Projektiranje programabilnih SoC platformi
ID 222683

SYSTEMC TLM IN RTL DESIGN FLOW

ELECTRONIC SYSTEM LEVEL DESIGN

- Raises the level of design abstraction

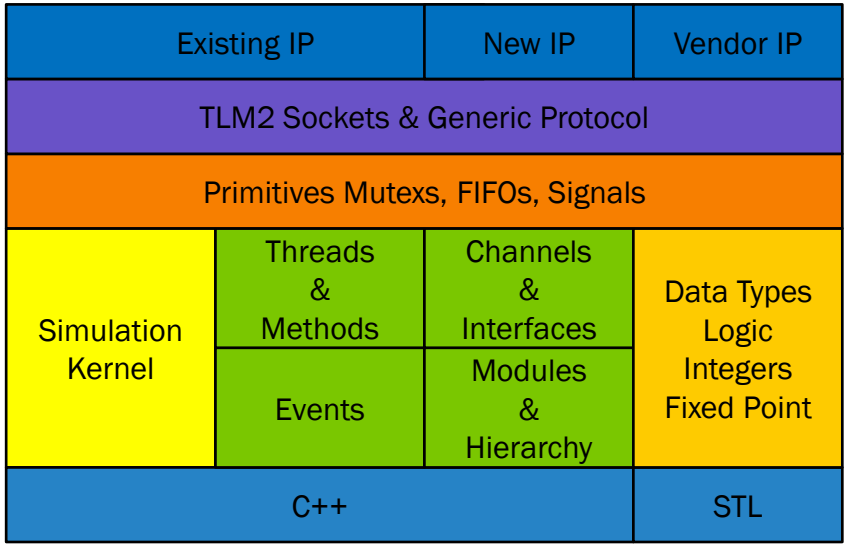
- Early performance model validates architecture concepts
  - Allows engineers to create, change, and validate concepts without implementing RTL
- Supports hardware/software co-design
- Virtual Prototype for early software bring-up

- ESL is supported by a collection of **Accellera standards**, tools, and concepts
  - SystemC & TLM
  - SystemRDL (Register Description Language)
  - Control and Status Register code generators
  - SystemC & RTL co-simulation
  - RTL to SystemC conversions

Adopted from Ed Nuckolls, Adding SystemC TLM to a RTL Design Flow, DAC

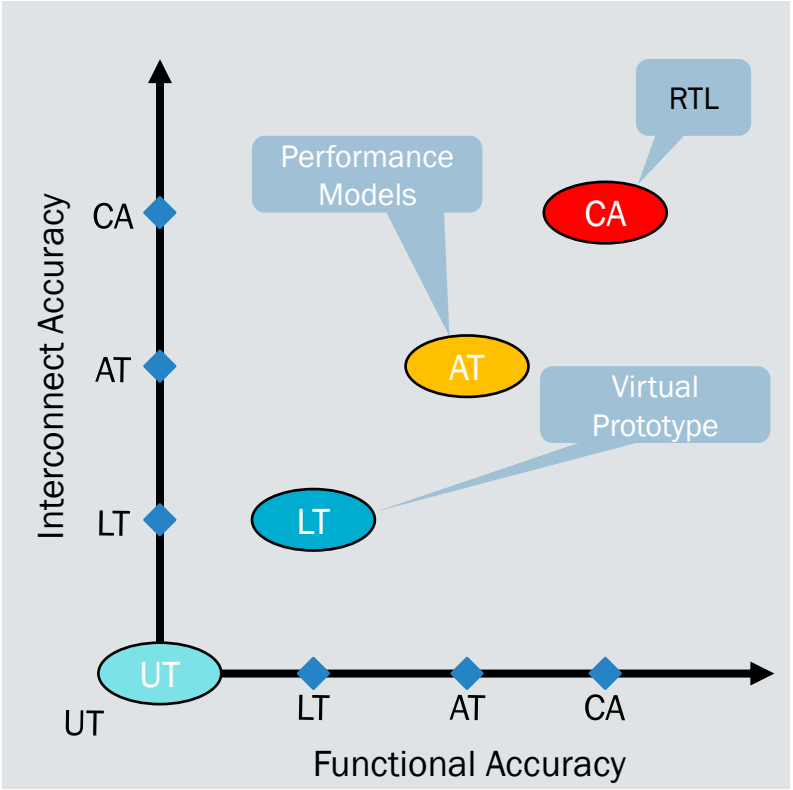
SYSTEMC CHARACTERISTICS

- IEEE Standard 1666-2011 SystemC
  - SystemC (accellera.org) 2022: SystemC 2.3.4
- SystemC
  - allows higher level of abstraction than RTL
  - allows much faster simulation
- SystemC extends C++ with classes, macros, and adds a scheduler, providing
  - Hierarchy and structure familiar to hardware designers
  - Block-to-Block communications
  - Hardware data types
  - Event driven scheduler that allows concurrent execution
- SystemC and RTL co-simulation is supported



TRANSACTION-LEVEL MODELING (TLM)

- TLM is a high-level approach to modeling digital systems that separates:
  - Details of interconnect (communication)
  - Details of functionality (computation)
- OSCI SystemC TLM-2 available June 2008
- IEEE 1666–2011 includes TLM-2:
  - Sockets for block-to-block interconnect
  - Generic protocol (memory mapped bus)
- SystemC TLM defines two modeling styles
  - Loosely-timed (LT)
    - Sufficient timing detail for Virtual Prototypes
    - Temporal decoupling (run ahead)
    - Uses direct memory interface (DMI)
  - Approximately timed (AT)
    - Sufficient timing detail for architecture exploration
    - Processes run in lock-step with simulation time

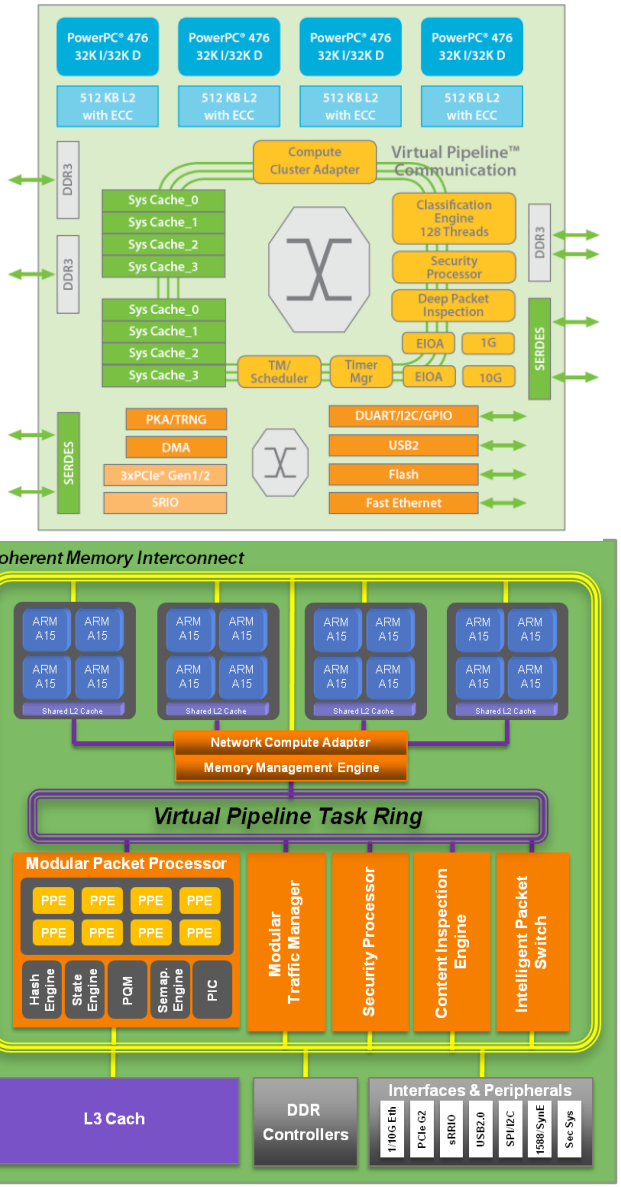


UT -- Untimed
LT -- Loosely timed
AT -- Approximately timed
CA -- Clock & Pin Accurate

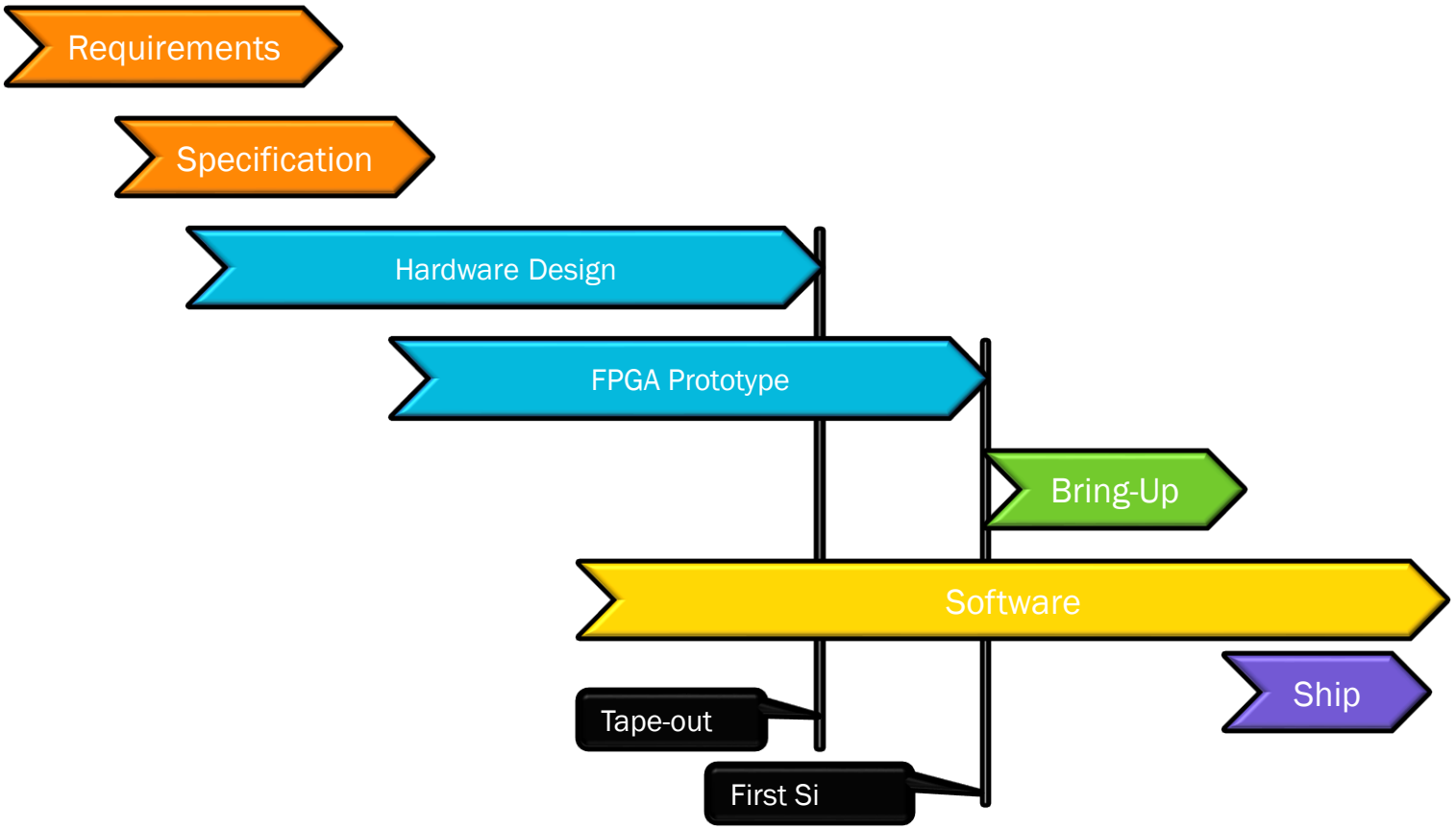
(D Gajski 2003)

RTL AND SOFTWARE DESIGN FLOWS EXIST

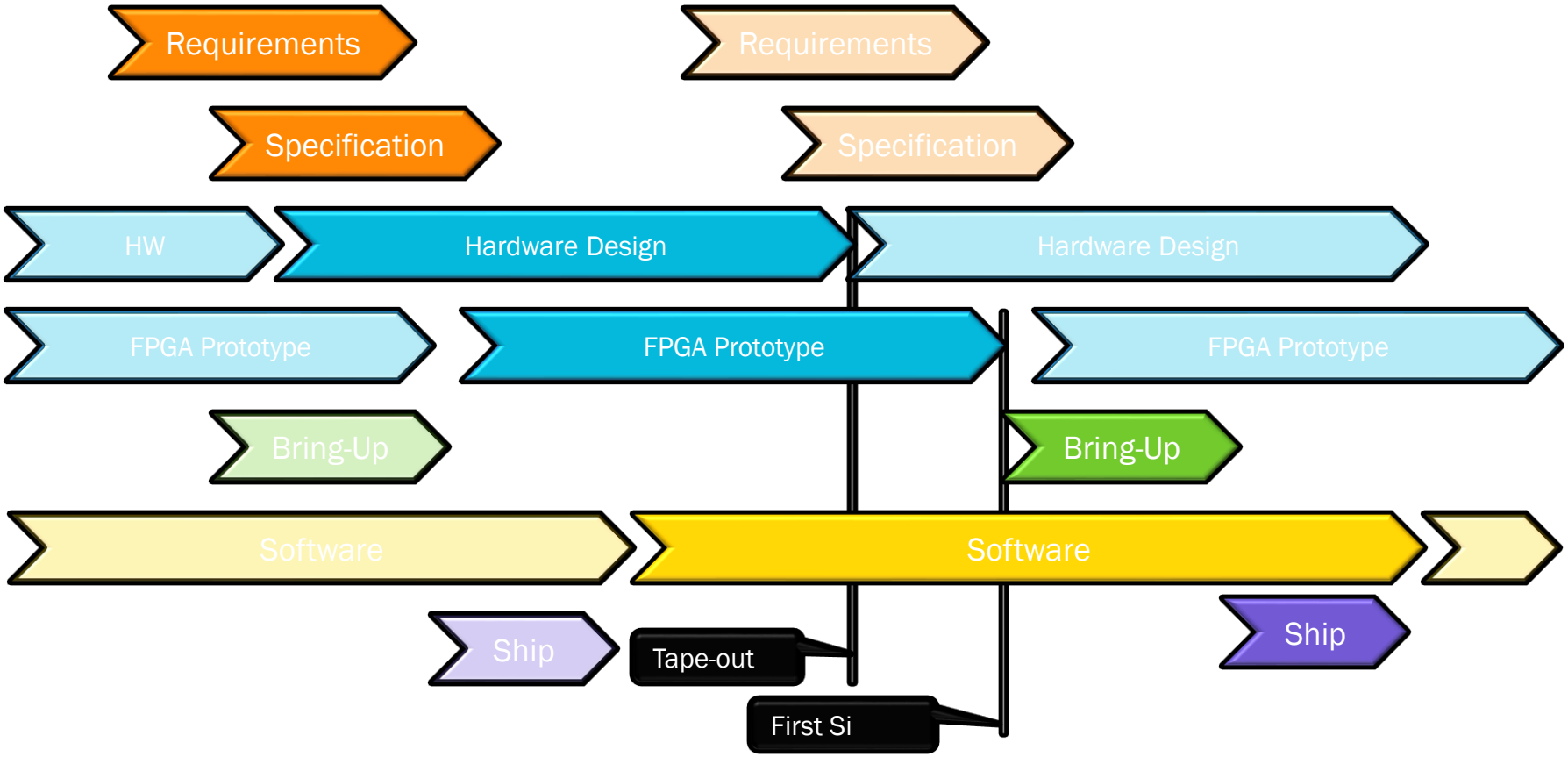
- Development process is proven
- You have customers using generations of your chips
- Chip complexity is growing more processors and more RAMs, all running faster
- Re-spins will be very costly
- Customers need the next generation systems sooner
- Each new system has more and more software
- Software is always the last to finish



EXAMPLE: SIMPLE PROJECT DESIGN FLOW (SOC)

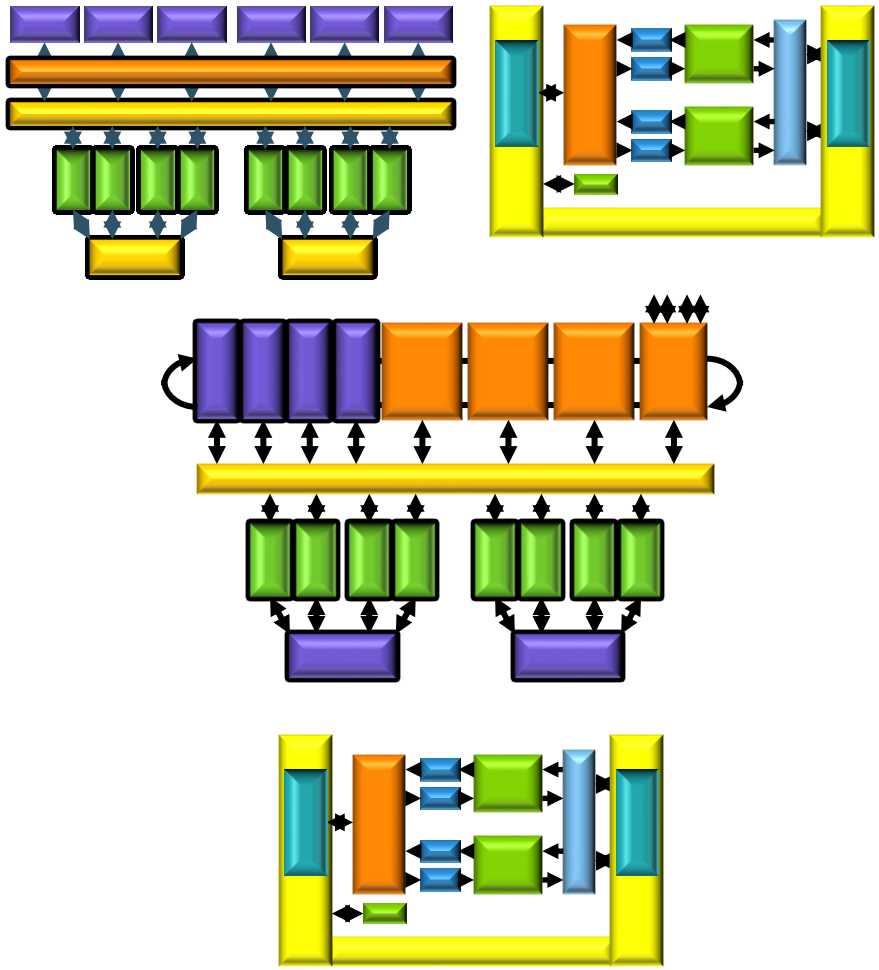


EXAMPLE: PROJECT DESIGN FLOW SOC



SYSTEMC TLM USE-CASES

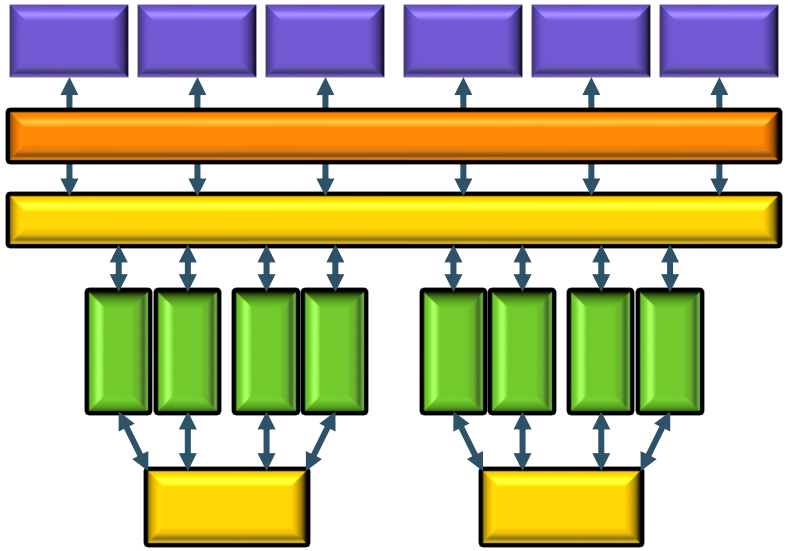
- System Architecture
  - Architecture exploration
  - System performance models
  - Functional block development
  - System architecture validation
- Software Bring-Up
  - Virtual Prototype
  - Early software development
  - System-level performance modeling
  - Software performance optimizations
- Logic Design
  - Design of High Level Synthesis (HLS)
  - Replace RTL for design entry
  - Tools optimize structure
  - Power aware synthesis



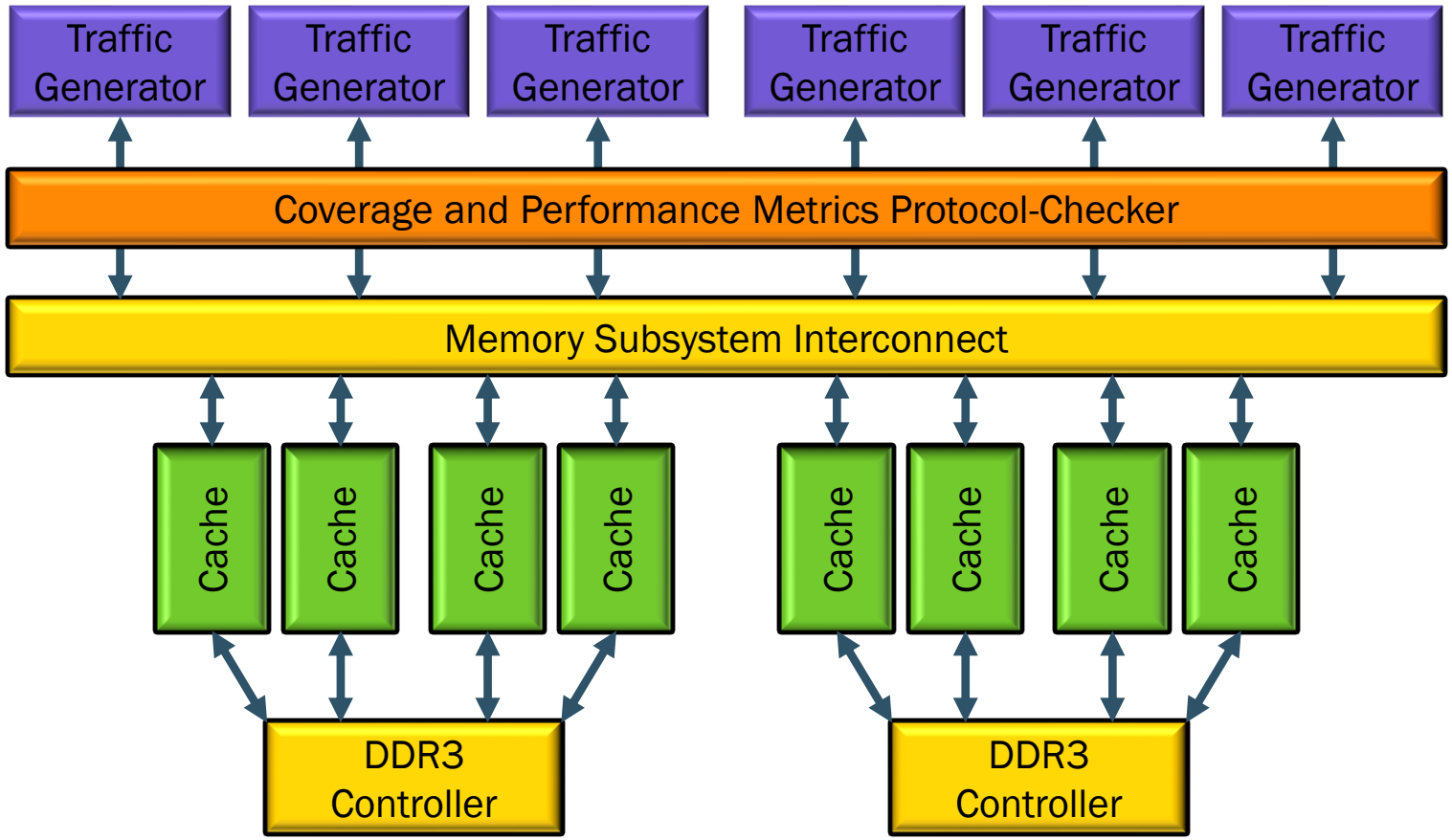
SYSTEM ARCHITECTURE

SYSTEM ARCHITECTURE MODELING INTENTIONS

- Architecture Optimization
  - Validated system-level block diagram
  - Optimized interconnect structure
  - Bottleneck analysis
  - Establish interconnect behavior
    - Bandwidth
    - Latency
    - Priority
  - Allocate system resources
    - Bandwidth
    - Priority
- Validate Requirements
  - Performance
  - New functions
  - Buffering Requirements



PERFORMANCE MODEL BLOCK DIAGRAM



SYSTEMC PERFORMANCE MODEL CONSTRUCTION

- SystemC Approximately Timed (AT) modeling style
  - Non-blocking transport with multiple timing points
  - Processes run in lock-step with simulation time
- Probably does not include Software
- Specialized traffic generators model functional blocks and software
- IP blocks come from multiple sources
  - New and Existing AT models
  - Supplier's AT models
  - RTL Converted to SystemC (faster than RTL)
  - RTL (very slow simulation)
- Includes simulation instrumentation
  - Performance
  - Functional coverage
  - Protocol-checker

SYSTEMC PERFORMANCE MODELING RESULTS

- Optimized interconnect structure
- Bottlenecks identified and corrected
- System resource allocation
  - Bandwidth
  - Latency
  - Priority
- Block-level performance requirements

Products

Reusable

- Test bench for tracking performance
- Foundation for future system sodels
- Domain-specific traffic generators

SYSTEMC PERFORMANCE MODELING RESULTS

- Optimized interconnect structure
- Bottlenecks identified and corrected
- System resource allocation
  - Bandwidth
  - Latency
  - Priority
- Block-level performance requirements

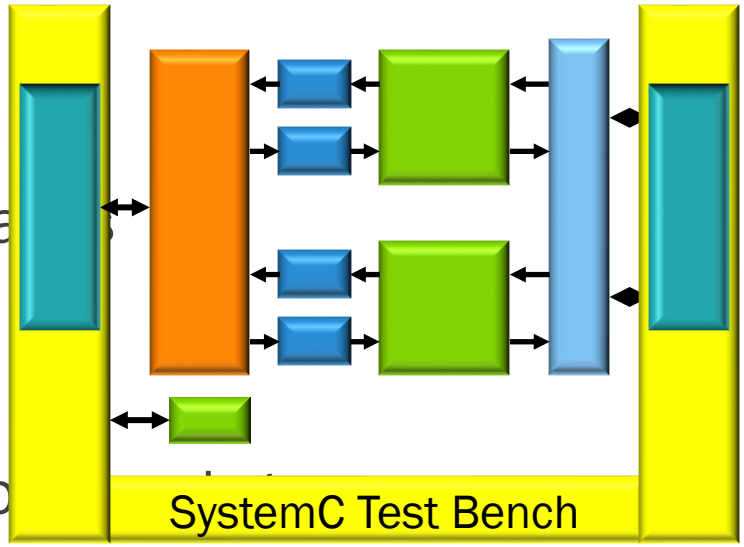
Products

Reusable

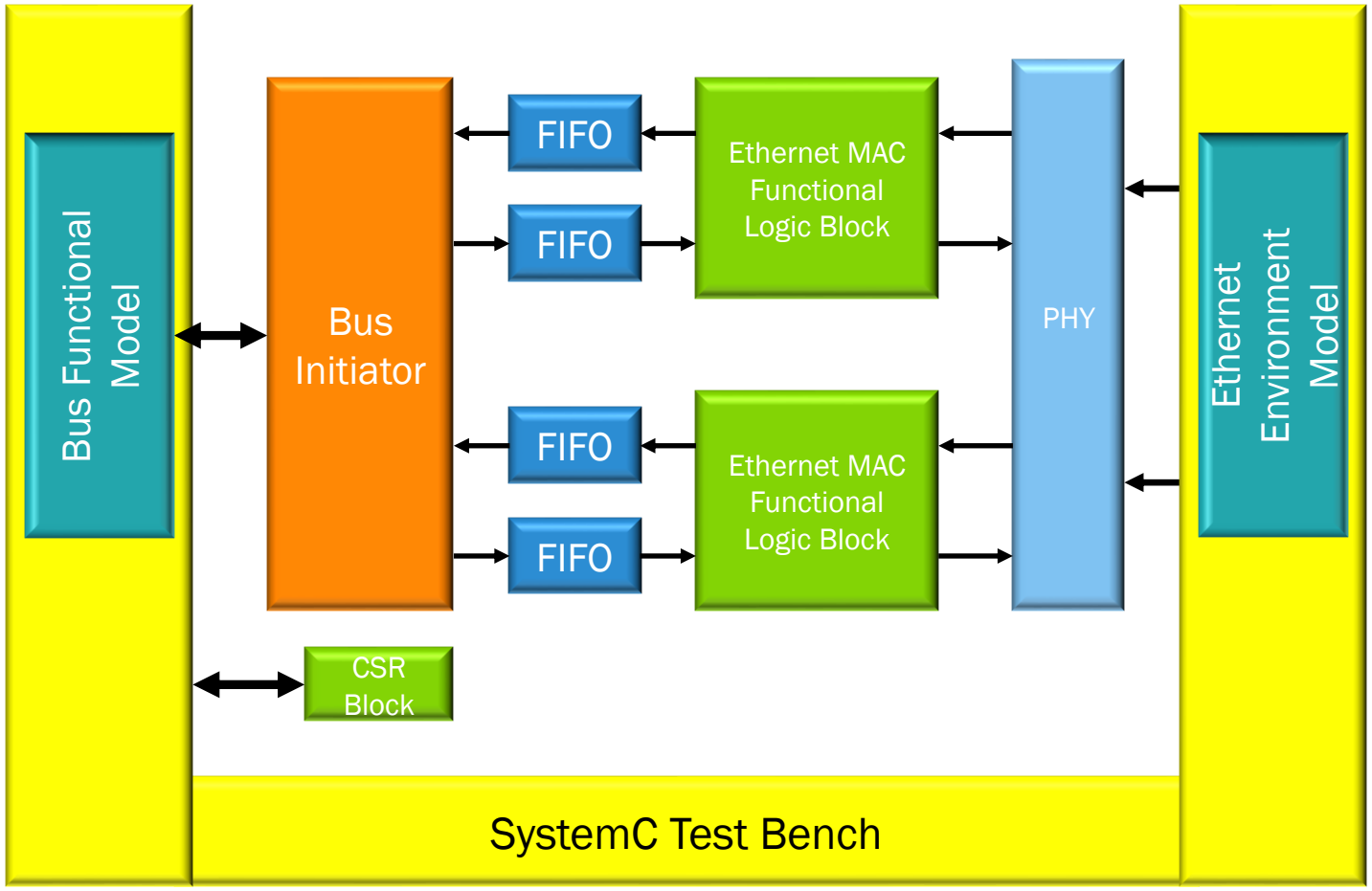
- Test bench for tracking performance
- Foundation for future System Models
- Domain Specific Traffic generators

NEW FUNCTIONAL BLOCK ARCHITECTURE MODELING OBJECTIVES

- Capture executable functional description
- Verify functional behavior
- Verify functional performance
- Define hardware and software interfa
  - Control and Status Registers
  - Shared data structures
- Explore block-level structures, functio
- Identify and model resource contention
- Size FIFO and buffer memories
- Eliminate low-level details from paper specification



FUNCTIONAL BLOCK ARCHITECTURAL MODELING



NEW FUNCTIONAL BLOCK ARCHITECTURAL MODELING CONSTRUCTION

- SystemC Approximately Timed (AT) modeling style
- SystemC environment provides C++, SystemC, and TLM library components
  - C++ Standard template library
  - SystemC data types
  - FIFO
  - Payload event queues (PEQ)
- Model implementation using
  - New functions
  - Sub-block reuse (arbiters and pipeline models)
  - Known accurate TLM interfaces
- Vender supplied sub-block models
- Mix of TLM generic & system unique TLM protocols
- Code generator provided SystemC Control and Status Registers (CSR)
- Configurable delays allow exploration and tracking of RTL implementation

- Test bench models hardware and software environment
- Test bench includes directed and constrained random tests

NEW FUNCTIONAL BLOCK ARCHITECTURAL MODELING

RESULTS

- Executable functional model of the new design
- Documentation and test for the most active CSRs
- Functional test bench
- Performance parameters for feedback to system performance model

Products

- Complete and detailed structure for RTL design
- Design base for High Level Synthesis (HLS)
- Near-complete model for a Virtual Prototype
- Functional test for RTL and HLS implementation
- Reference model for SystemVerilog RTL test bench
- System unique TLM protocols
- New sub-components for future models

Reusable

SOFTWARE INTEGRATION

TRADITIONAL SOFTWARE BRING-UP

- Hardware team implements RTL
- Software team continues working on previous project

Design Specification Communication

- Initial Software bring-up waits for a Hardware Prototype
  - Hardware emulation
  - FPGA prototypes

Requires Completed RTL

- Hardware prototype shortcomings
  - Only a limited number of prototype systems available
  - Difficult to fit complete SoC on prototyping hardware
  - Prototypes run at a reduced clock rate
  - I/O clock-rates may be fast or slow relative to prototype core clo
  - Low-level physical interfaces may not be the same as ASIC

Not Complete SoC

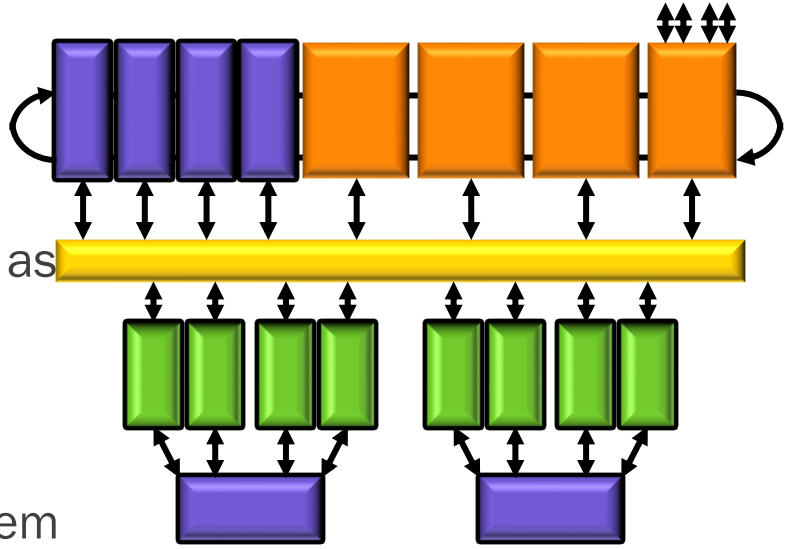
Timing & Function Not 100% Accurate

- Final software bring-up and debug requires complete Sy

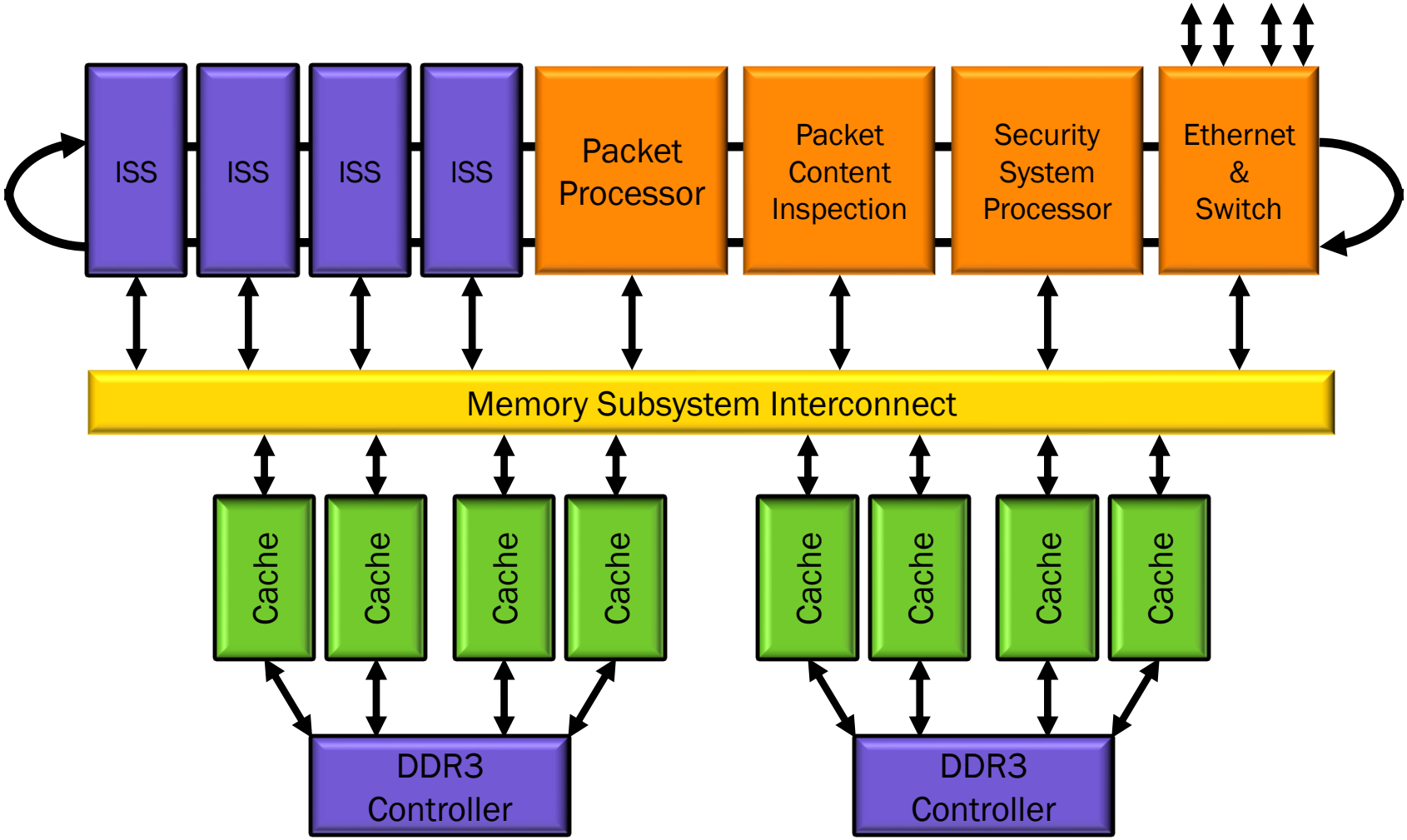
Always True

VIRTUAL PROTOTYPE MODEL
OBJECTIVES

- Minimize post-silicon software delays (deliver product sooner)
- Allow time for interface refinement before RTL design freeze
- Start software bring-up and debug as early as possible
- Provide a superior software debug environment
- Provide a bit-accurate prototype of the system
- Favor execution speed over fine-grain operation ordering
- May have performance-accurate mode



SYSTEMC VIRTUAL PROTOTYPE MODEL



VIRTUAL PROTOTYPE MODEL
CONSTRUCTION

- SystemC Loosely Timed (LT) modeling style
  - Blocking transport with only two timing points
  - Temporal decoupling to allow processes to run ahead of simulation time
  - Direct memory interface (DMI) for high-speed memory access

- Use vendor-supplied processor models (ISS)
- Reuse system performance model structure and memory subsystem
- Speed up AT functional block by adding
  - Blocking transport
  - Debug transport
  - Direct memory interface
  - Optimized functional implementations for speed

- Implement new LT models as needed for a complete system simulation
- Use a code generator to implement complete CSRs for all blocks

- Provide configuration options to select AT or LT simulation mode

VIRTUAL PROTOTYPE MODEL
RESULTS

- Pre-silicon full system model of software for software bring-up
- Full system model for customer application development
- Application performance optimization
- Simulation prototypes are less expensive than hardware emulation

- LT only block can be converted LT/AT for future performance modeling
- AT and LT blocks are a design base for High Level Synthesis (HLS)
- Virtual Prototype can be configured for performance modeling
- Foundation for next generation Virtual Prototype

VIRTUAL PROTOTYPE MODEL
RESULTS

- Pre-silicon full system model for software for software bring-up
- Full system model customer application development
- Application performance optimization
- Simulation prototypes are less expensive the hardware emulation

Products

- LT only block can be enhanced (AT) for future performance modeling
- AT and LT blocks are a design base for High Level Synthesis (HLS)
- Virtual Prototype could be configured for performance modeling
- Foundation for next generation Virtual Prototype

Reusable

LOGIC DESIGN

TRADITIONAL LOGIC DESIGN

- System architecture team defines requirements
  - Function
  - Performance
  - Power and area allocation
  - bandwidth & priority

Requirements

- RTL team creates design specification
  - HW/SW interface definition
  - Functional sub-block description
  - Buffer and FIFO sizing

RTL
Design Specification

- RTL team implements micro-architecture
- Verification team creates test environment

RTL & Test Bench
Implementation

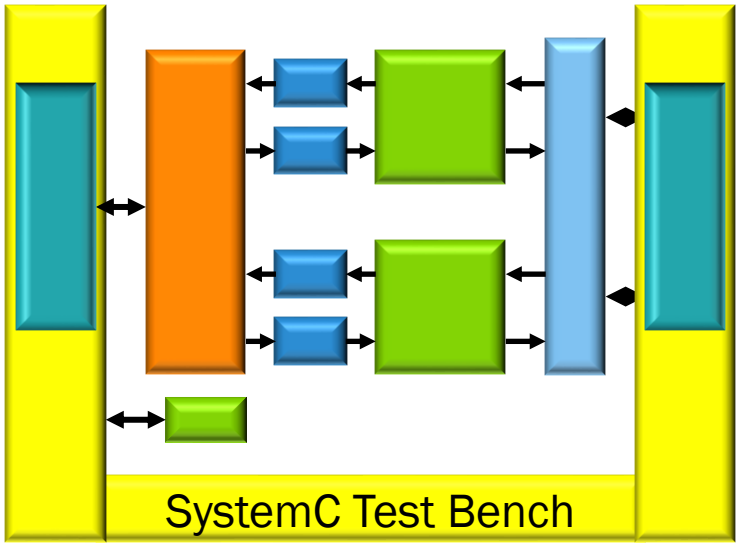
- Hardware team achieves design closure by iterating
  - Micro-architecture changes
  - RTL-to-gates compilation

Design Closure
Iterations

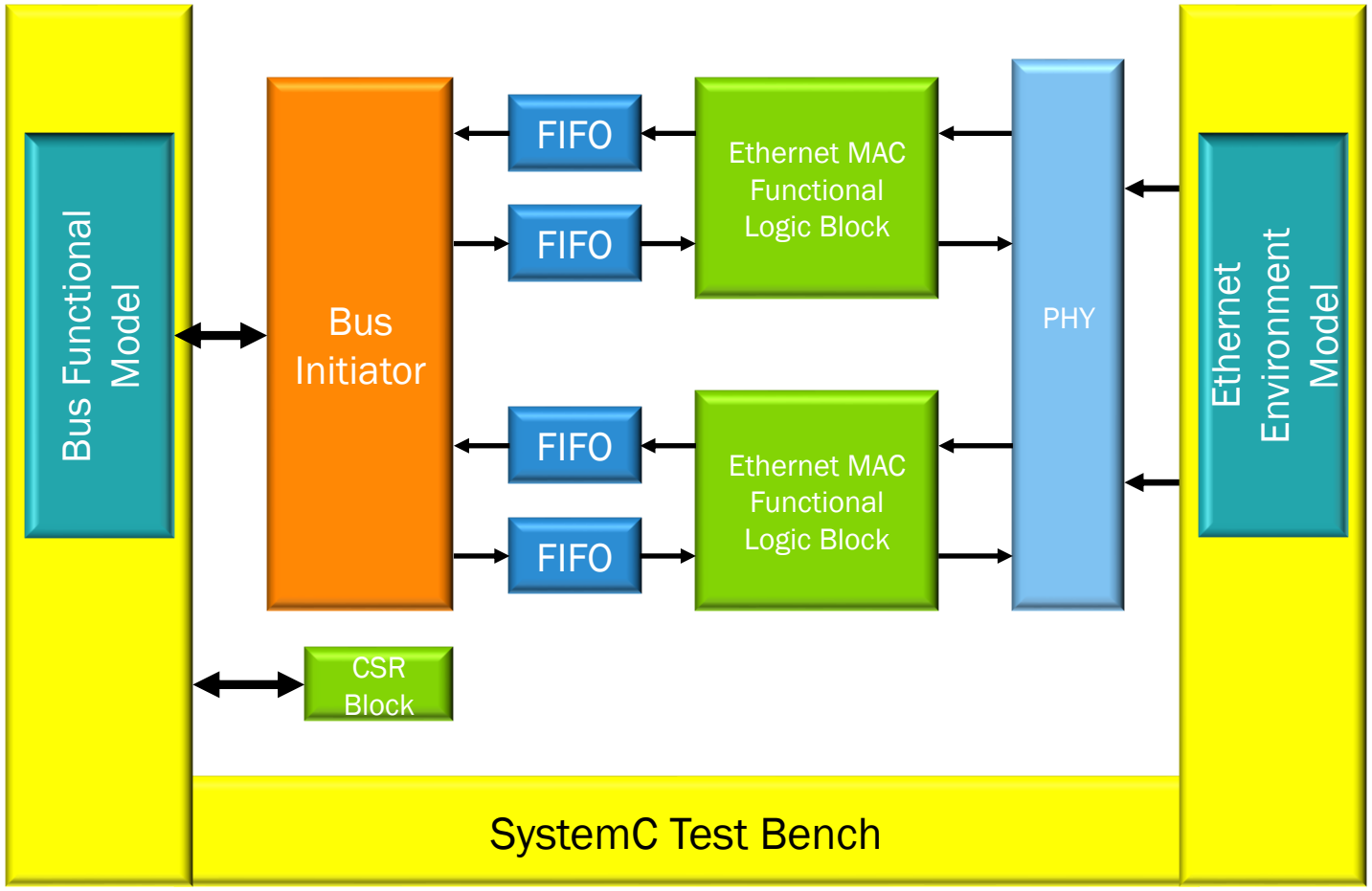
DESIGN OF HIGH LEVEL SYNTHESIS (HLS) OBJECTIVES

- Let the tool do the implementation of RTL micro-architecture

- Use architecture or Virtual Prototype model as design base
- Use algorithms written in C or C++ as functional base
- Eliminate RTL implementation of micro-architecture
  - Pipelining and state-machines
  - FIFO and arbiters
- Create design that can be easily optimized
  - Speed
  - Area
  - Power



DESIGN OF HIGH LEVEL SYNTHESIS (HLS)



DESIGN OF HLS: CONSTRUCTION

- Existing SystemC models should be used as a design base.
  - Paper spec – not the best option
  - C C++ algorithm – better
  - An existing SystemC LT model – better still
  - AT models may over constrain – may be good
- Synthesis tools cannot support the full richness of SystemC and C++
  - Capabilities differ from vender to vendor
- SystemC models may require restructuring for synthesis
- Synthesis is controlled by directives and a technology library
- Synthesis results can be optimized for
  - Speed
  - Power
  - Area
- Synthesis-generated RTL merges with existing RTL design flow

DESIGN OF HLS: RESULTS

- Optimized RTL implementation

Product

- Synthesizable SystemC code for reuse
  - Different ASIC technologies
  - Different optimizations of speed, area and power

Reusable

- SystemC models for future use
  - Performance modeling
  - Architectural exploration

- SystemC models for functional design evolution