SYNOPSYS®

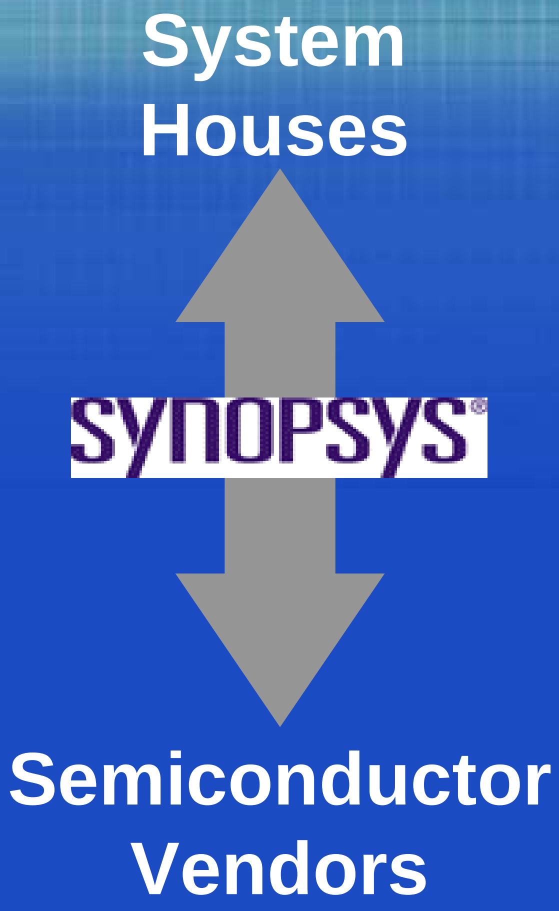
CoCentric SystemC Compiler

Eric C. Smith

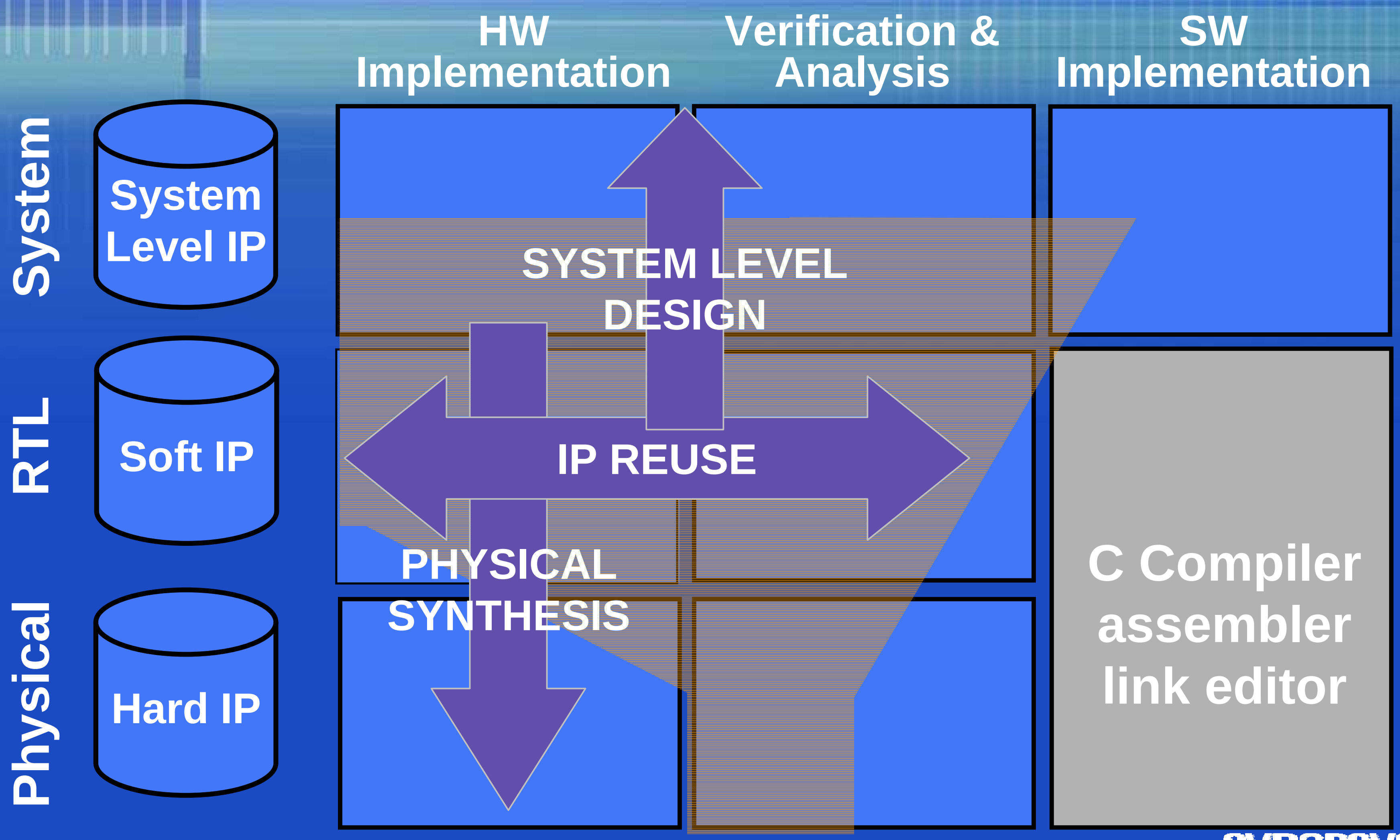
Product Marketing Manager

Synopsys, Inc.

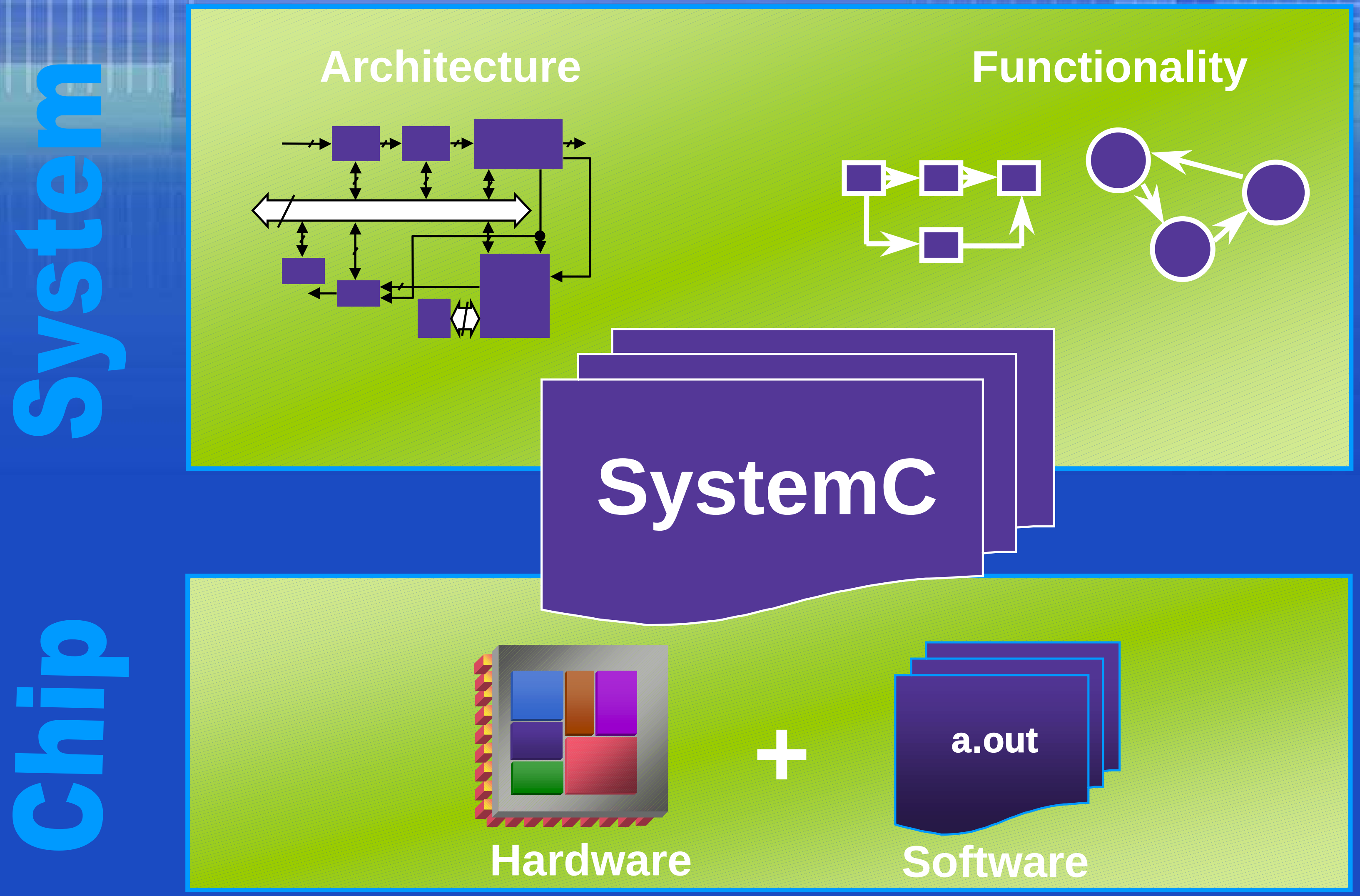
Bringing System Houses and Semiconductor Vendors Together



Three Strategic Thrusts for SoC



SystemC
Common Modeling Platform for SoC



SYNOPSYS

CoCentric™: New Generation of System Level Design Solutions

- System Studio
  - C based co-design and co-verification environment
- Fixed Point Designer
  - Converts floating-point to fixed-point
- SystemC Compiler
  - Synthesizes hardware from SystemC
- Ref. Design Kits, Libraries, Workbenches
  - Standards-compliant to jump-start designs

SYNOPSYS®

SoC Hardware Design Methodology …Yesterday

C/C++
HDL
System Designer
Hardware Designer

4. Hand over specification document...

1. Conceptualize
2. Simulate in C/C++
3. Write specification document

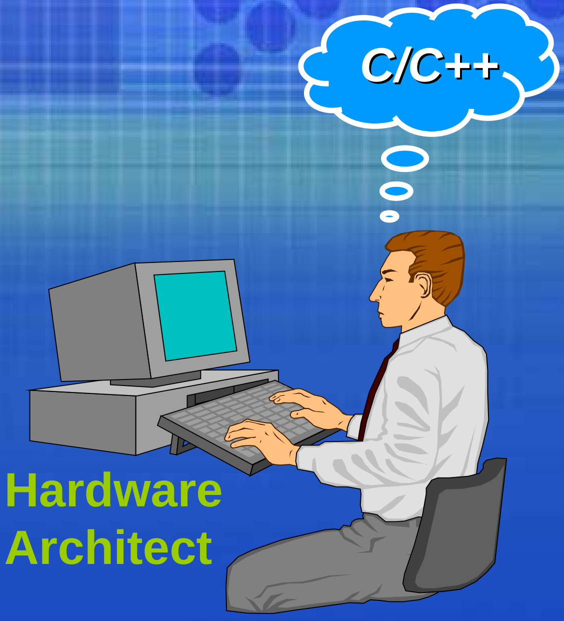
5. Understand/Refine for HW
6. (Re)Implement in HDL
7. (Re)Verify
8. Synthesize from HDL

Creating HDL model is time consuming and error prone

SYNOPSYS®

SoC Hardware Design Methodology …Today

4. *Hand over…*
   **Functional Spec*
   **Testbenches*



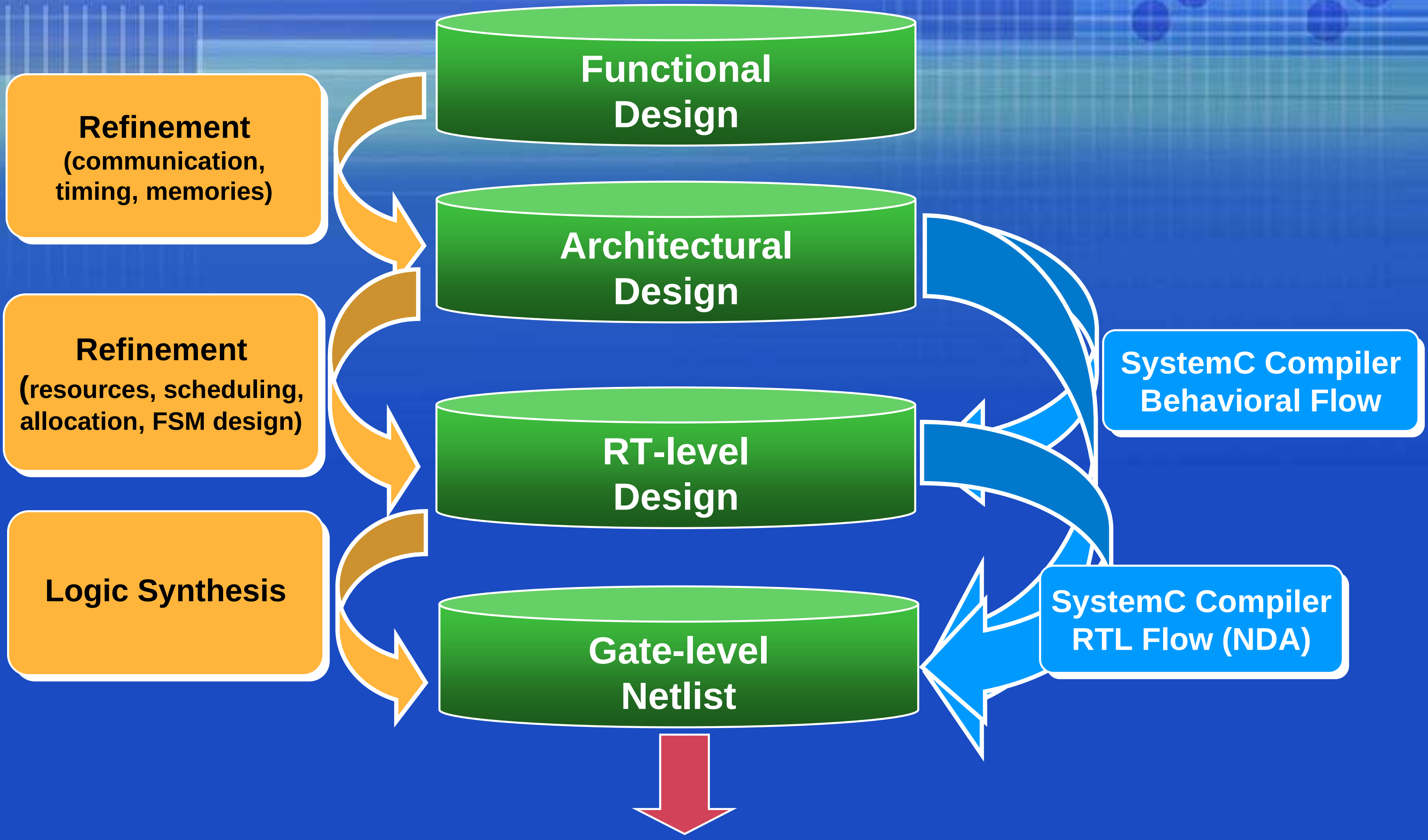
1. *Conceptualize*
2. *Simulate in C/C++*
3. *Write specification document (if needed)*

5. *Understand in SystemC*
6. *Refine to Implementable Model*
7. *Verify using original testbench*
8. *Synthesize into HDL/gates*

SystemC Compiler provides the high productivity path to hardware!

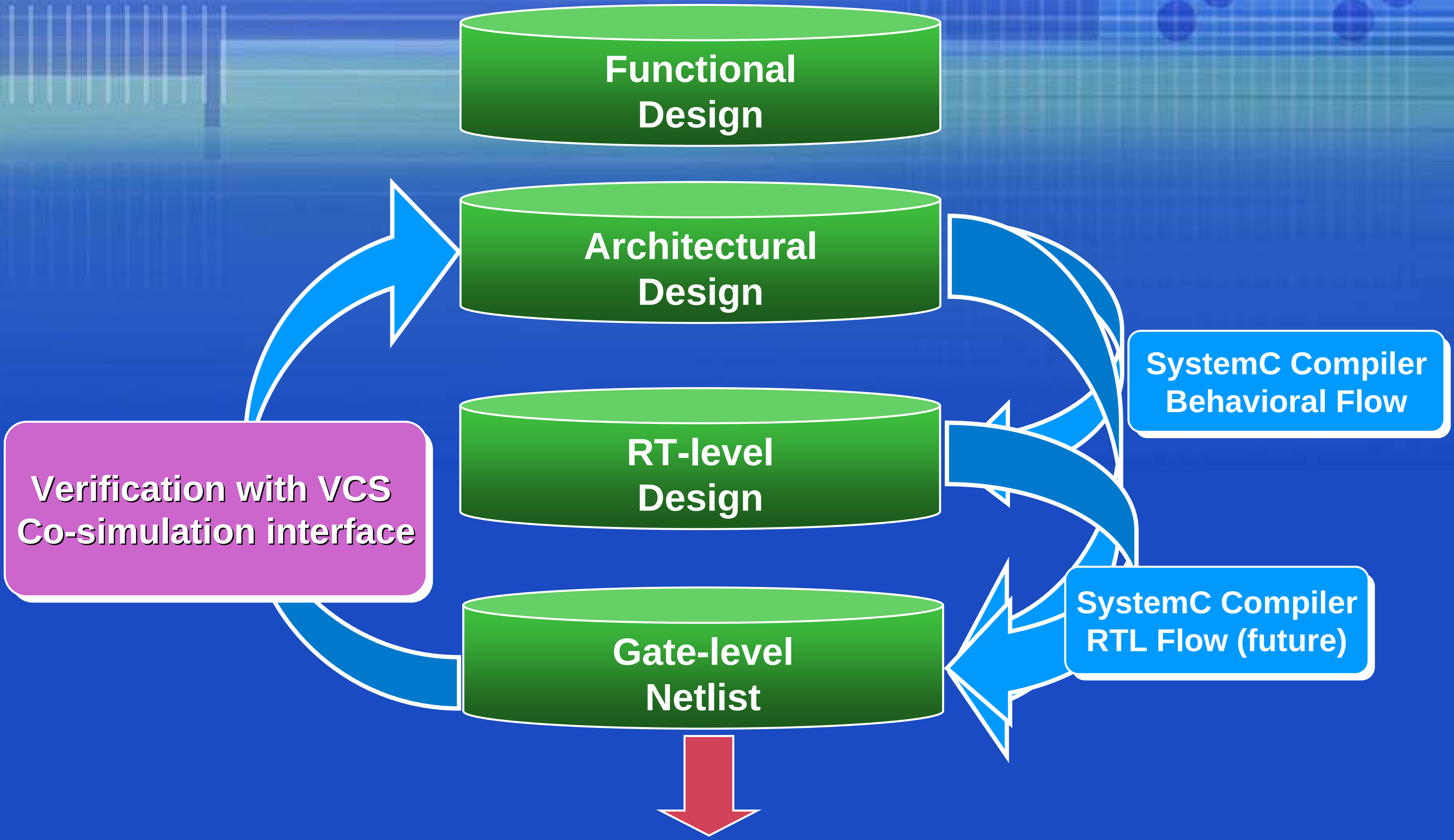
SYNOPSYS

SystemC Compiler in the SoC Flow


Functional Design
Architectural Design
RT-level Design
Gate-level Netlist
Refinement (communication, timing, memories)
Refinement (resources, scheduling, allocation, FSM design)
Logic Synthesis
SystemC Compiler Behavioral Flow
SystemC Compiler RTL Flow (NDA)

SYNOPSYS

… and Verification Flow


Functional Design
Architectural Design
RT-level Design
Gate-level Netlist
Verification with VCS Co-simulation interface
SystemC Compiler Behavioral Flow
SystemC Compiler RTL Flow (future)

SYNOPSYS

SystemC Synthesis Benefits

- Rapid time to market
- Graphical design analysis
- High quality of results

SYNOPSYS®

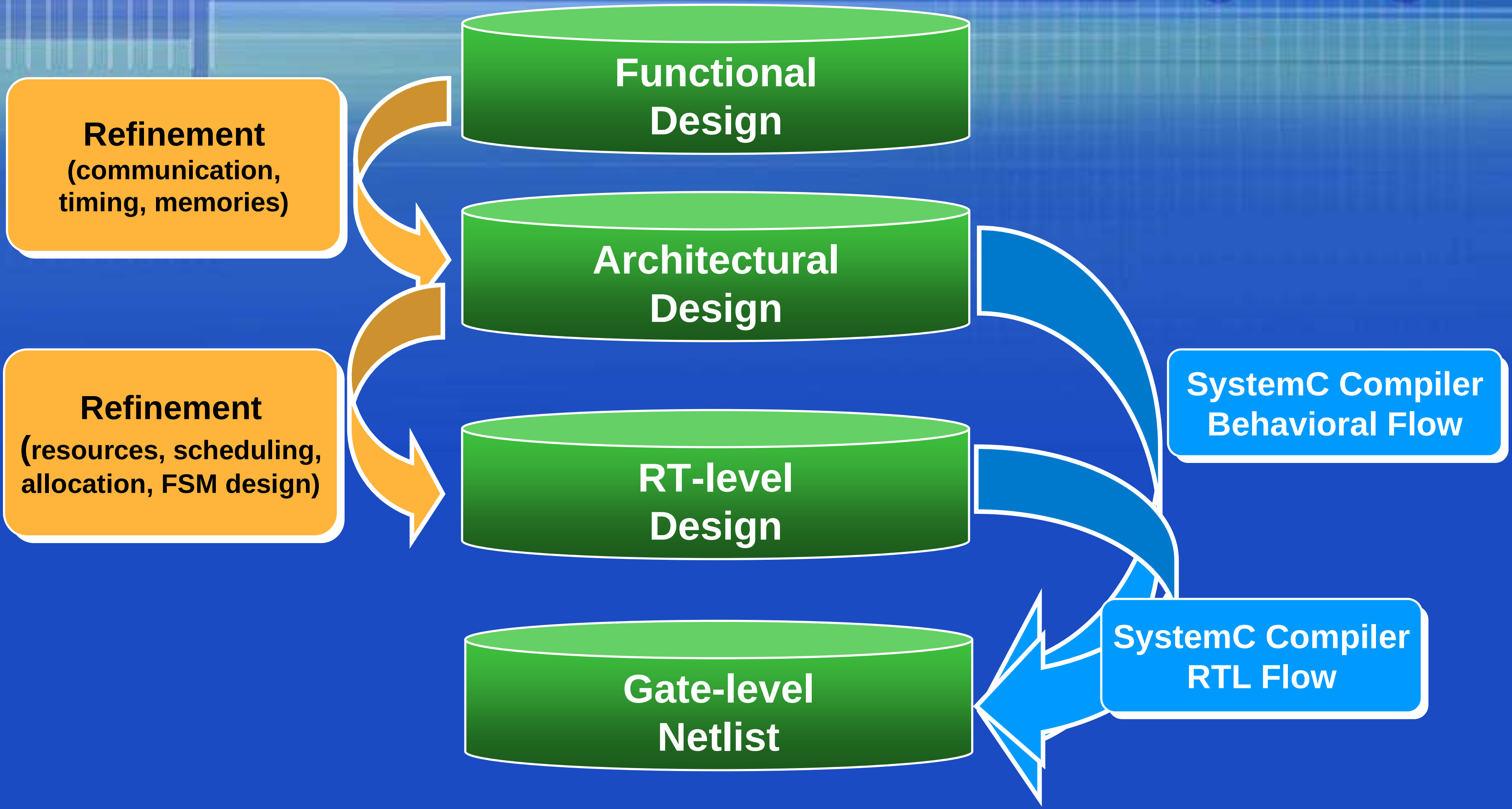
CoCentric SystemC Compiler NDA Suite

Rocco Jonack

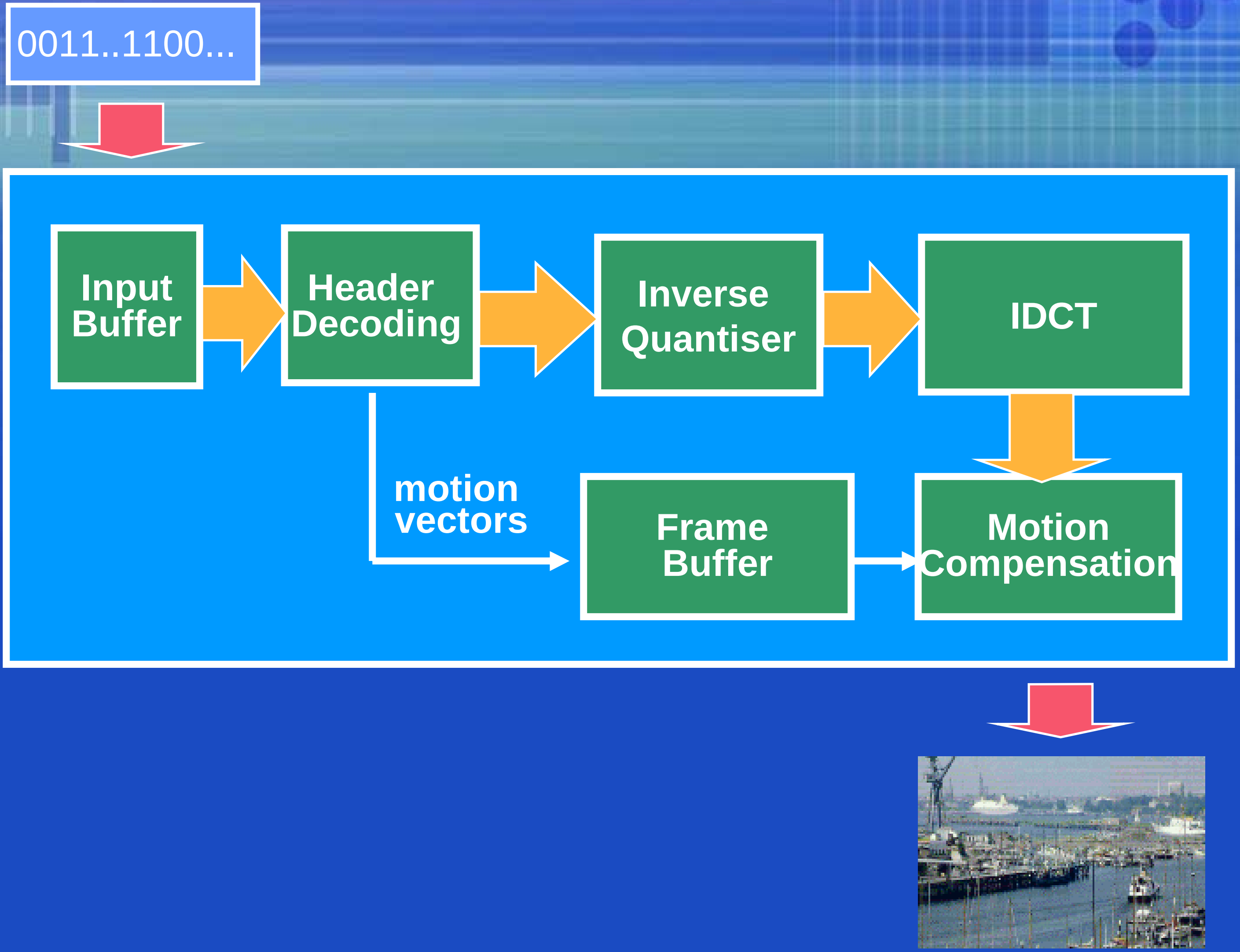
Corporate Applications Engineer

Synopsys, Inc.

CoCentric SystemC Compiler Hardware Implementation Flow

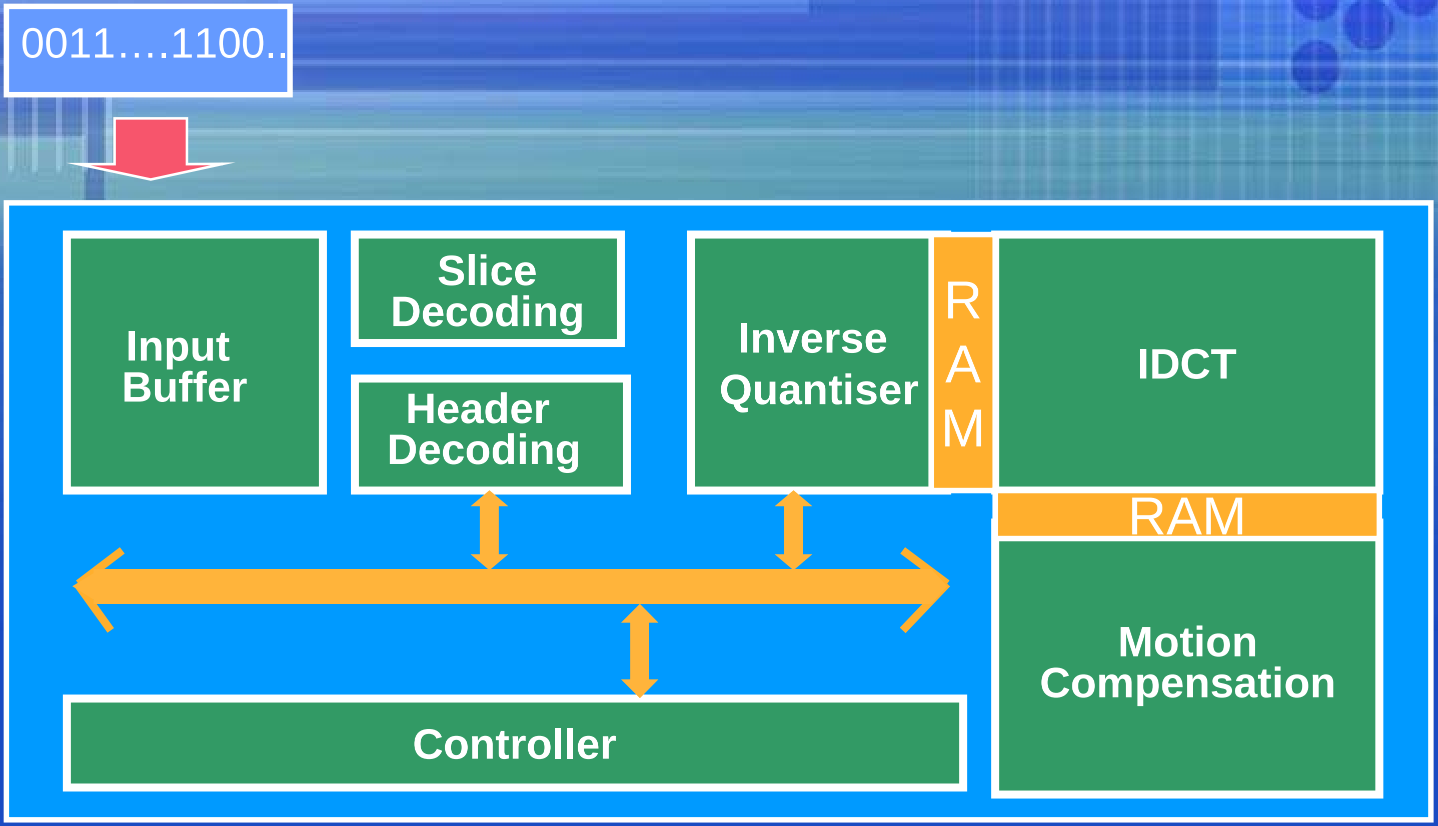

Refinement
(communication, timing, memories)
Functional Design
Architectural Design
Refinement
(resources, scheduling, allocation, FSM design)
RT-level Design
Gate-level Netlist
SystemC Compiler Behavioral Flow
SystemC Compiler RTL Flow

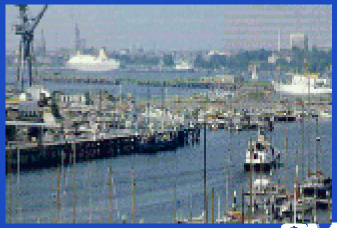
Video Decoder - Functional View



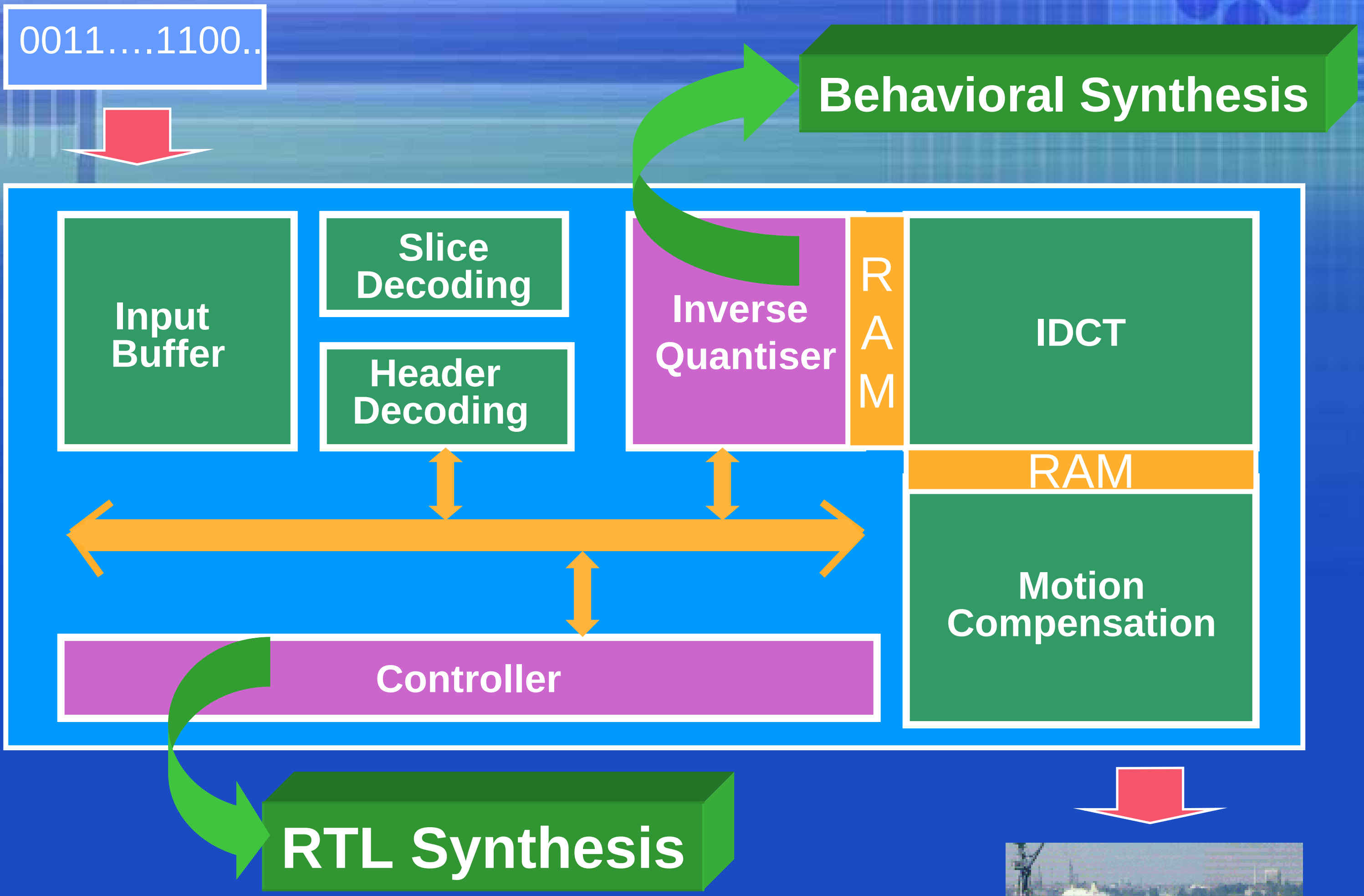
SYNOPSYS®

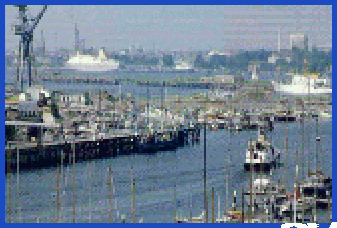
Video Decoder - Architectural View





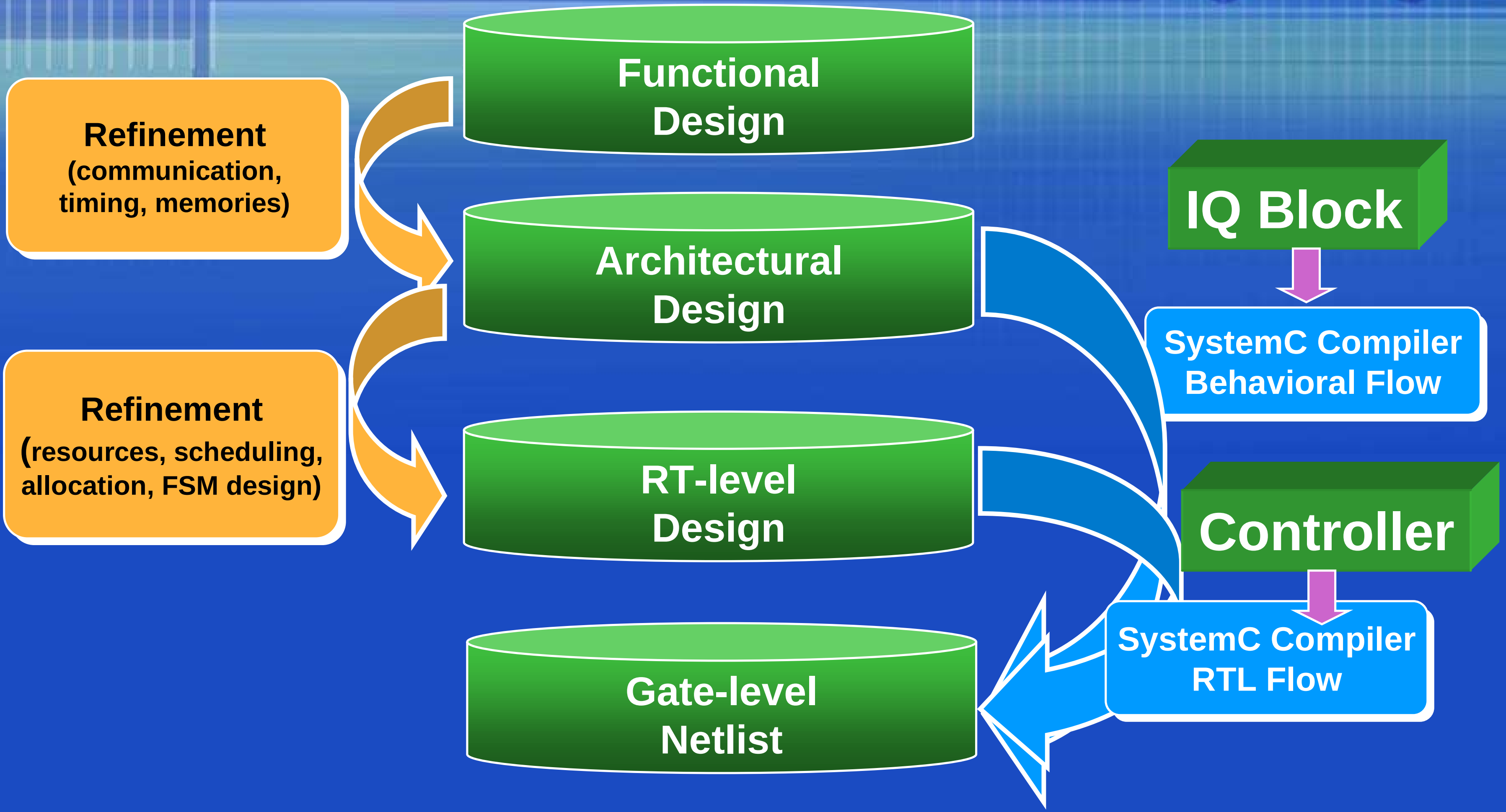
Video Decoder - Inplementation



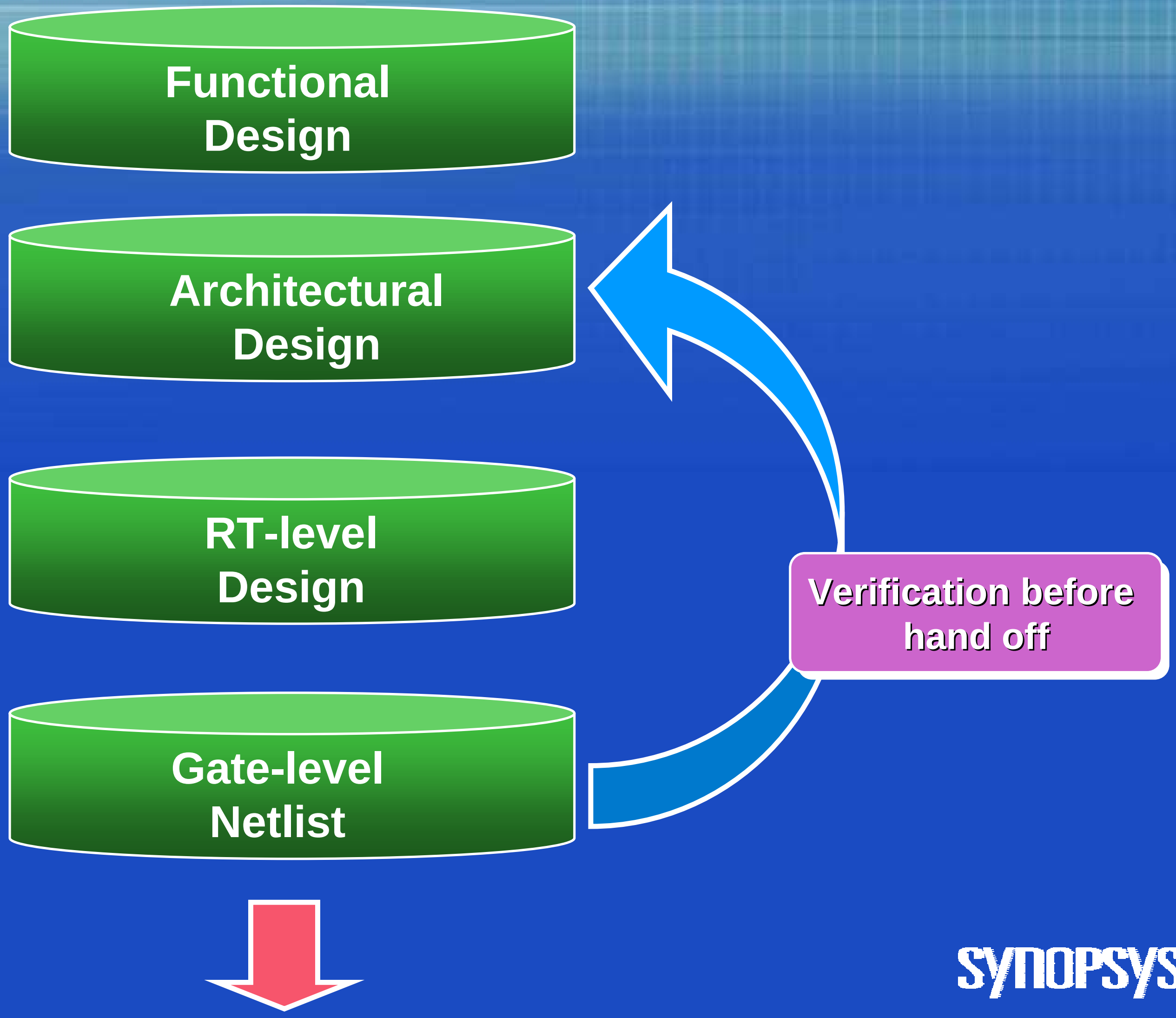


SYNOPSYS

CoCentric SystemC Compiler Hardware Implementation Flow


Refinement
(communication, timing, memories)
Refinement
(resources, scheduling, allocation, FSM design)
Functional Design
Architectural Design
RT-level Design
Gate-level Netlist
IQ Block
SystemC Compiler Behavioral Flow
Controller
SystemC Compiler RTL Flow

CoCentric SystemC Compiler Hardware Verification Flow



SYNOPSYS

SystemC Compiler Benefits

- Rapid time to market
  - fast refinement from functional model to behavioral model
  - accommodating late spec changes
- Easy Analysis of design
- High quality of results
  - tight integration into Synopsys synthesis flow
  - flexibility for datapath components

Summary

- SystemC/SystemC Compiler bridge the gap between System Level Architect and the HW designer
- SystemC Compiler is the higher productivity path to ASIC/FPGA implementation
- SystemC Compiler leverages existing Synopsys hardware design methodologies and tools

SYNOPSYS®

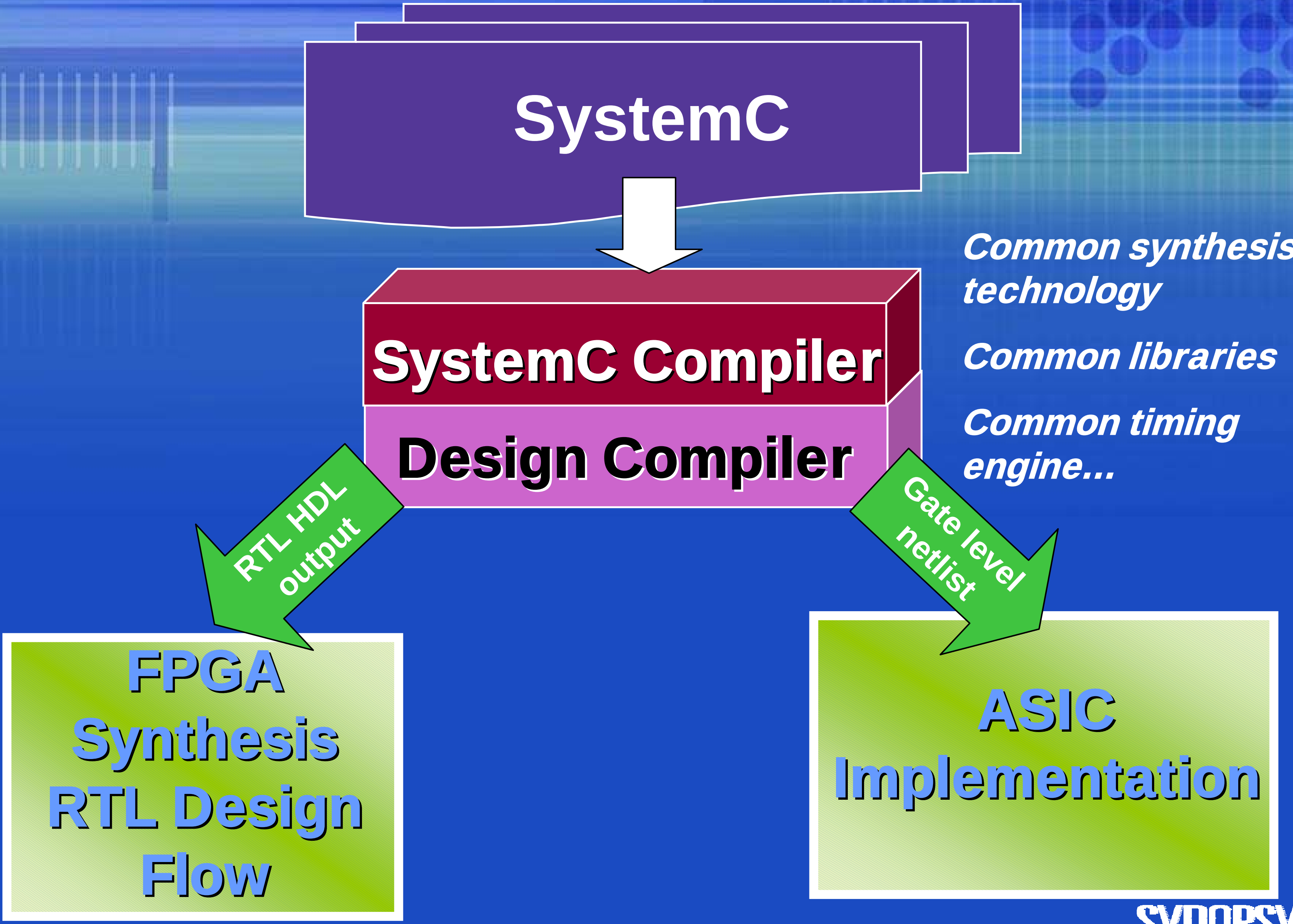
Extra Slides

SystemC: Natural Extension into HW Modeling

- Create C++ class libraries to provide
  - Concurrency - processes
  - Communication - signals and channels
  - Hardware data types
    - bit vectors
    - arbitrary precision signed and unsigned integers
    - fixed point numbers
  - Reactive behavior - waiting and watching
  - Notion of time - clocks

SYNOPSYS

SystemC Compiler Applications



Common synthesis technology

Common libraries

Common timing engine...

Behavioral Synthesis with SystemC Compiler

- Fast refinement from functional model to behavioral model
  - description close to specification
  - describing complex functional units
- high quality of behavioral synthesis results
  - tight integration into Synopsys synthesis flow(using common timing engine)
  - providing flexibility for datapath generation(precompiled netlists, pipelined operators)
  - convenient interfacing to memory models
- Specification Changes
  - flexibility in terms of latency can be used to optimize architecture

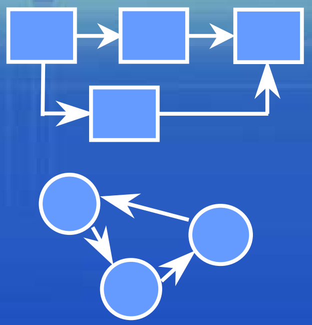
RTL synthesis

- RTL synthesis is main stream methodology for ASIC design
  - coding style is well defined
  - no changes for implementation flow necessary
- apply when register-to-register architecture is already defined
  - sometimes functionality is easiest to describe as RTL
  - behavior is a pure FSM

SYNOPSYS®

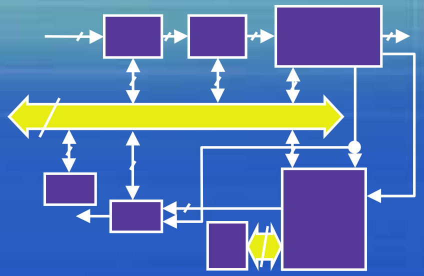
The C++ Movement: the Language of System Design

Algorithms



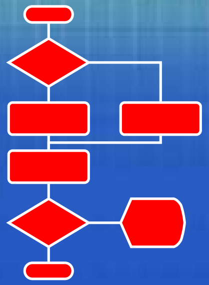
• Using C/C++

Architecture



• Proprietary Languages
• Proprietary C Extensions
• C++ Class Libraries

Application



• Using C/C++

- **Multiple language solutions exist**
- **Need common dialect to unify the industry**
  - **to model/exchange system level IP**
  - **to build interoperable tools**

SYNOPSYS®