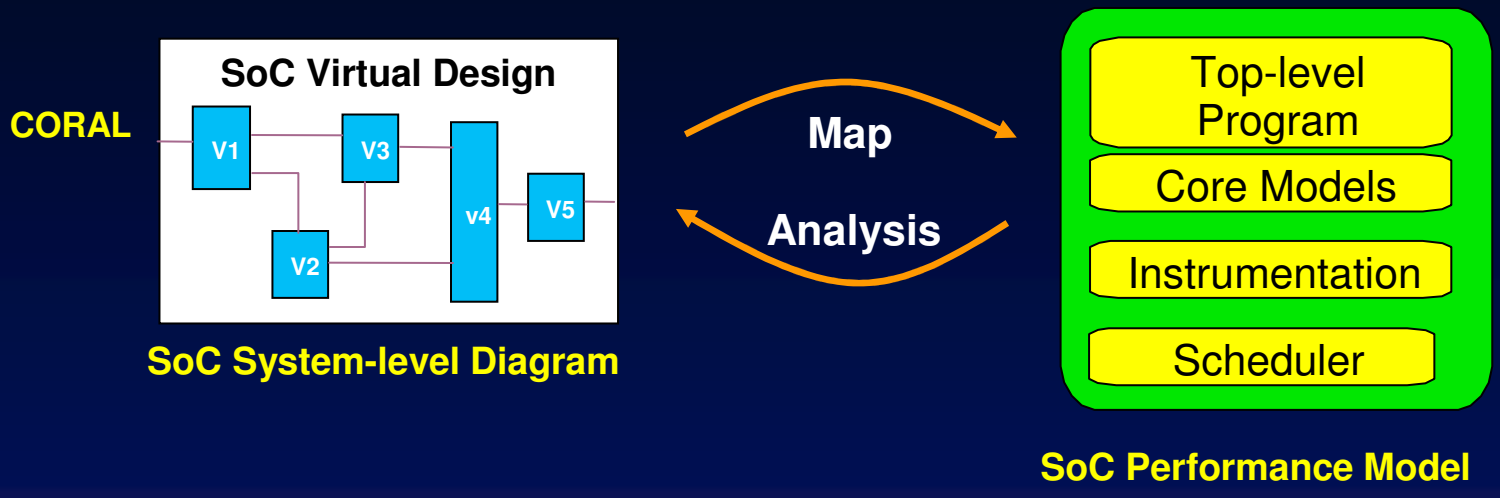
Performance Analysis



- Performance Model of each core
  - Token-based, timed model
  - Non-functional, just performance
  - Parameterizable (buffer sizes, etc.)
  - Software profiled, not executed
  - SystemC

```
sc_main (..) {
    v1 = new emac_pm(...);
    v2 = new fifo_pm(…);
    v3 = new powerpc(…);
    v4 = new bus_arb(…);
    // ….
}
```

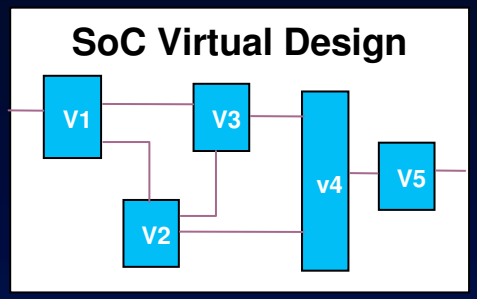

Power Analysis



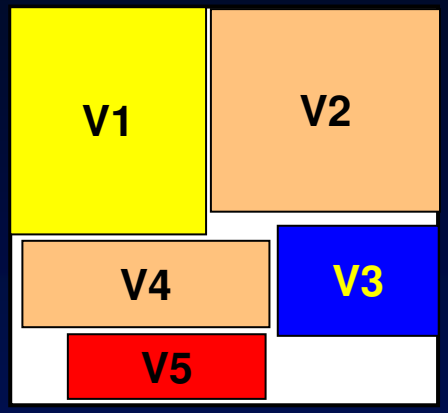
- **Power Models for cores**
  - **State-based (e.g., active, idle, sleep).**
  - **Power values obtained from characterization (at RTL, or spreadsheet)**
  - **Performance simulation determines the state of each core**
  - **Power values for cores are added up during simulation**
  - **Workload-driven power analysis**

Area / Timing / Wiring Analysis

CORAL



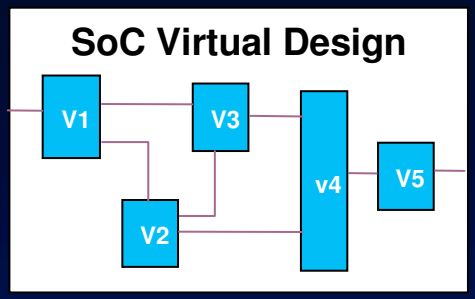
SoC System-level Diagram



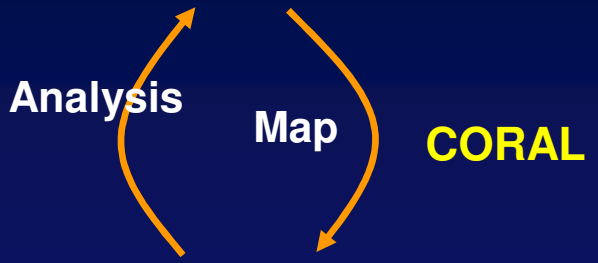
SoC Floorplan

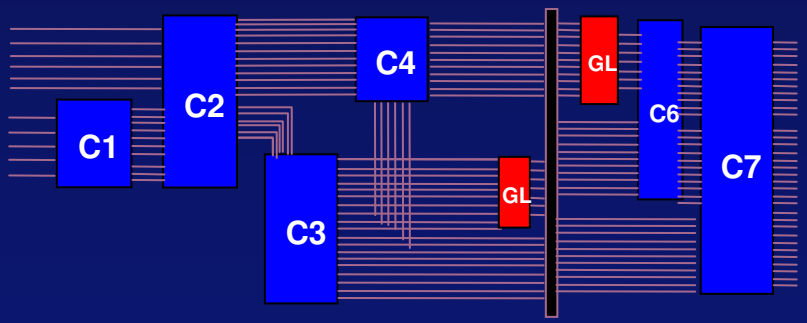
- SoC Floorplan at Virtual Design level
- Full FP capability (hard/soft cores, area planning, pin assignment)
- Global Routing of virtual connections
- Area Analysis
  - Accurate Die size estimation
- Timing Analysis
  - Virtual design has only virtual nets and pins, no drivers, no buffers. Timing is not accurate
- Wire Lengths
  - Good correlation between virtual wire lengths and average real wire lengths. Use wire length to estimate timing criticality.

Mapping to Real Design (RTL)



SoC System-level Diagram

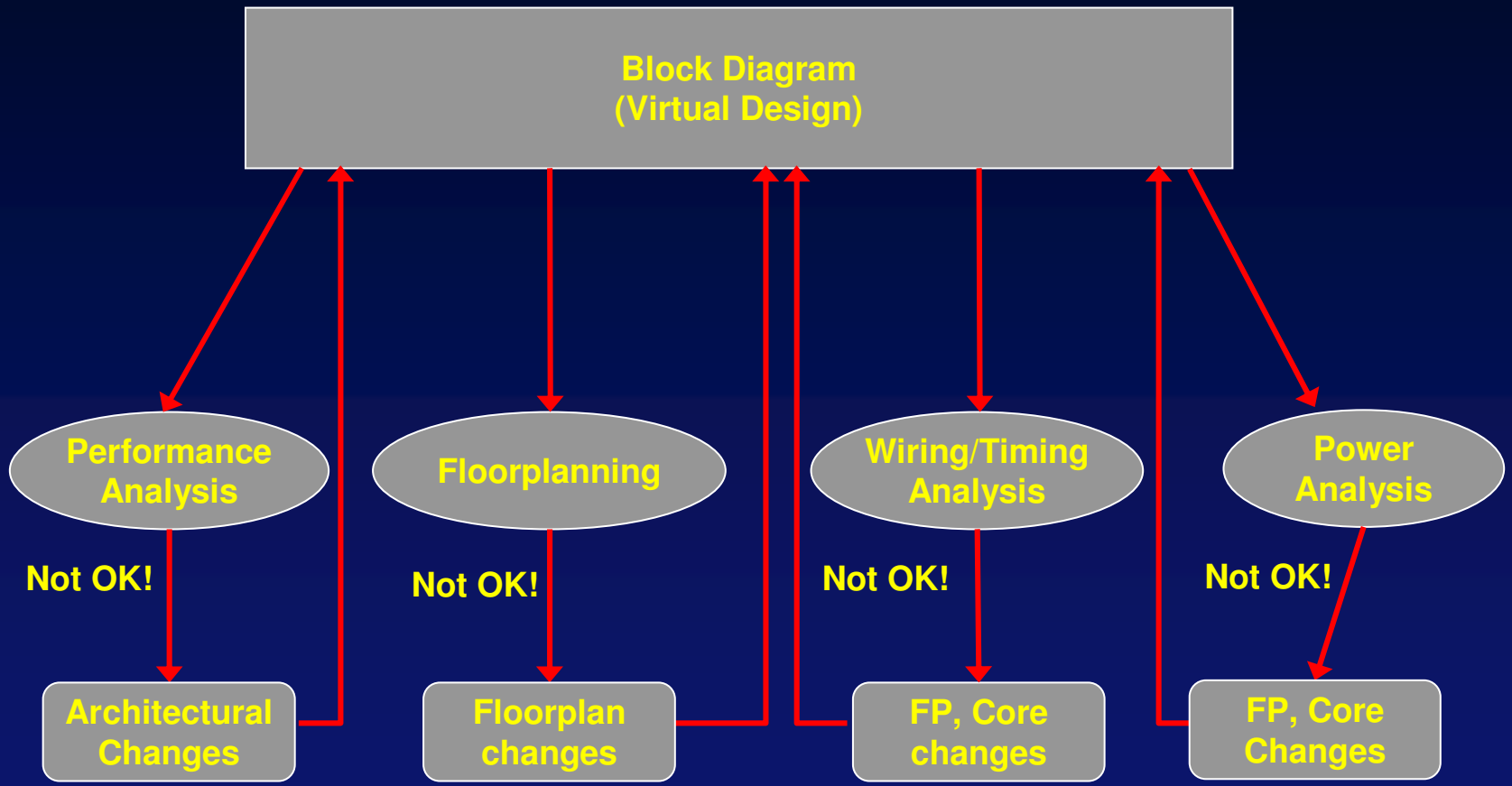




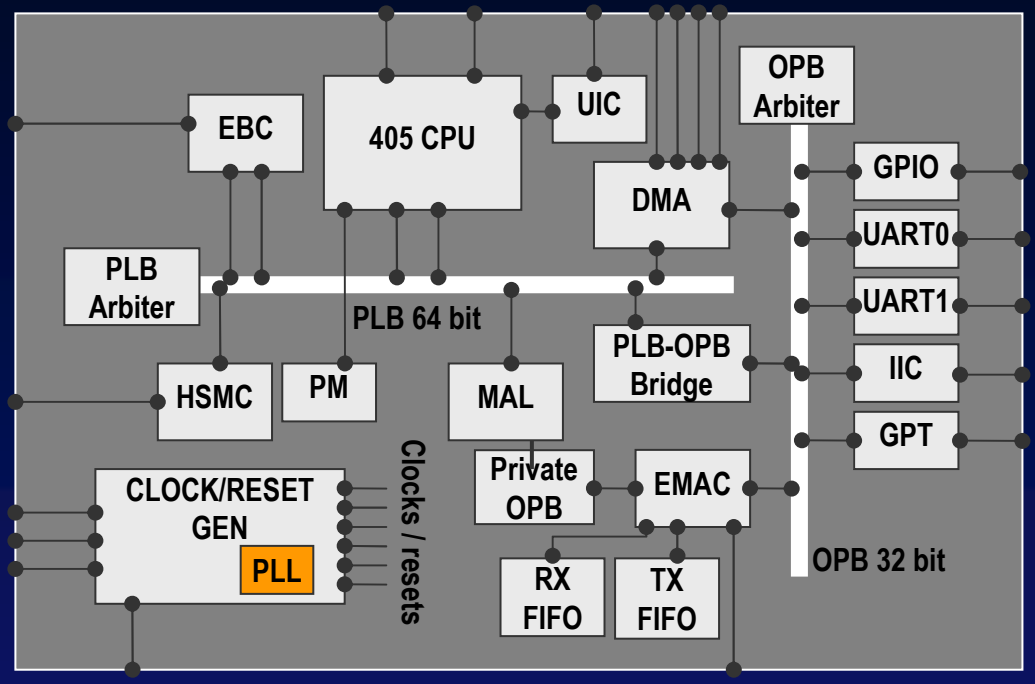
SoC RTL design

- From Virtual Design to RTL
- CORAL
  - Map Virtual components to Real RTL components + glue logic
  - Expand all virtual nets and pins into real nets and real pins
  - Make RTL connections as specified by pin properties
  - Generate RTL VHDL/Verilog
- Links to Implementation
  - Pass Virtual FP initial placements to Real FP initial placements
  - Complete FP of RTL / Real Design
  - Die Size
  - Timing analysis

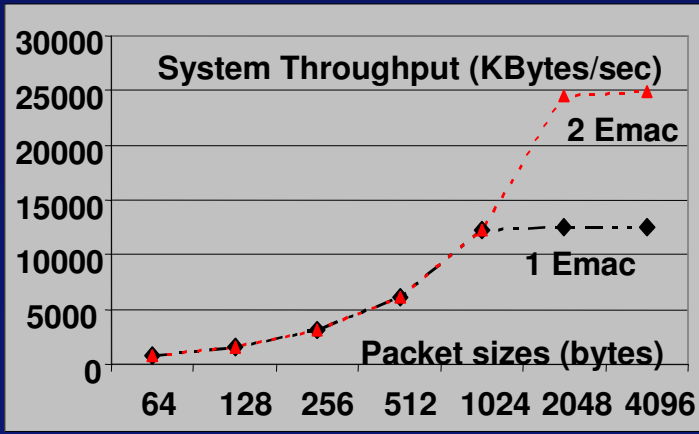
SEAS Environment

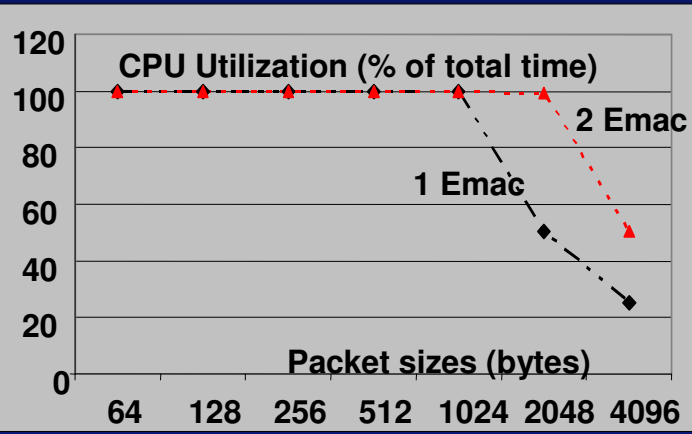


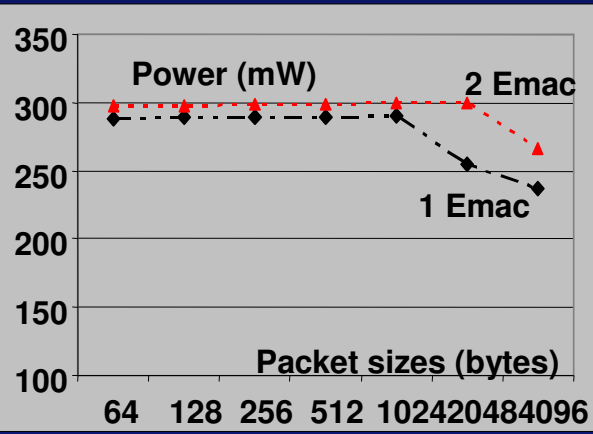
SEAS Experiment



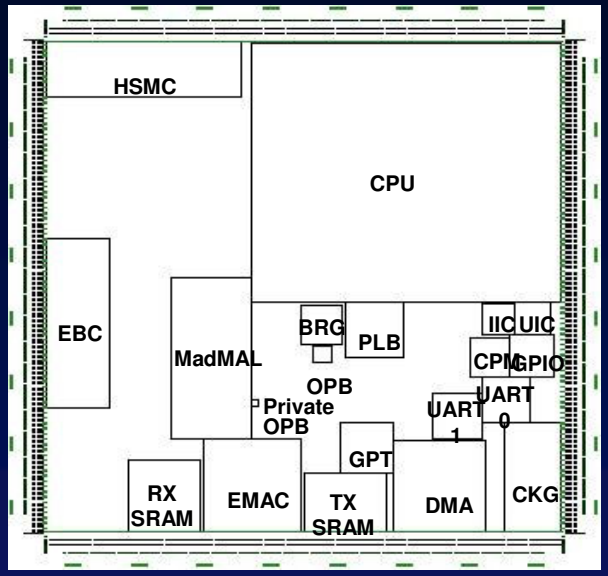
- 405PBD
  - Ethernet Subsystem
    - 1 EMAC
    - 1 Madmal
- Change to improve performance
  - Added an extra EMAC + Fifos
- Measure effects on die-size, fp, timing, power



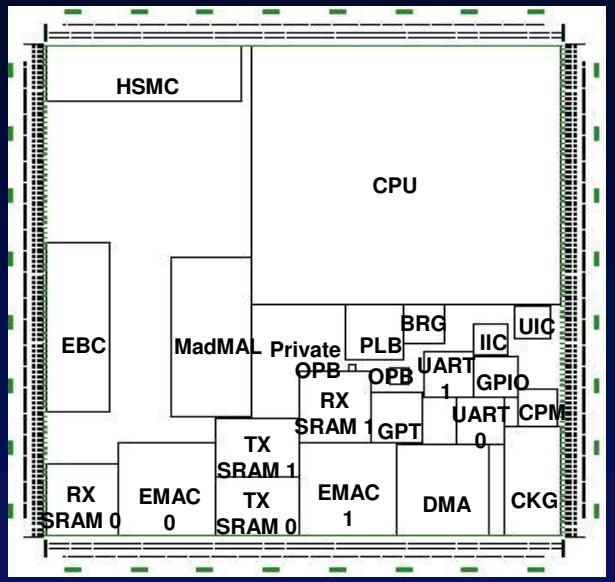




SEAS Experiment



1 EMAC
6.05mmx6.05mm



2 EMACs
6.05mmx6.05mm

Real Design Layout (1 EMAC)
6.05mmx6.05mm

- **Results**
  - **Two Emac solution delivered the required performance**
  - **Could fit in the same die-size as the original one**
  - **Met the same timing requirements as the original one**

SEAS Conclusions

- Demonstrated that early analysis can be “accurate enough” and valuable to the design flow
- Most Important Aspects
  - Abstract Input Specification. Easy to write and change. Minimum complexity
  - Estimation algorithms that understand this abstract spec (through IP models and characterization up front)
  - Links to implementation
- Towards the ultimate sign-off level: “Block Diagram Sign-off”

Recipe for Early and Accurate Estimation

- Estimate at the "Right Level of Abstraction" for the desired accuracy
  - Top-down constructive method (e.g., SEAS) potentially more accurate than bottom-up abstraction method
- Use the same data model throughout the different representations and tools
  - Increases consistency among different analyses (timing, layout, rtl)
  - Prevents loss of data
  - Speeds up tool development time
- Early estimation and analysis requires:
  - Simple, early specification, easy to change
  - Libraries of models at the various levels being estimated
  - Keep one single golden spec (e.g., Virtual Design) and mapping algorithms between the spec and the various levels being estimated.

