How to apply TLM 2.0 to the "Real World"
A Tutorial

Electronic **S**ystem **L**evel **X**perts

www.eslx.com

Bill Bunton – presenting

David Black – co-author

Agenda

- Motivation and Background 
- Overview of TLM 2 Terminology
- Project Level Use Case - Evolution

ESLX

TLM Motivation

- Industry
  - Enable system-level virtual prototypes for software development
  - Enable modeling of Memory Mapped Bus
  - Define an interoperable interface for IP vendor
  - Create an IP component ecosystem
- OSCI
  - Evolve the SystemC Standard to address industry's need
  - Ensure interoperability with the existing standard
  - Provide guidance to continued evolution of modeling interconnect
  - Maintain SystemC leadership as the language for ESL modeling

ESLX

History of TLM

- TLM 1.0
  - Approved by OSCI April 2005
  - Blocking, Pass-by-value, common payload unspecified
- TLM2 2.0 D1
  - Released for review December 2006
  - Non-blocking, Pass-by-value, common payload
- TLM 2.0 D2
  - Released for review November 2007
  - Interoperable levels of abstraction
  - Pass-by-reference for speed
  - Common and extensible generic payload

ESLX

TLM 2.0 Draft 2 Requirements

- Speed, speed, and more speed
  - Multi-megahertz on a modern server
  - Boot OS in less than 2 minutes
- Focus on Memory-Mapped Bus interconnects
  - Perceived industry need
- Selectable levels of abstraction (fidelity)
  - Addresses different use cases – both technical and business
- Use case driven
  - Platform for early software development
  - Hardware Software integration
  - Additional use cases
- Ease of use

ESLX

References

- OSCI Resources www.systemc.org
- TLM 2.0-d2 Manual 1_0_0
  - TLM-2 draft kit examples and unit test
  - Additional examples pending review by working group
- The OSCI TLM-2 in 2008: Tutorial Tuesday at 1:30
- Watch for updates of web pages throughout the industry

ESLX

Agenda

- Motivation and Background
- Overview of TLM 2 Terminology 
- Project Level Use Case - Evolution

ESLX

Overview of TLM 2 Terminology

- Generic Payload
- TLM transport Blocking and Non-Blocking
- Un-Timed Modeling Style
- Loosely-Timed Modeling Style
- Approximately-Timed Modeling Style
- Performance Accelerators

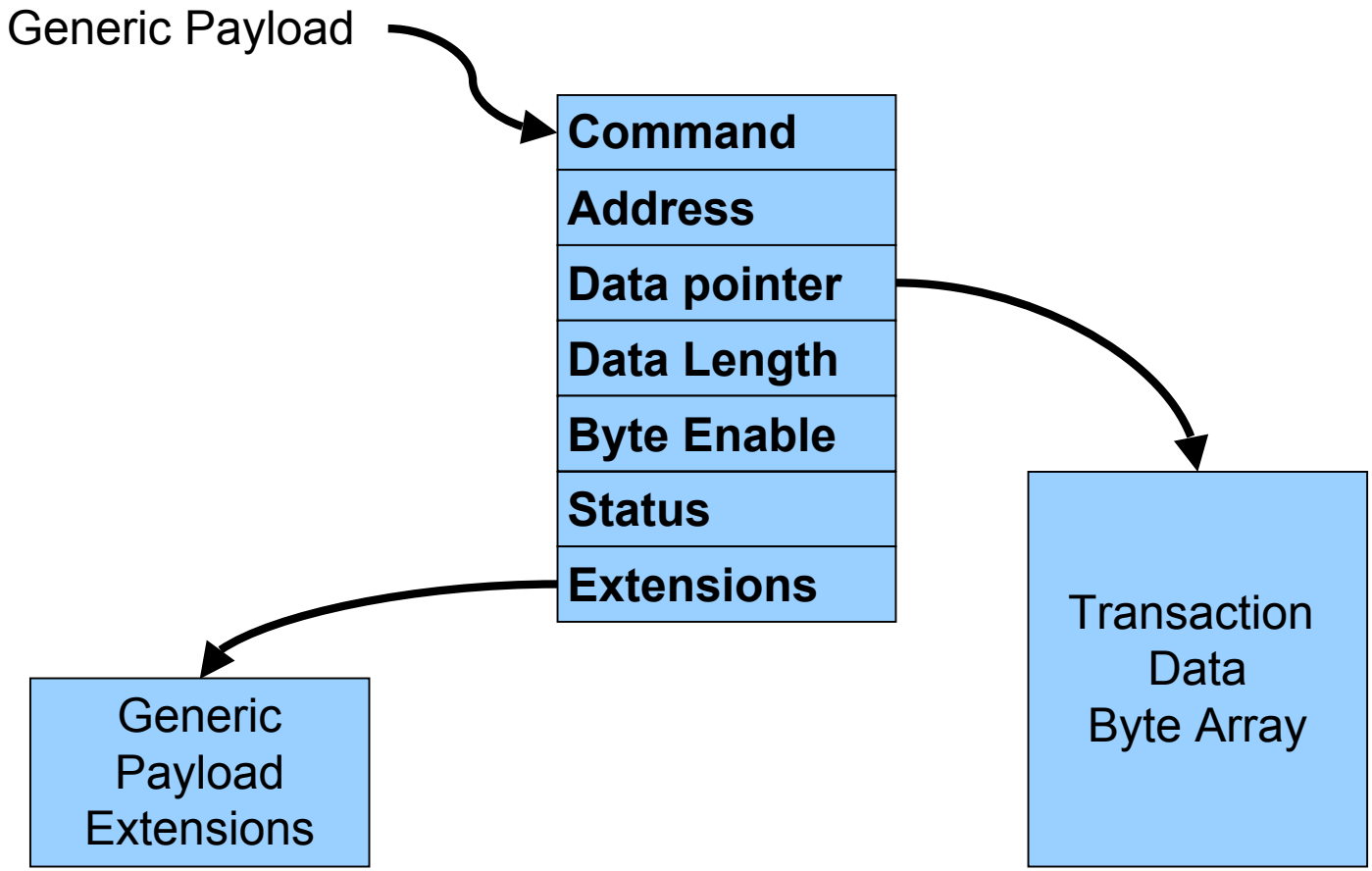
ESLX

The Generic Payload

- Off-the-shelf general purpose payload for MMB
  - Abstract bus modelling
  - Typical attributes of memory-mapped busses
    - Command, address, data, byte enables, single word transfers, burst transfers, streaming, response status
- Extensibility for specific MMB protocols
  - Ignorable extensions
  - Mandatory extensions with compile-time checking
  - Simplified protocol bridging

ESLX

Generic Payload Structure



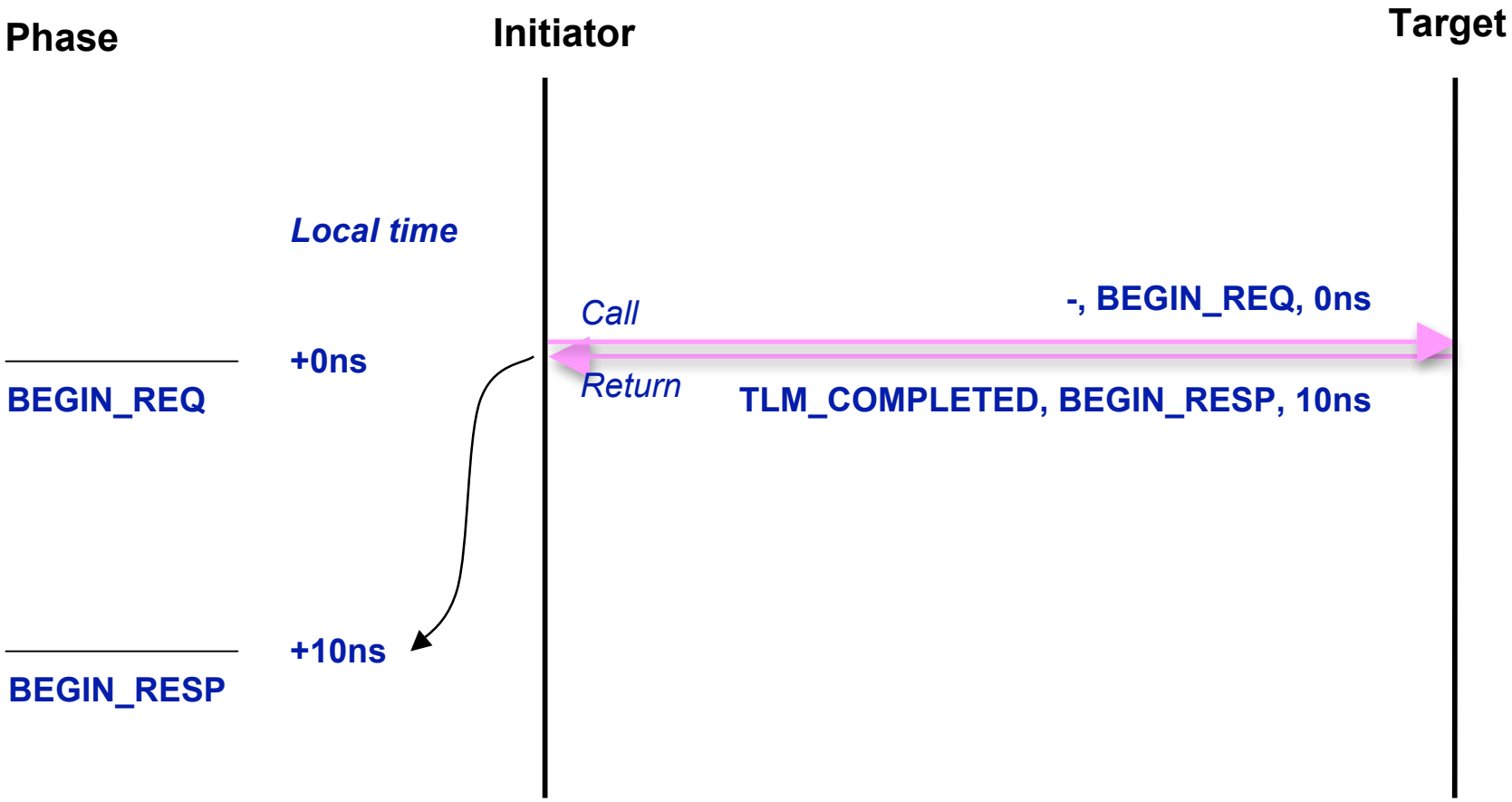
ESLX

Loosely-Timed (LT)

- Focus on speed
- Single-phase
  - Expected style
  - Non-blocking call where transaction is complete on return
    - Completes without relinquishing the simulator
    - Target elapsed time is returned to initiator
- Two-Phase
  - Allows target to advance simulator time
  - Two Non-blocking calls
    - Initiator to target (request)
    - Target to initiator (response)
  - Synchronization feature to support temporal decoupling

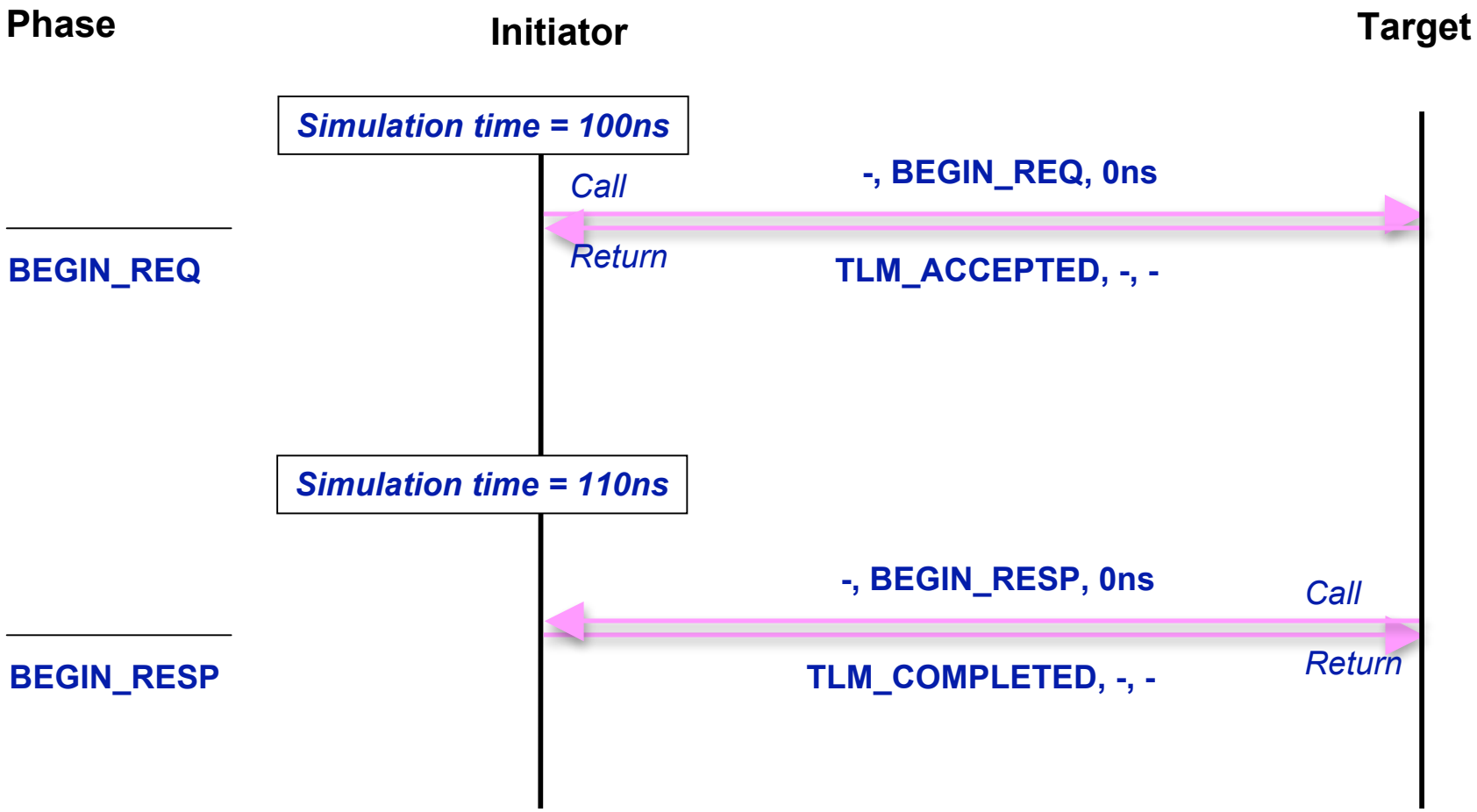
ESLX

Loosely-Timed (LT) Single-Phase Timing Ladder



ESLX

Loosely-Timed (LT) Two-Phase Timing Ladder



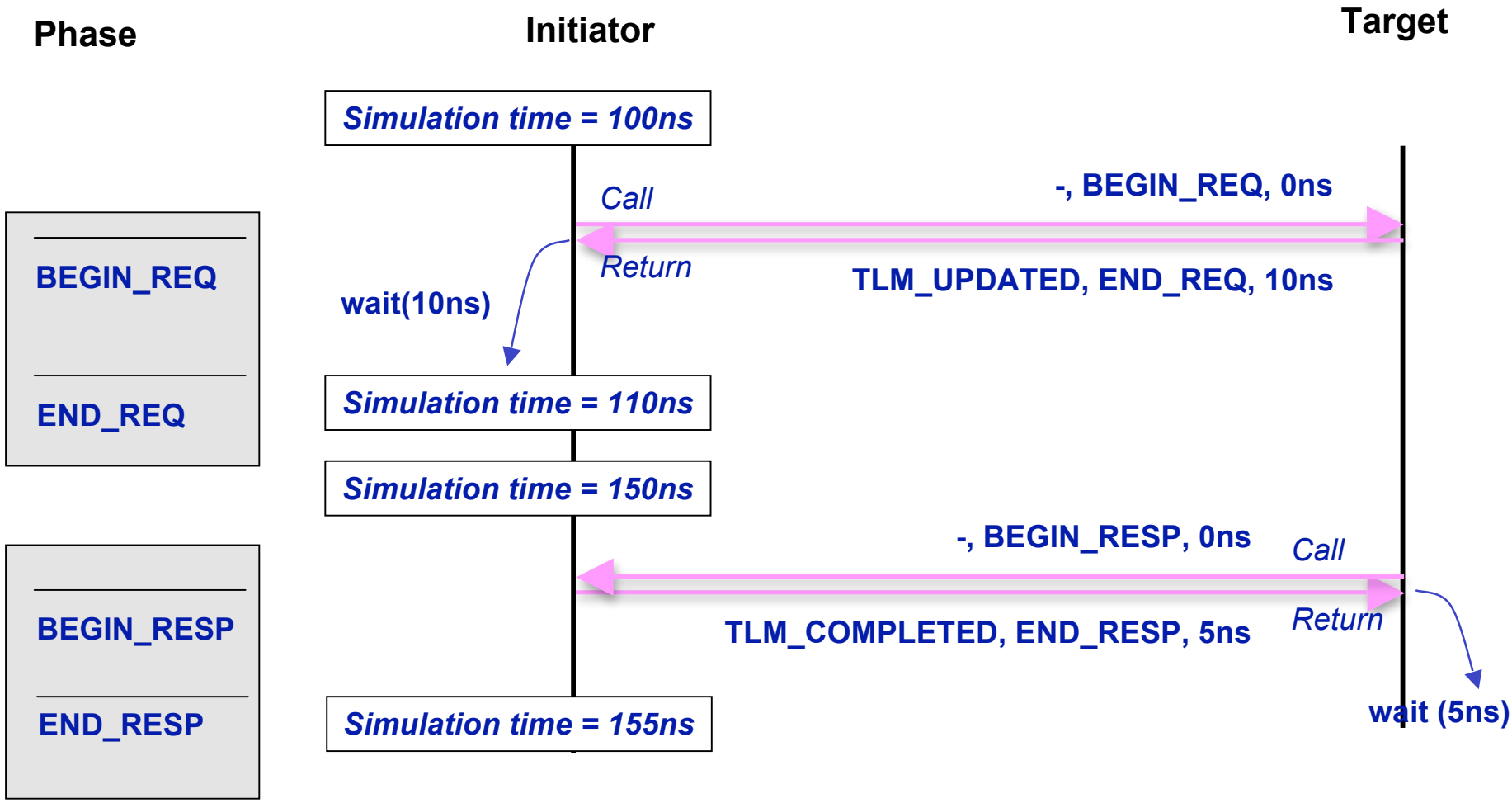
ESLX

Approximately-Timed (AT)

- Enhanced timing accuracy
- Two-Phase
  - Two Non-blocking calls
    - Initiator to target (request) – target returns delay to next phase
    - Target to initiator (response) – initiator returns delay to next phase
- Four-Phase
  - Four Non-blocking calls
    - Initiator to target (begin request)
    - Target to initiator (end request)
    - Target to initiator (begin response)
    - Initiator to target (end response)

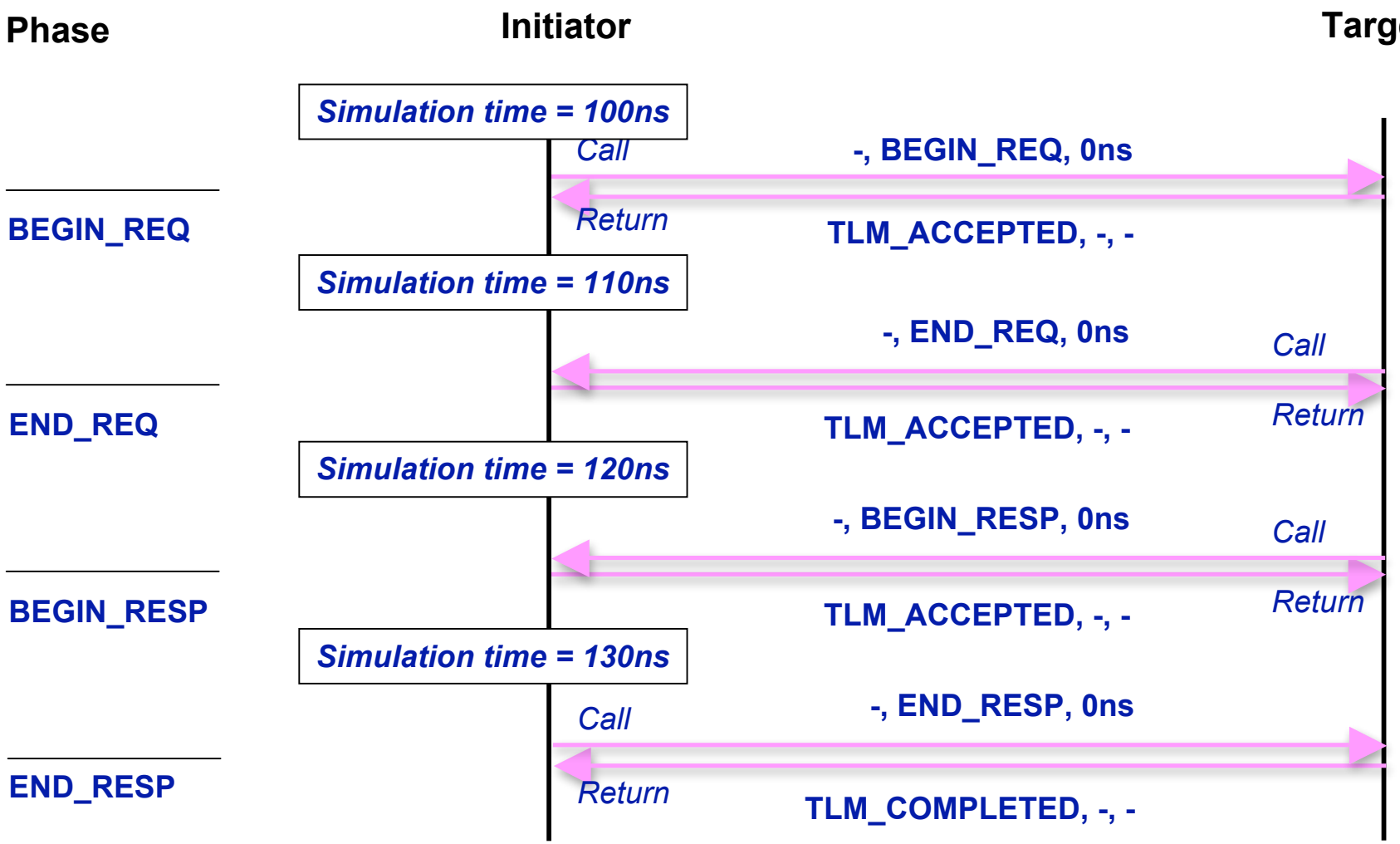
ESLX

Two-Phase AT Timing Ladder



ESLX

Four-Phase AT Timing Ladder



ESLX

Performance Accelerators

- Model Style Switching
  - LT and AT modeling styles fundamentally compatible
  - Components can switch styles during execution to select simulation speed or accuracy
- Temporal Decoupling
  - Multiple initiator transactions without relinquishing control simulator
  - At time quantum, control is returned to the simulator kernel
  - Not restricted to a single initiator
  - Reduces simulator context switching (speed)
- Direct Memory Interface (DMI)
  - Cooperating targets and initiators bypass the model interconnect
  - Initiators use pointer access to target memory
  - Eliminates functions call
  - May eliminate context switching

ESLX

Agenda

- Motivation and Background
- Overview of TLM 2 Terminology
- Project Level Use Case - Evolution 

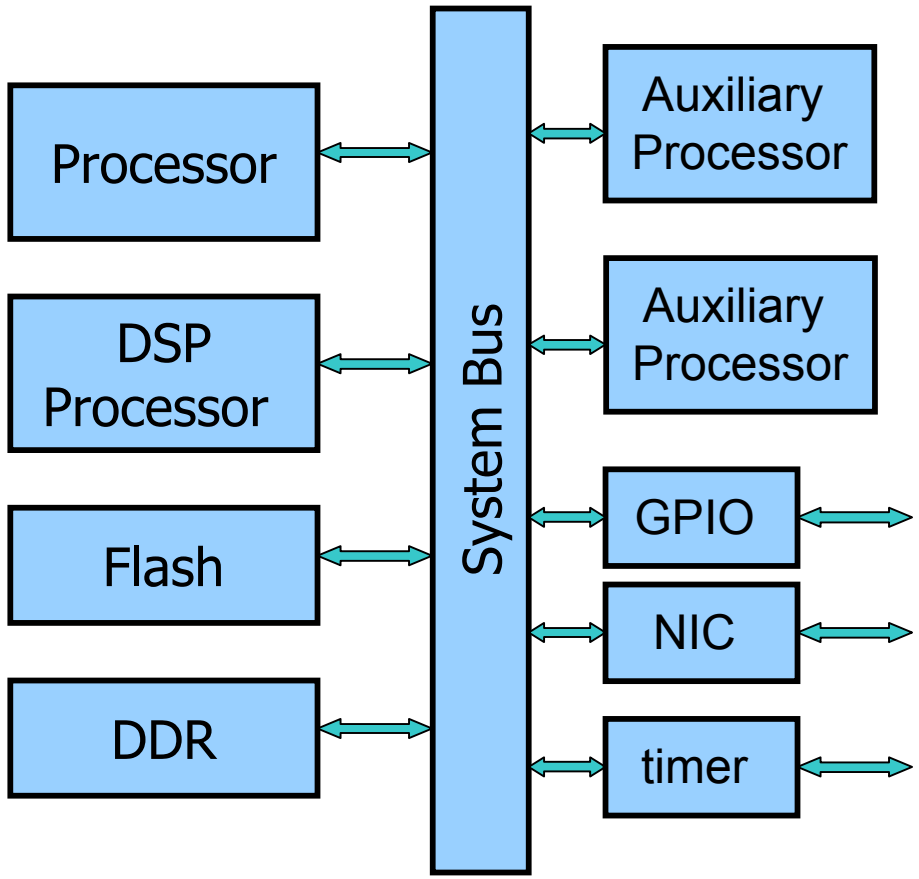
ESLX

Project Level Use Case - Evolution

- System Architectural Modeling
- SW development platform
- HW/SW integration (co-simulation of detailed HW with SW)
- HW refinement performance verification

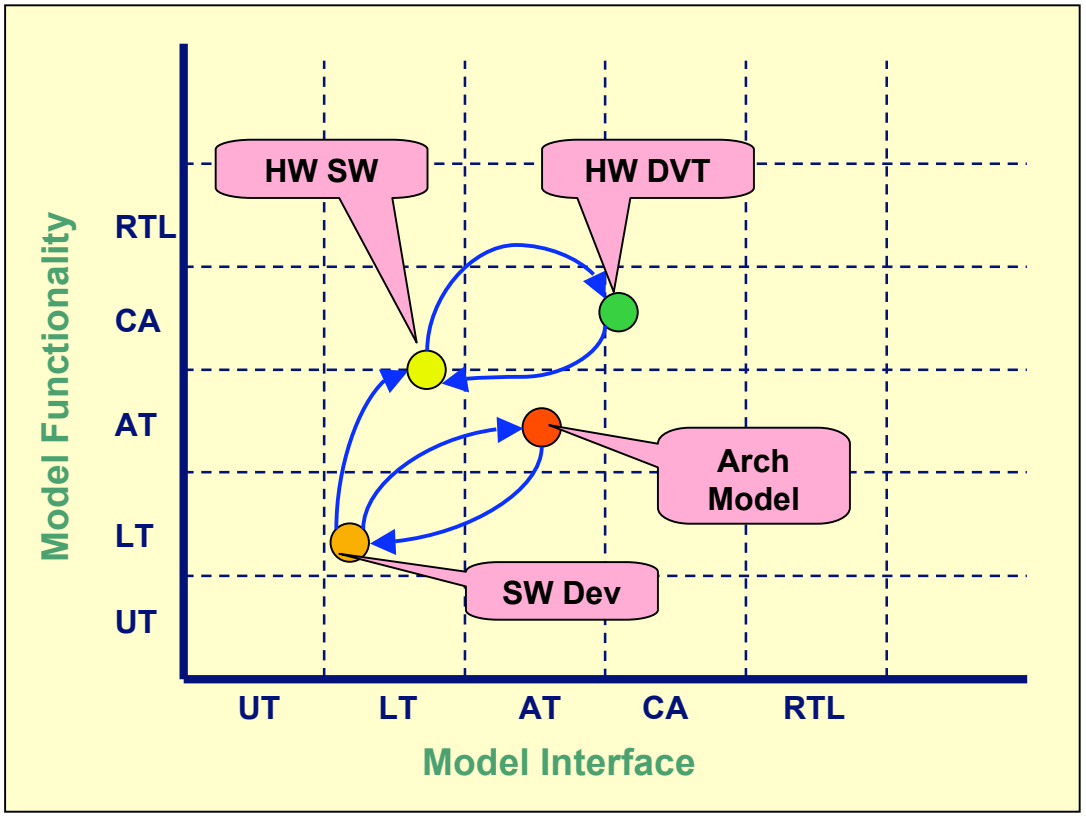
ESLX

Initial System Model



ESLX

Project Evolution



AT Traffic Gen

LT Function Correct

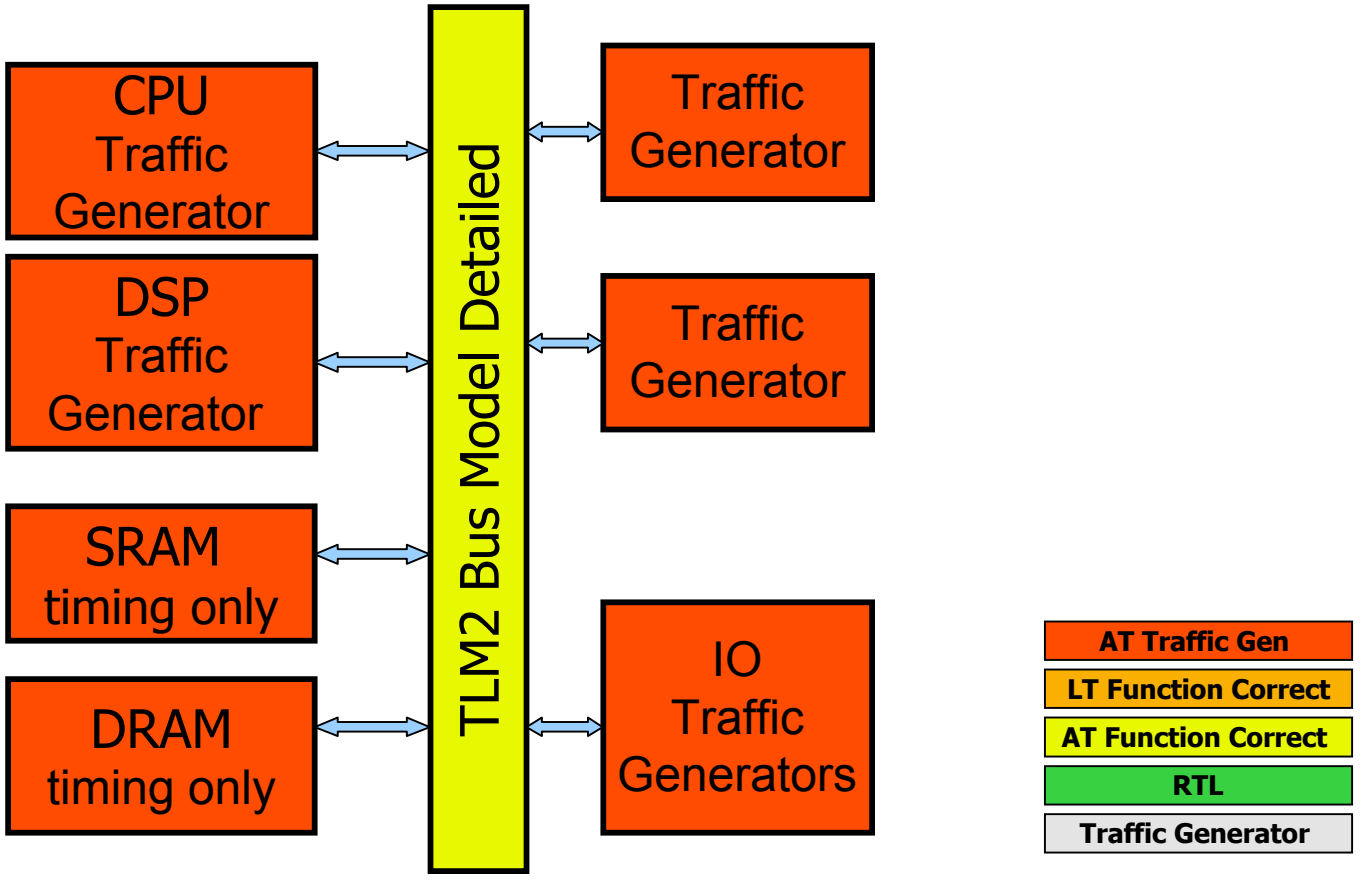
AT Function Correct

RTL

Traffic Generator

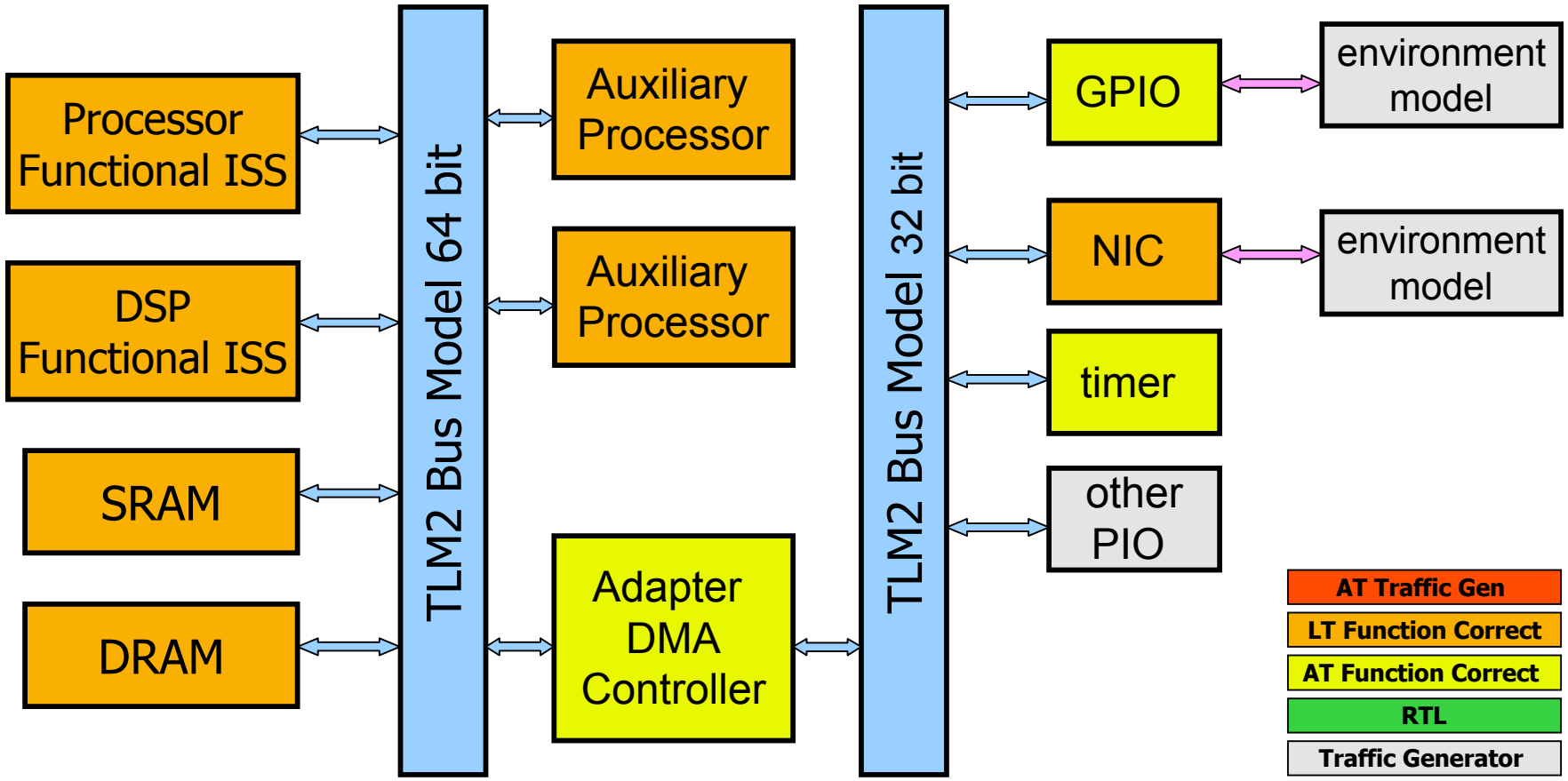
ESLX

System Architectural Modeling



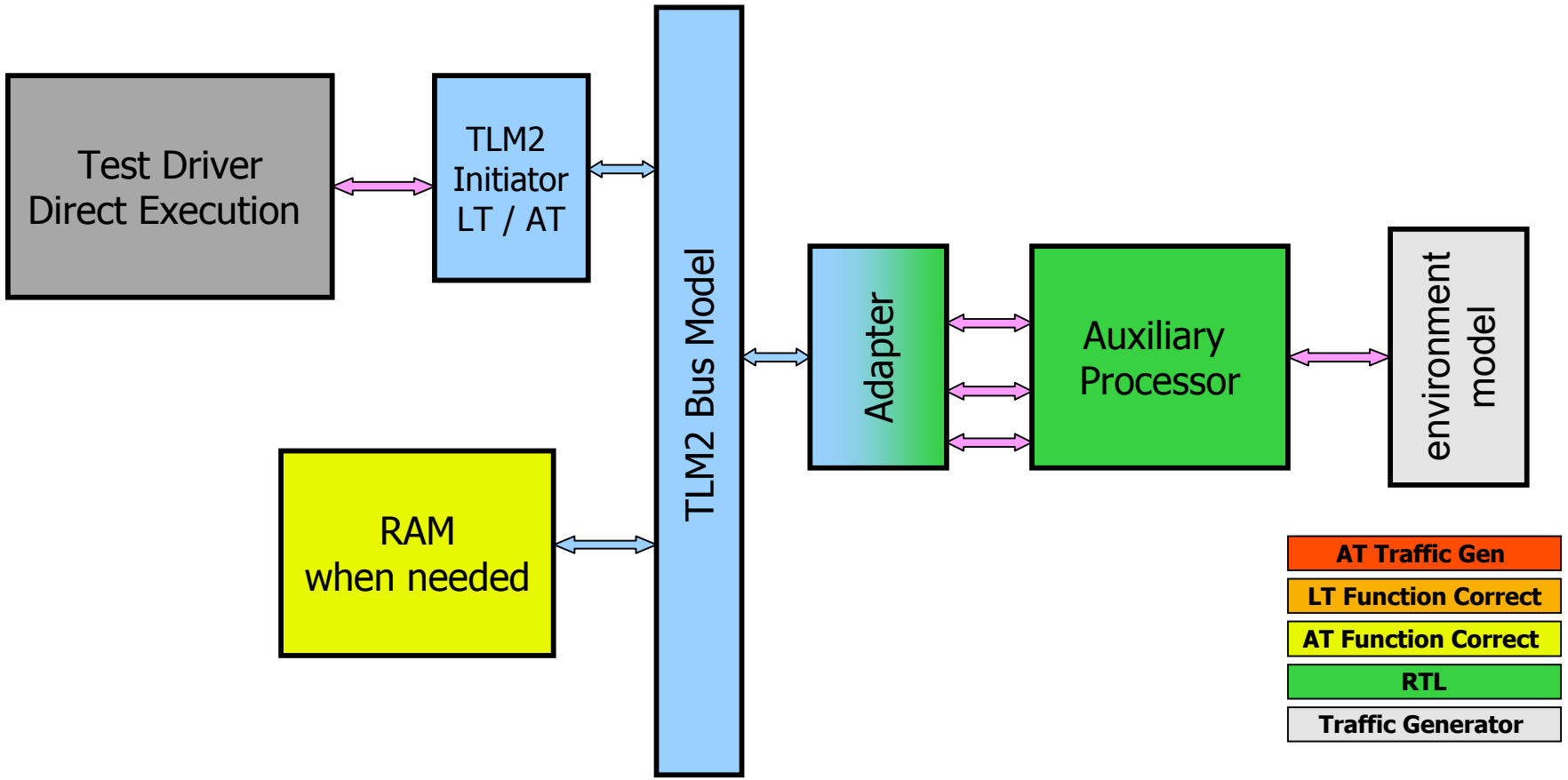
ESLX

System Model for Software Development



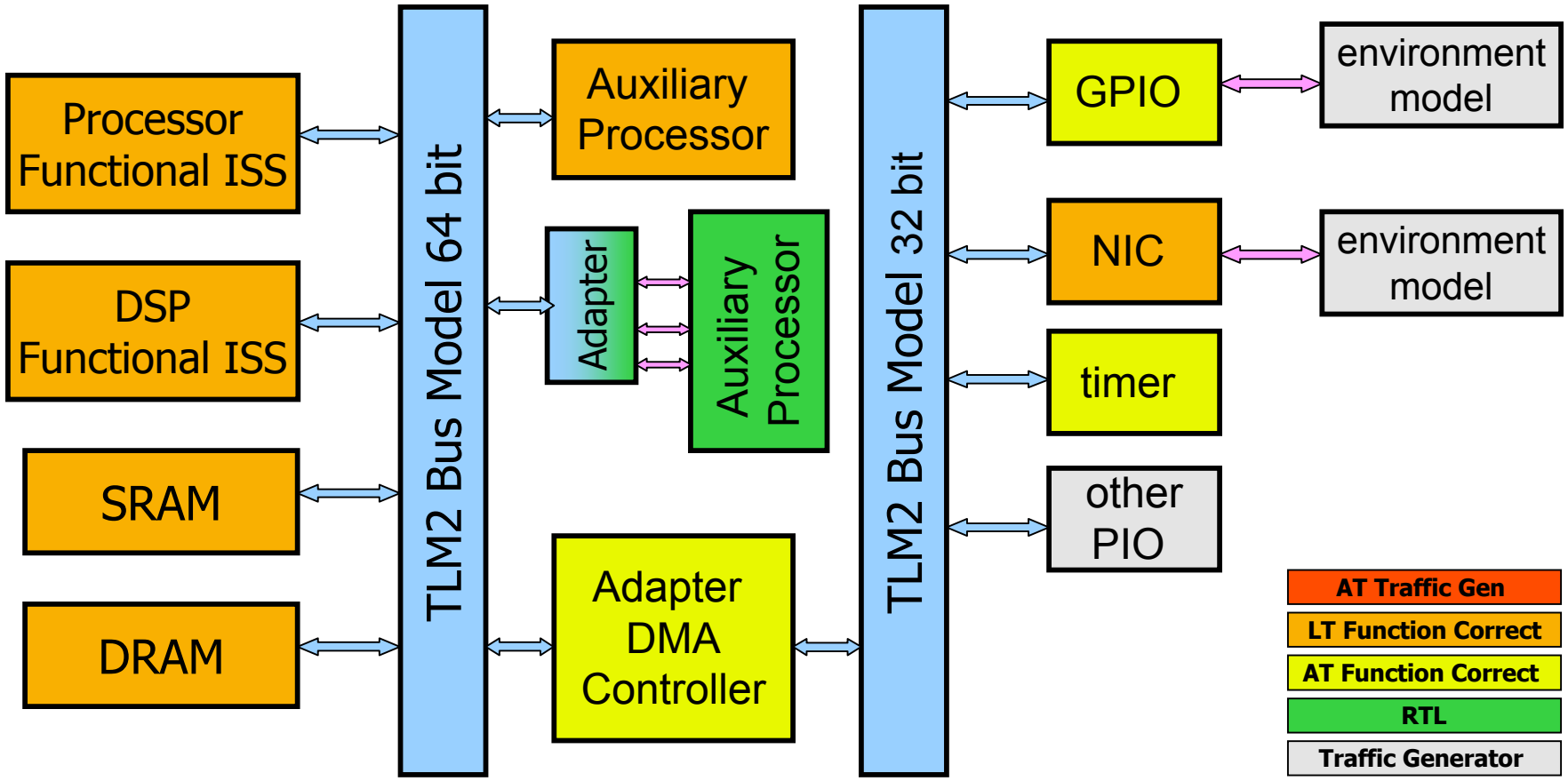
ESLX

HW Refinement and Performance Verification



ESLX

System Model for HW SW integration



ESLX

Conclusion

- TLM2 Draft 2 is available and very usable
  - Meets real world goals of inter-operability of abstraction levels
  - Allows easy, independent refinement of IP components
- ESLX fully supports TLM2 Draft 2
- Questions

ESLX