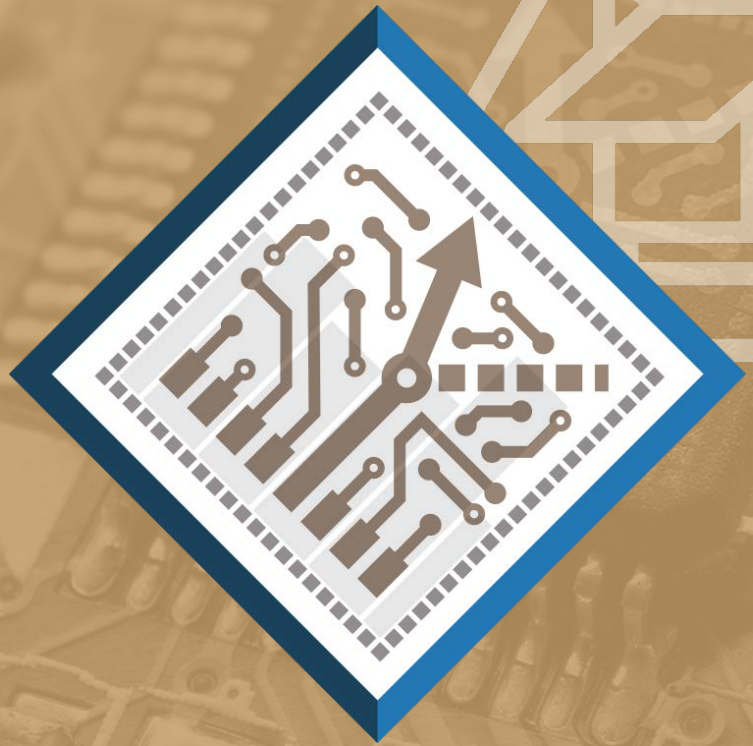


POSH MIXED SIMULATION

EDGAR IGLESIAS

FRANCISCO IGLESIAS
SAKIS PANOU

THE DEMO

POSH OPEN SOURCE HARDWARE – MIXED SIMULATION LIVE DEMO

WWW

QEMU Emulator

Linux

XILINX

ZYNQ UltraSCALE+

TLM

POSH Simulation Bridge

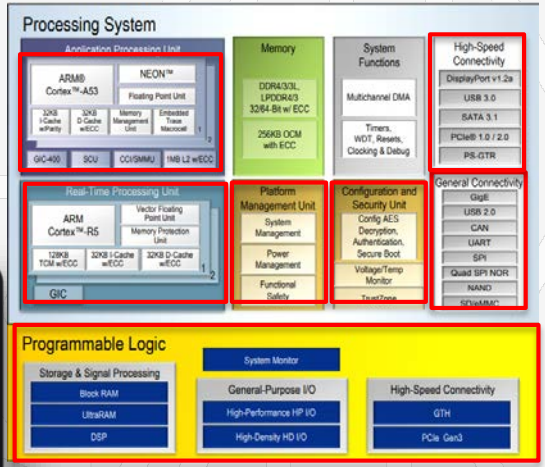
AXI Lite AXI 3 AXI 4 ACE ACE Lite CHI

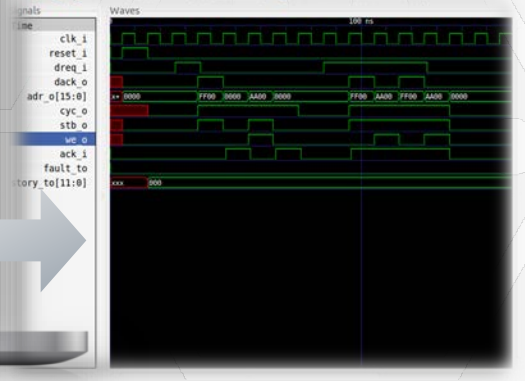
AXI

RTL Simulator

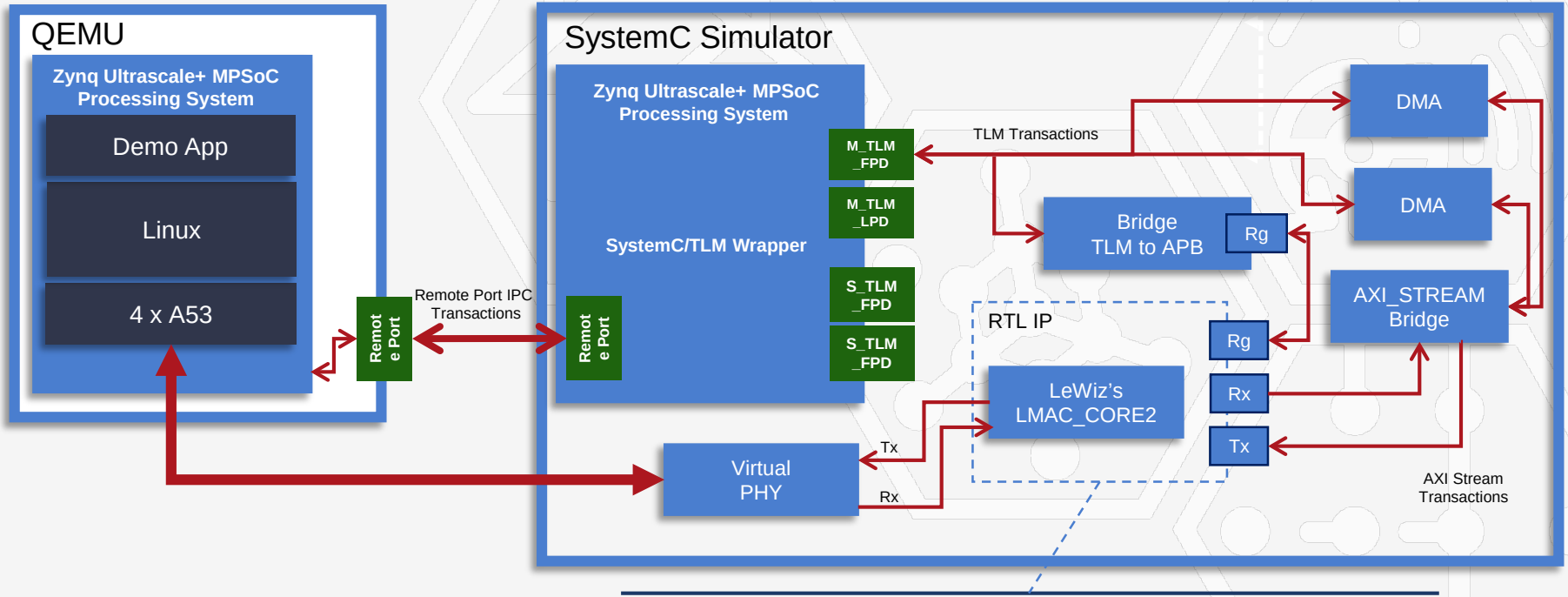
Ethernet MAC Core2
10G/5G/2.5G/1G

LeWiz
Communications





POSH OPEN SOURCE HARDWARE – MIXED SIMULATION LIVE DEMO

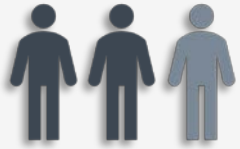


LeWiz Communications Ethernet MAC Core2 10G/5G/2.5G/1G

https://github.com/lewiz-support/LMAC_CORE2

POSH - THE ALTERNATIVE !

3 People



2 Different Companies

+ 1.5 DAYS +

Using only Open Source Tools

10's Millions of Lines of Code
(Software + Hardware)

=

Very Complex System

World's First Fully Open Source Simulation with Real Time Ethernet Data

Discovered Hard To Track, Costly H/W Bugs, Long Before Silicon

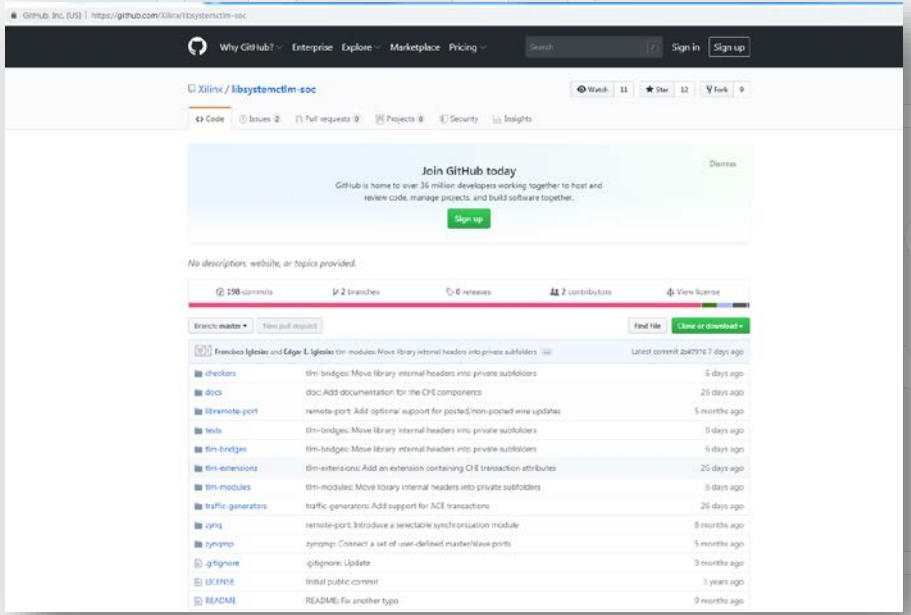
Fast Development

Secure

High Quality

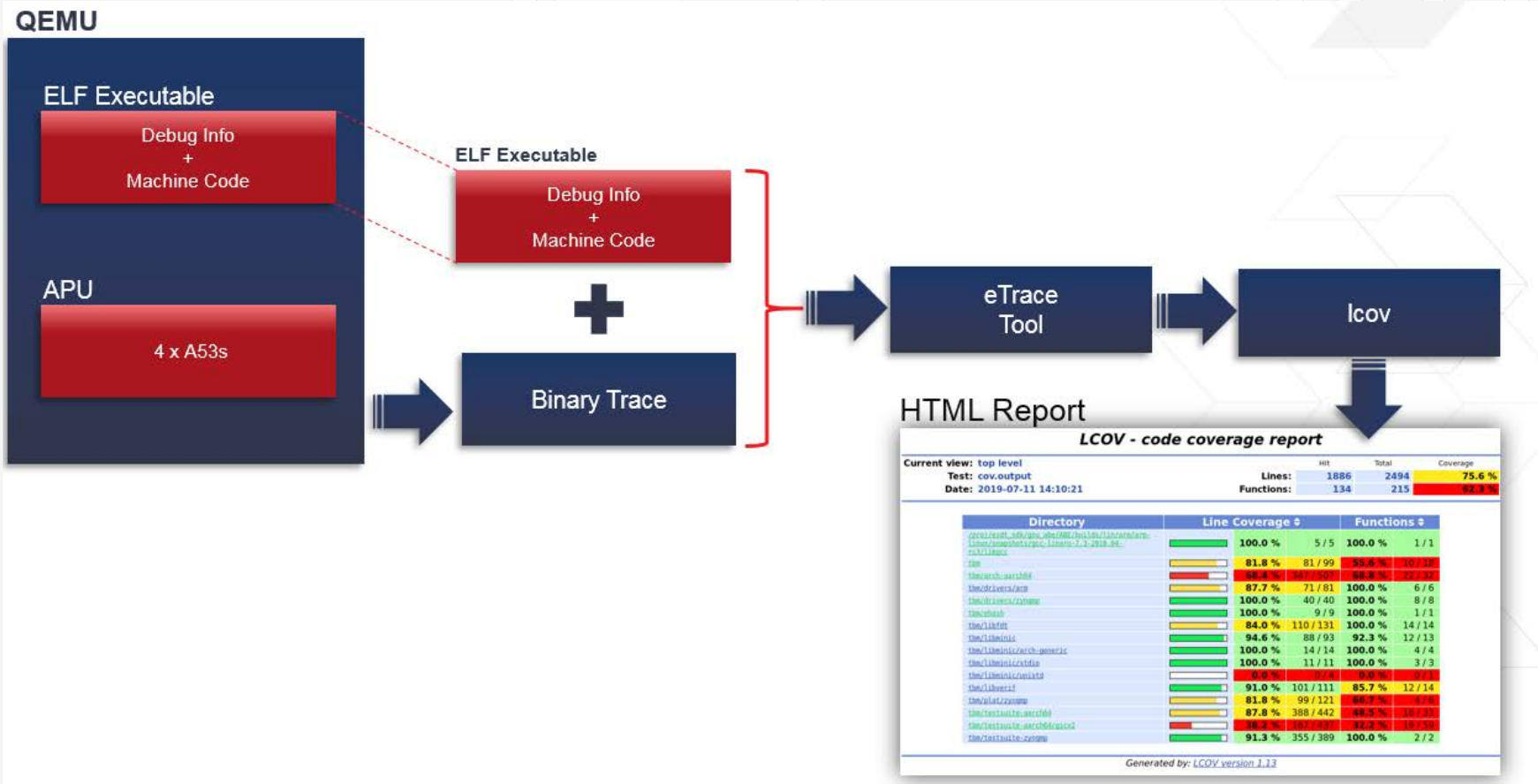
GETTING STARTED

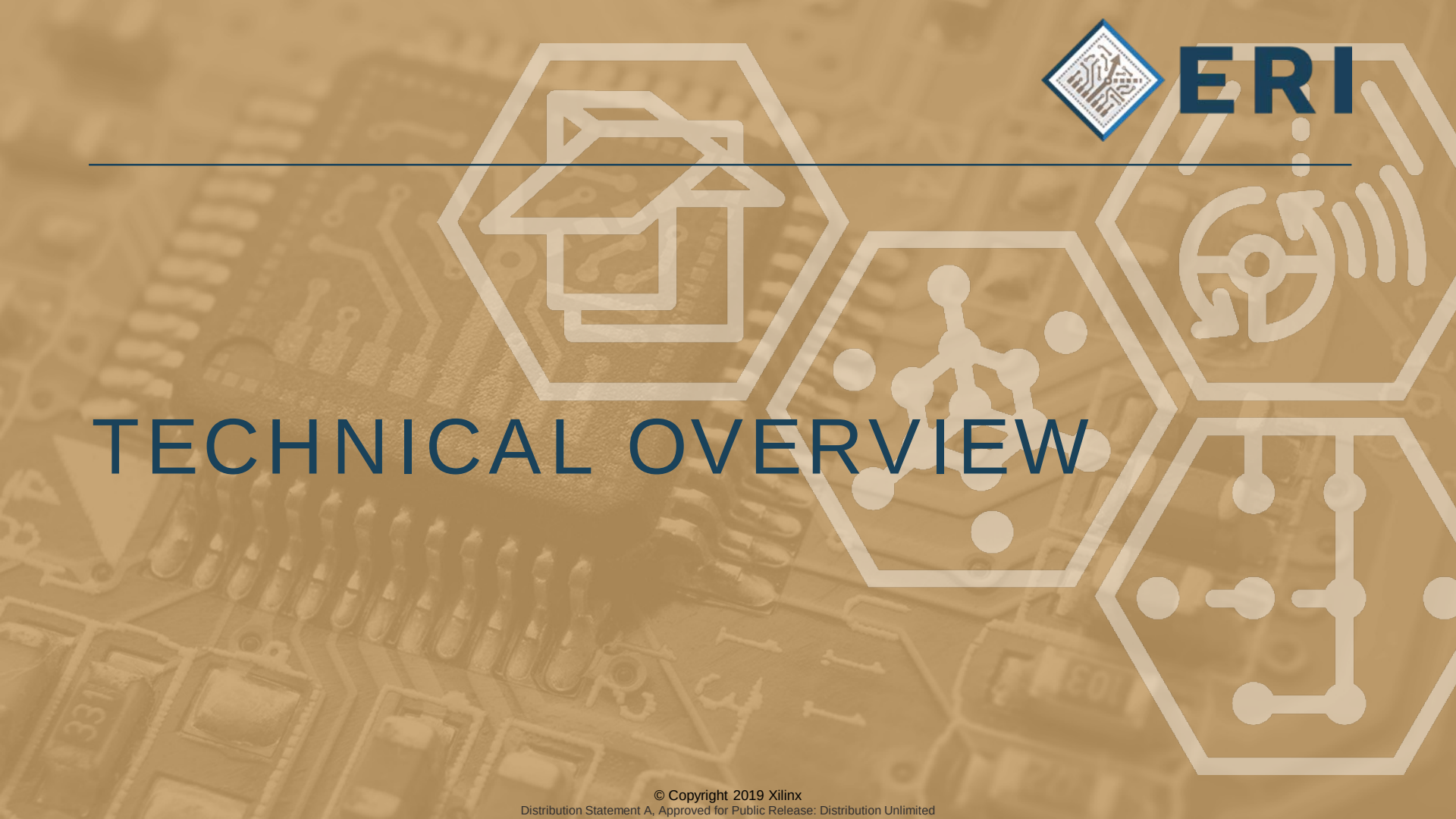
- **Project Location:** https://github.com/Xilinx/libsystemctlm-soc



- **Getting Started:** https://github.com/Xilinx/libsystemctlm-soc/blob/master/README
  - **Step 1:** Read Dependencies and Configuration Sections in README
  - **Step 2:** git clone https://github.com/Xilinx/libsystemctlm-soc.git
  - **Step 3:** cd libsystemctlm-soc/tests/ && make examples-run

CODE COVERAGE FRAMEWORK



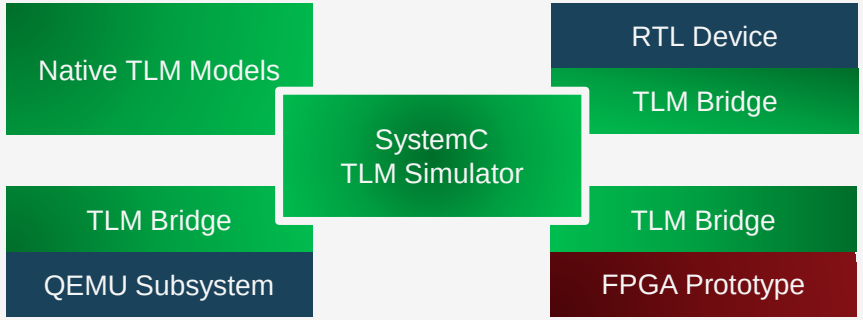


TECHNICAL OVERVIEW

WHAT ARE WE TRYING TO ACHIEVE

- **Goal:** Provide a Robust and Scalable Open Source Mixed Simulation Environment For Complex Designs

LibSystemCTLM-SoC
https://github.com/Xilinx/libsystemctlm-soc

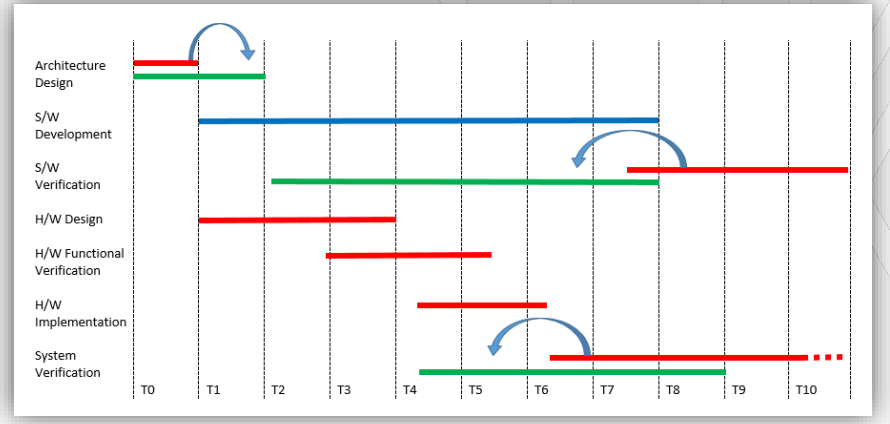


Phase 1a
Phase 1b

Based Entirely On Open Source Projects

WHY ARE WE INVESTING

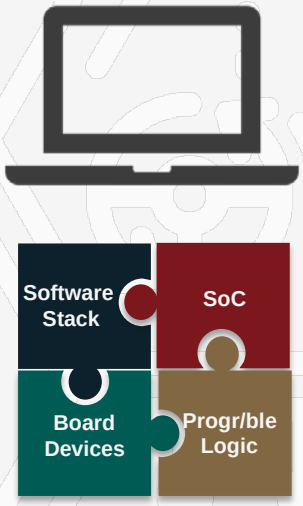
- Model Early, Integrate Continuously, Increase Adoption
  - Behavioral integration bugs are difficult to find and far more costly at later stages
  - Use hardware functional models to drive early and continuous integration
  - Suitable for software development, system architecture, functional hardware IP verification



- Enabling Open Source Hardware & Software Development
  - Think System Level Solutions - Full System Emulation
  - Using Only Open Source Tools & Resources
  - Enable the “ARMY” of open source software engineers to cross the hardware threshold

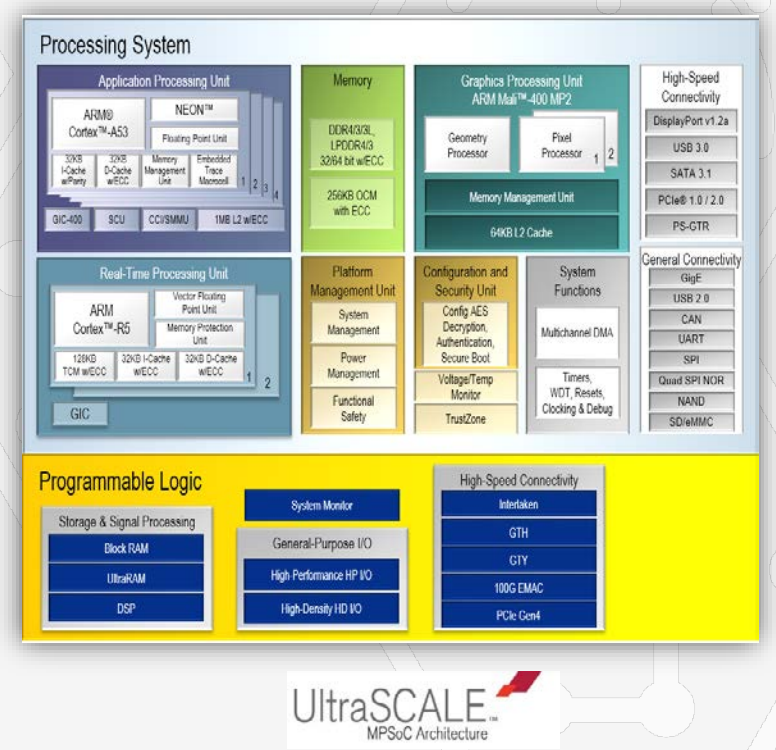
THE BASICS OF QEMU

- QEMU is a Functionally Accurate Virtual Emulation Platform
  - QEMU Stands For **Q**uick **EMU**lator
  - Develop, Debug & Run Your Software Exactly As You Would Do On A Board
  - But…Without the Hassle of Cables, Boards. Your Laptop Is All You Need
- QEMU Platform Puts All the System Pieces Together
  - The Multi Core Heterogeneous SoC + Peripherals
  - The Programmable Logic IP
  - The Various Board Devices
- QEMU is Really Fast !
  - Our Zynq Ultrascale+ MPSoC QEMU can boot Linux Kernel in ~ 20 secs
- QEMU very popular among open source developers
  - KVM
  - GCC
  - Linux
  - Yocto
  - Linux Kernel

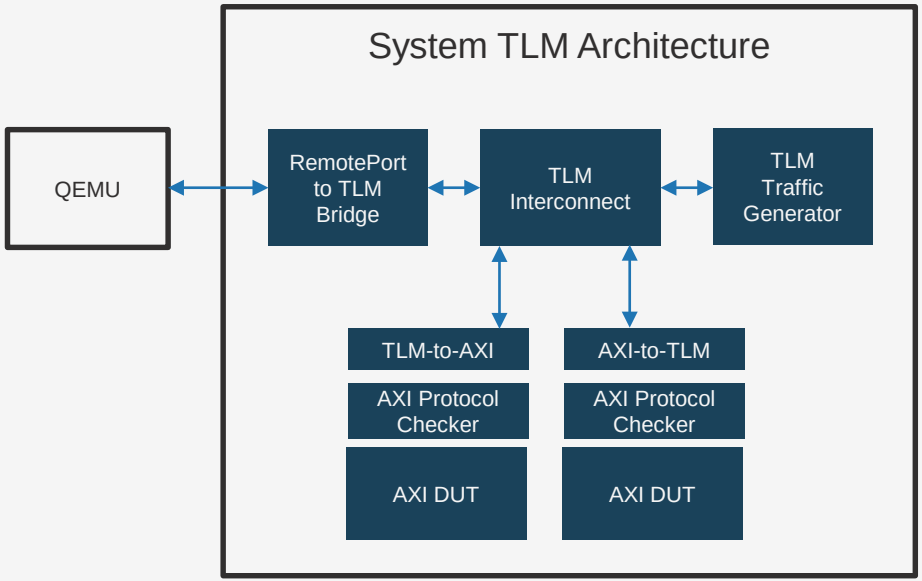


WE PRACTICE WHAT WE PREACH

- We are just trying to make it better
  - At Xilinx we have successfully delivered 3 generations of products using full system mixed simulation environments
    - Boot ROM
    - Bootloaders
    - OS (Linux, FreeRTOS, Baremetal…)
    - Hypervisors (Xen)
    - Applications
    - Libraries…..
- When talking about complex designs
  - Think Zynq Ultrascale+ MPSoC is a highly powerful, highly flexible platform, featuring 4 x A53s, 2 x R5s and 1 MicroBlaze (PMU) along with rich collection of high speed peripherals and low power IOs



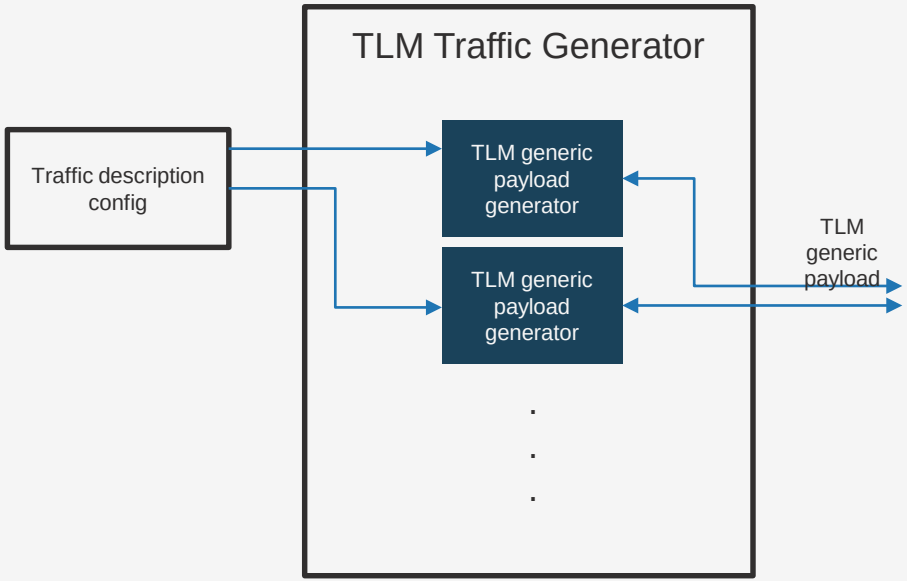
POSH QEMU CO-SIM TLM ARCHITECTURE



>> **Key Architectural Points**

- Remote-port converted to TLM Generic Payloads
  - With Generic Attributes extensions carrying AXI specifics
    - Optional use by intermediate interconnect components
- TLM interconnect allows interconnection of TLM compatible or TLM wrapped devices (such as QEMU)
- TLM/AXI Bridges to direct mapping of TLM to AXI signal wiggling and vice versa but do not add optimizations (such as caching, buffering etc)
- Protocol checker is a passive component monitoring the AXI signals and reporting any protocol errors or inefficiencies that it discovers

POSH TECH DETAIL - TRAFFIC GENERATOR



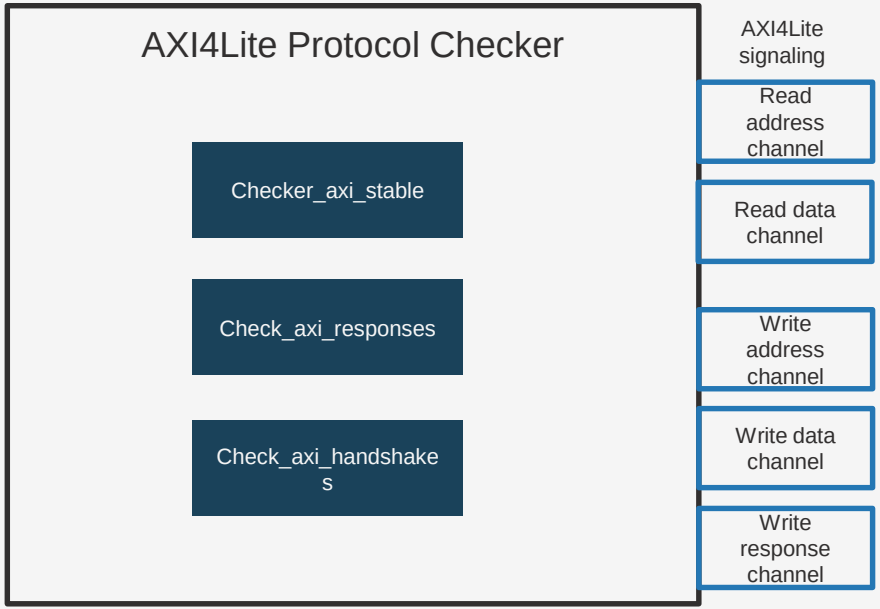
Key Architectural Points

- Generates TLM generic payloads according to a traffic configuration
- Protocol specifics are carried through TLM generic payload extensions
- Supports generation of multiple simultaneous TLM transactions
- Reusable with different type of TLM bridges

Next Steps

- Enable parsing of traffic configurations in JSON format

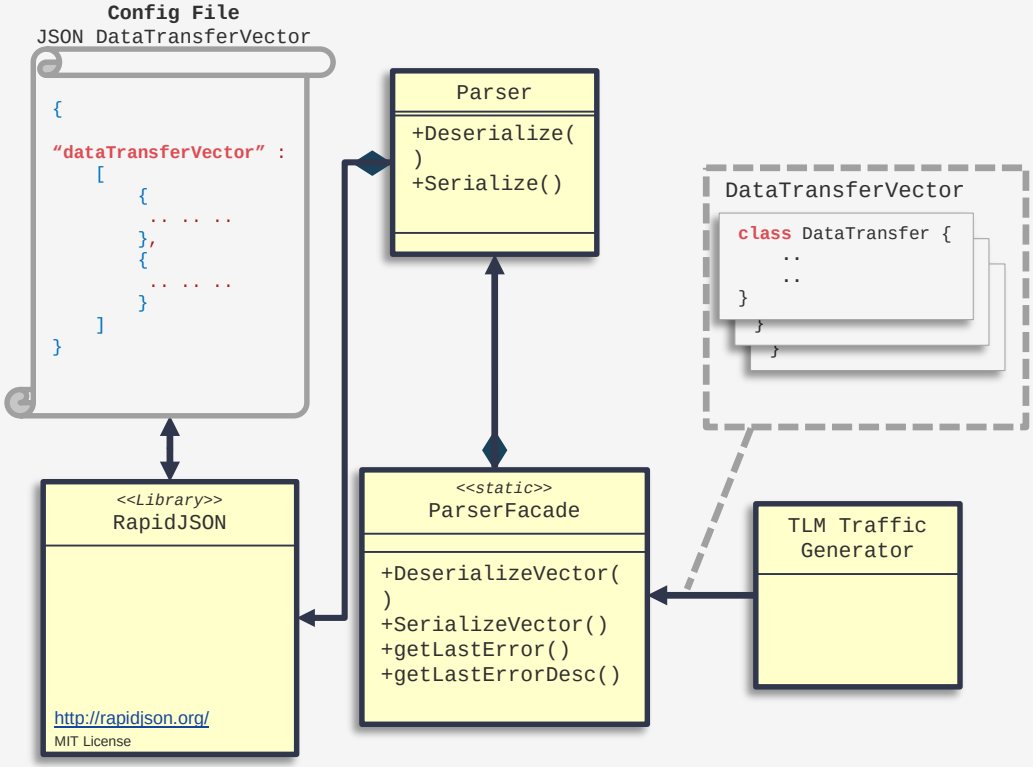
POSH TECH DETAIL – AXI4LITE PROTOCOL CHECKER



>> Key Architectural Points

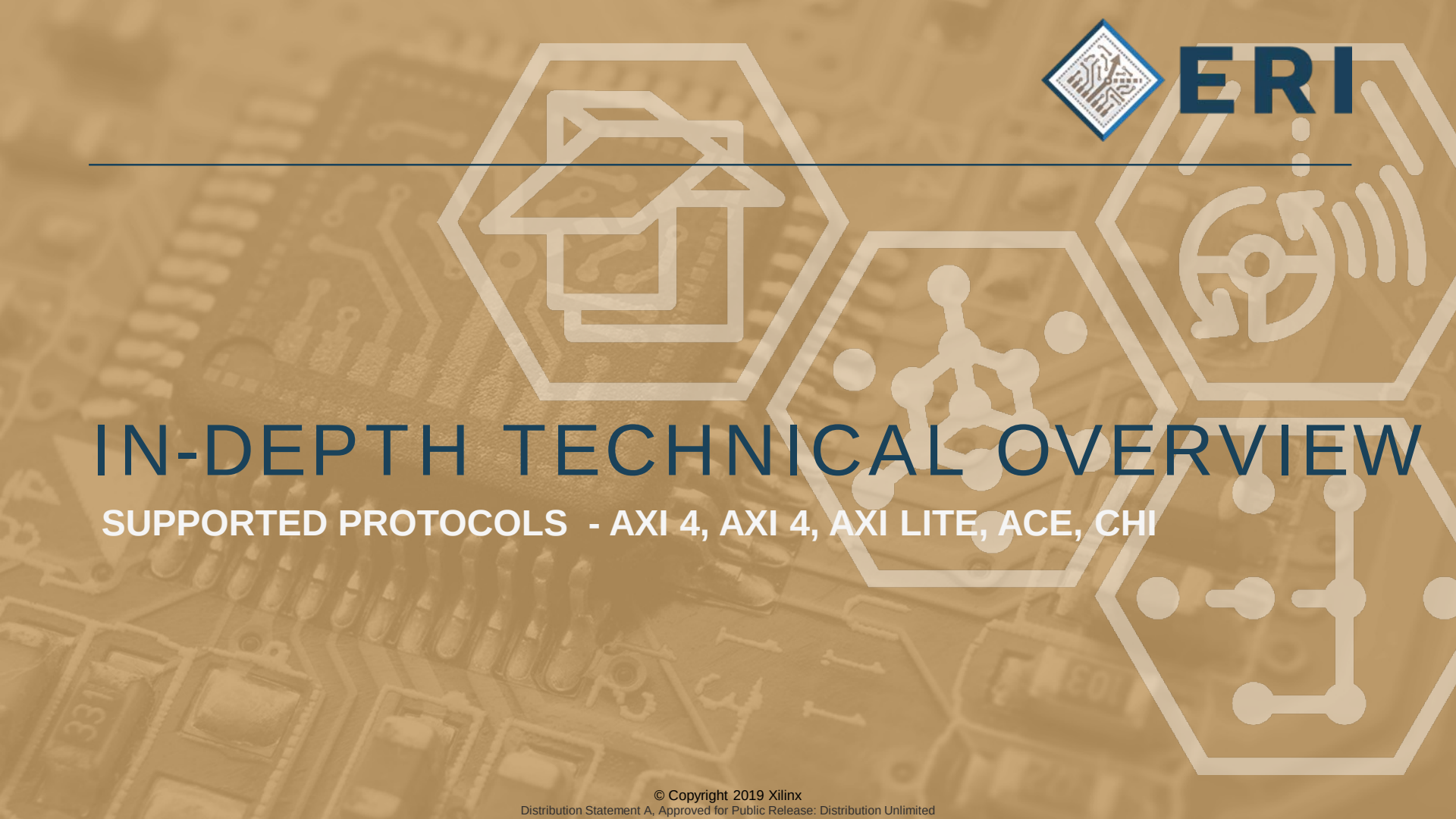
- Monitors and verifies AX4Lite signaling
- Consists of internal checkers
  - Each internal checker performs one or several checks on the AXI4Lite signaling
- A configuration enables and disables individual checks to be performed
- Reports errors through systemc's SC_REPORT_ERROR
- Examples of checks:
  - Check that signals are stable in any channel while valid is asserted but ready is low
  - Monitors and verifies transaction responses
  - Verifies that expected handshake signals are detected after receiving awvalid arvalid signals

POSH TECH DETAIL – CONFIG & PARSER



>> Key Architectural Points

- Optimized human readable/editable config file
- Atomic, Stateless, Fixed API Interface (Façade)
- Underlying Lib completely decoupled from client (Parser)
  - No client dependencies on rapidjson
- Supports deserialization from partial or full object descriptors
- Supports data randomization through annotations in JSON
- Fully automated test suite.

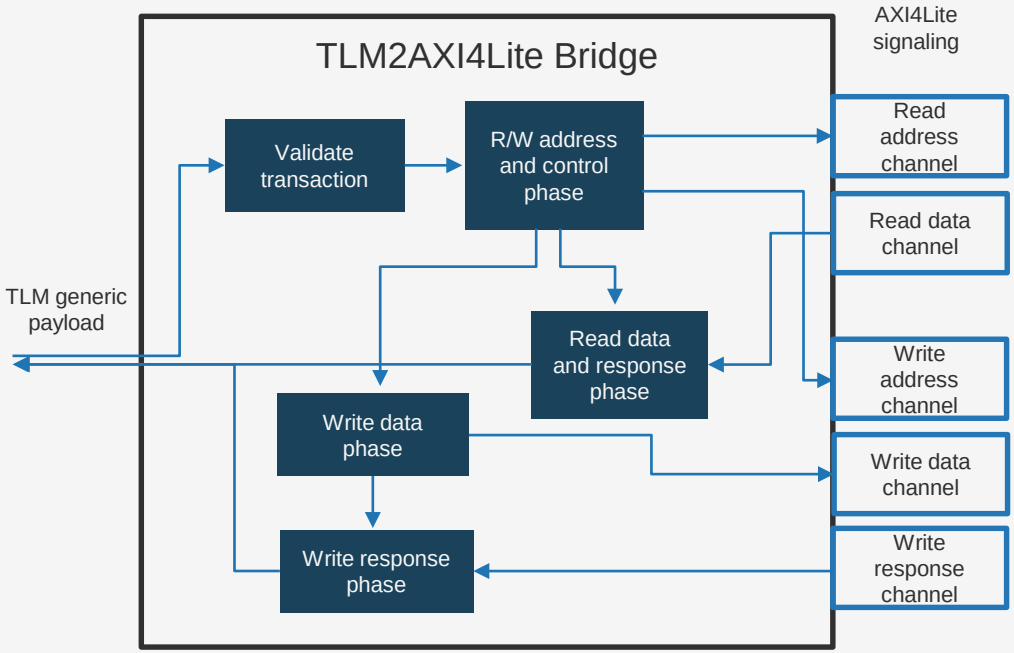


IN-DEPTH TECHNICAL OVERVIEW

SUPPORTED PROTOCOLS - AXI 4, AXI 4, AXI LITE, ACE, CHI

AXI4, AXI3, AXI LITE

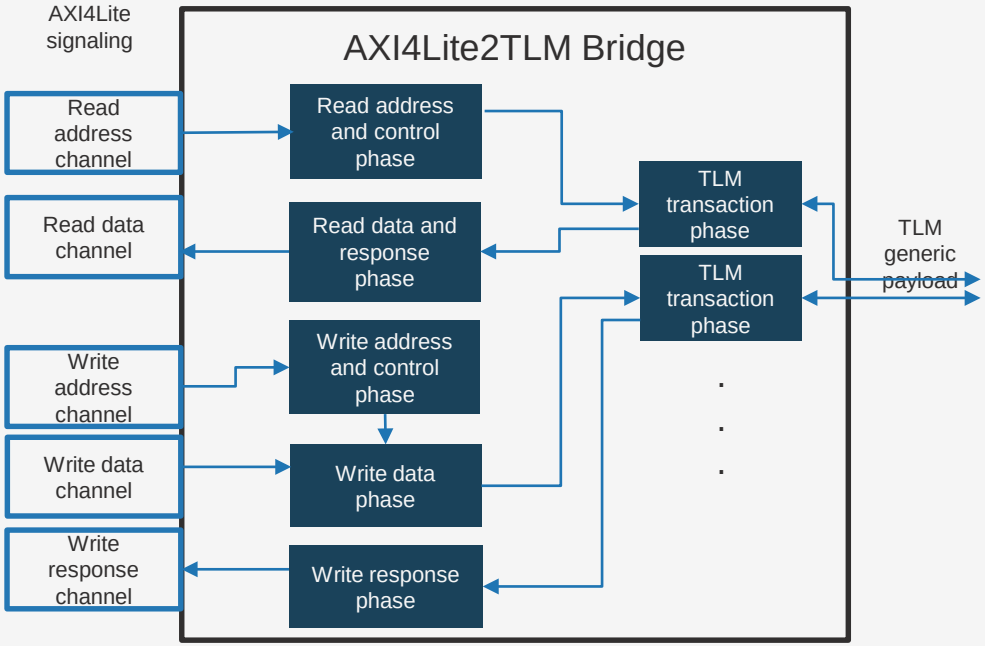
POSH TECH DETAIL – TLM 2.0 TO AXI4LITE BRIDGE



Key Architectural Points

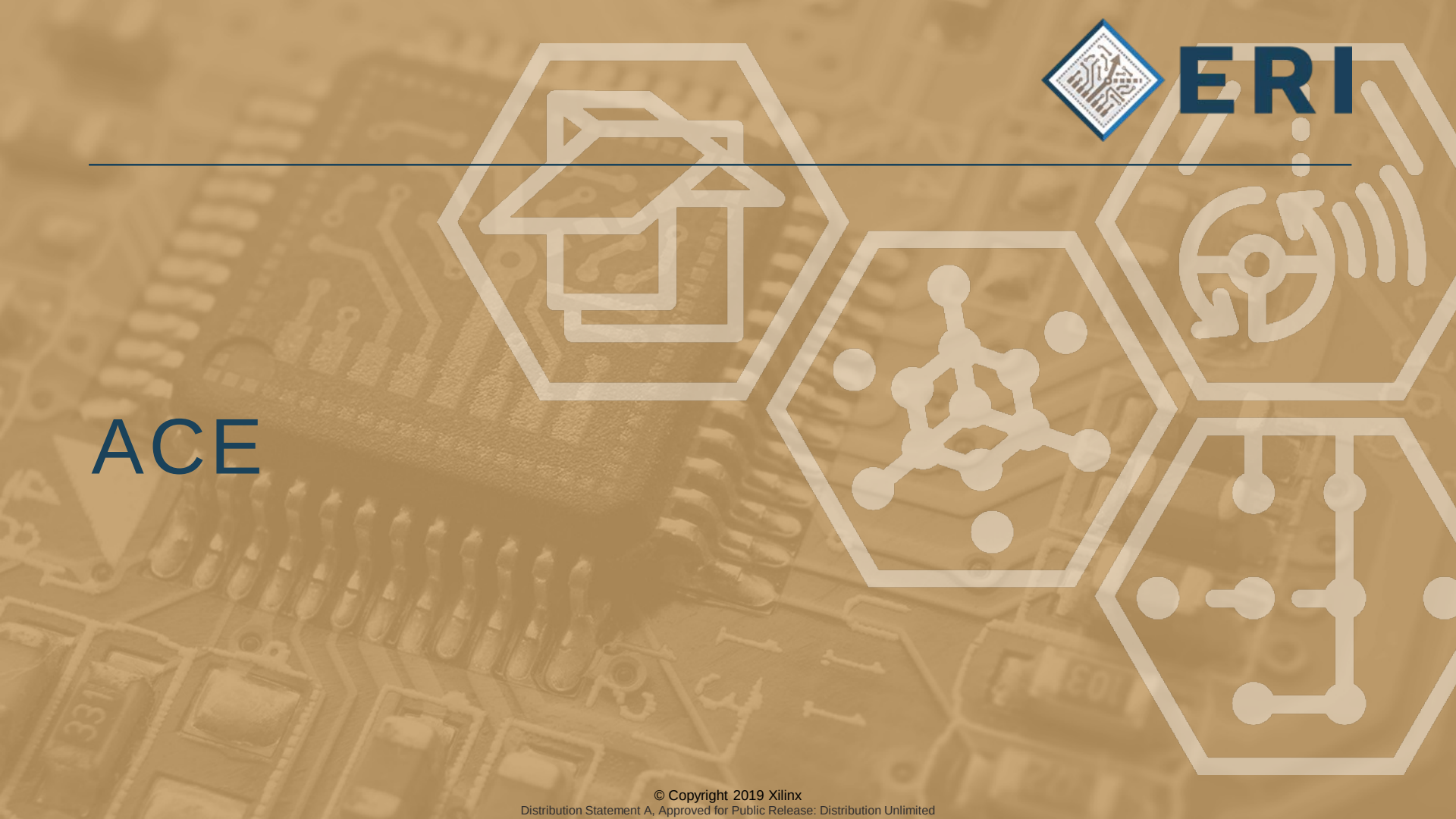
- Translates TLM generic payloads into AXI4Lite transactions
  - Large TLM transactions become multiple AXI4Lite transactions
- AXI4Lite Protocol specific details are carried through TLM generic payload extensions (for example AXI access permissions: secure, privileged)
- The phases of the transactions are run in parallel for supporting multiple outstanding AXI4Lite transactions
- An AXI4Lite traffic generator can be created by connecting the bridge to a TLM traffic generator (developed in the first deliverable)
- Examples of generated AXI4Lite transactions:
  - Narrow
  - Unaligned
  - With access permissions

POSH TECH DETAIL – AXI4LITE TO TLM 2.0 BRIDGE

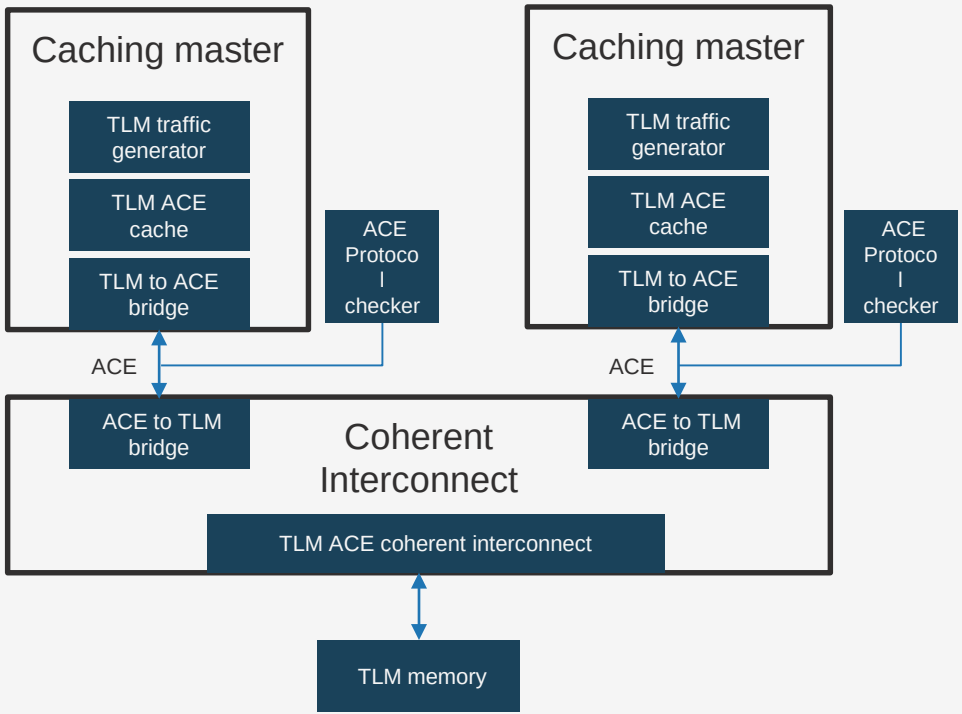


Key Architectural Points

- Translates AXI4Lite transactions into TLM generic payloads
- AXI4Lite Protocol specific details are carried through TLM generic payload extensions (for example AXI access permissions: secure, privileged)
- The phases of the transactions are run in parallel for supporting multiple outstanding AXI transactions
- Issues TLM generic payloads in order (since AXI4Lite does not support AXI IDs)
- Examples of translated AXI4Lite transactions:
  - Narrow
  - Unaligned
  - With access permissions



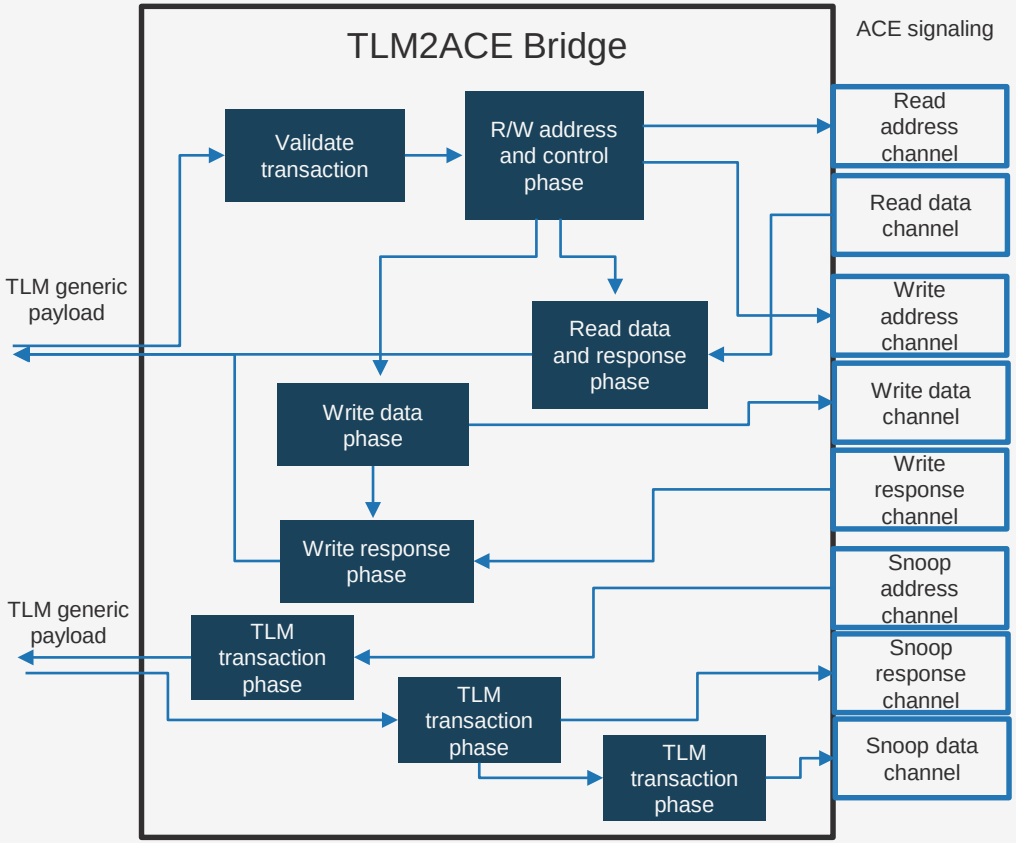
POSH TECH DETAIL – ACE TRAFFIC GENERATION



>> **Key Architectural Points**

- Caching masters generating ACE transactions
- An ACE coherent interconnect generating snoop transactions

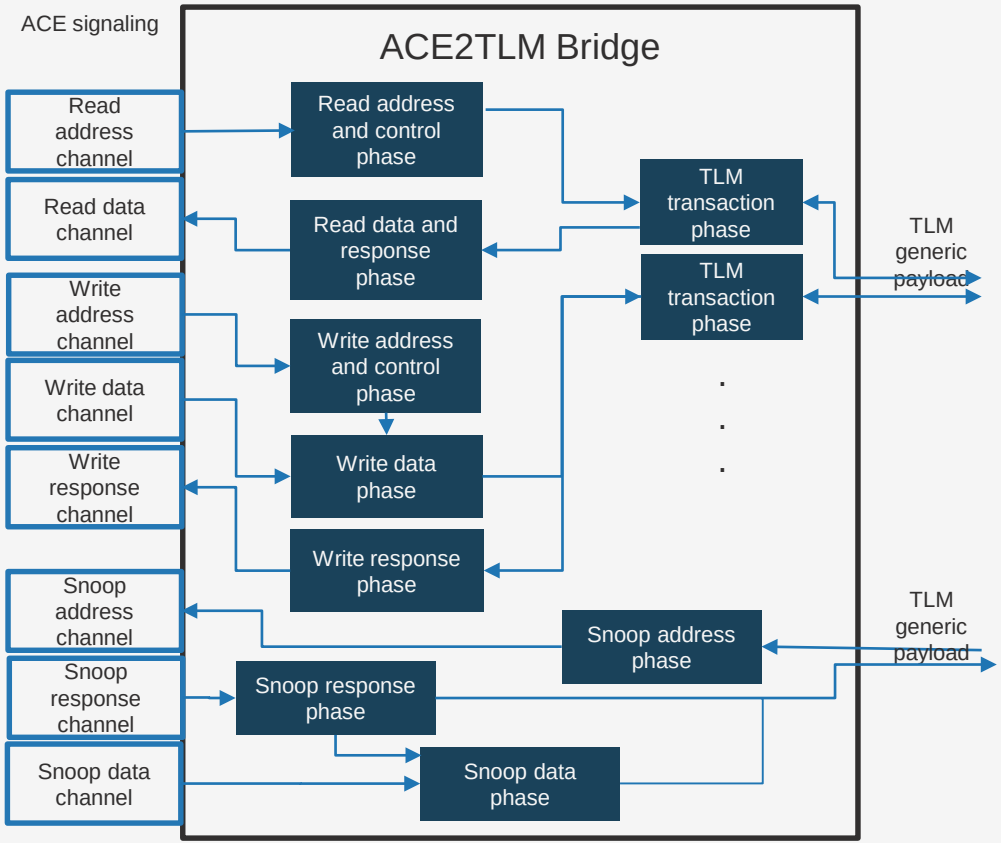
POSH TECH DETAIL – TLM 2.0 TO ACE BRIDGE



Key Architectural Points

- Translates TLM generic payloads into ACE transactions
- Translates ACE snoop transactions to TLM generic payloads
- ACE Protocol specific details are carried through TLM generic payload extensions (for example: snoop type, domain, barrier)
- The phases of the transactions are run in parallel for supporting multiple outstanding transactions
- Examples of translated ACE transactions:
  - Non snooping transactions
  - Coherent transactions
  - Memory update transactions
  - Cache maintenance transactions

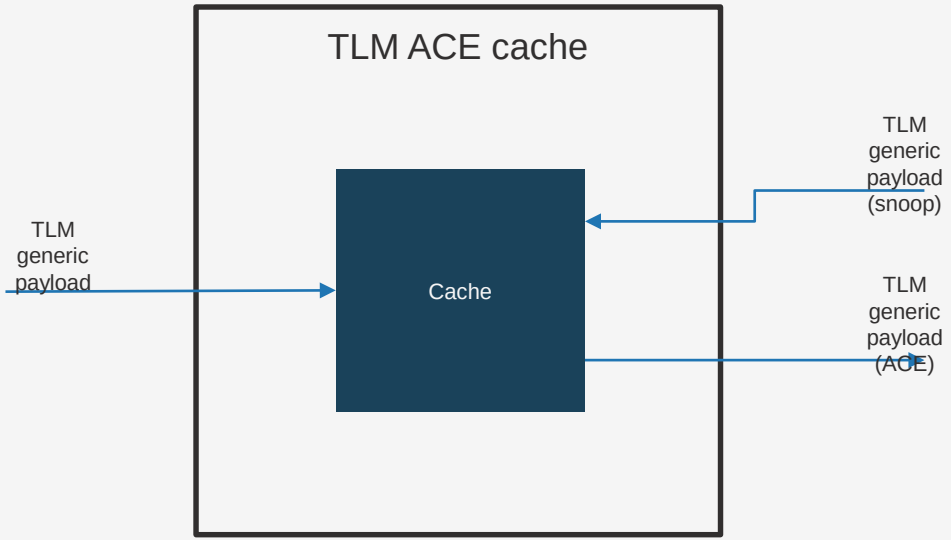
POSH TECH DETAIL – ACE TO TLM 2.0 BRIDGE



Key Architectural Points

- Translates ACE transactions into TLM generic payloads
- Translates TLM generic payloads into snoop transactions
- ACE Protocol specific details are carried through TLM generic payload extensions (for example: snoop type, domain, barrier)
- The phases of the transactions are run in parallel for supporting multiple outstanding transactions
- Examples of translated ACE transactions:
  - Non snooping transactions
  - Coherent transactions
  - Memory update transactions
  - Cache maintenance transactions

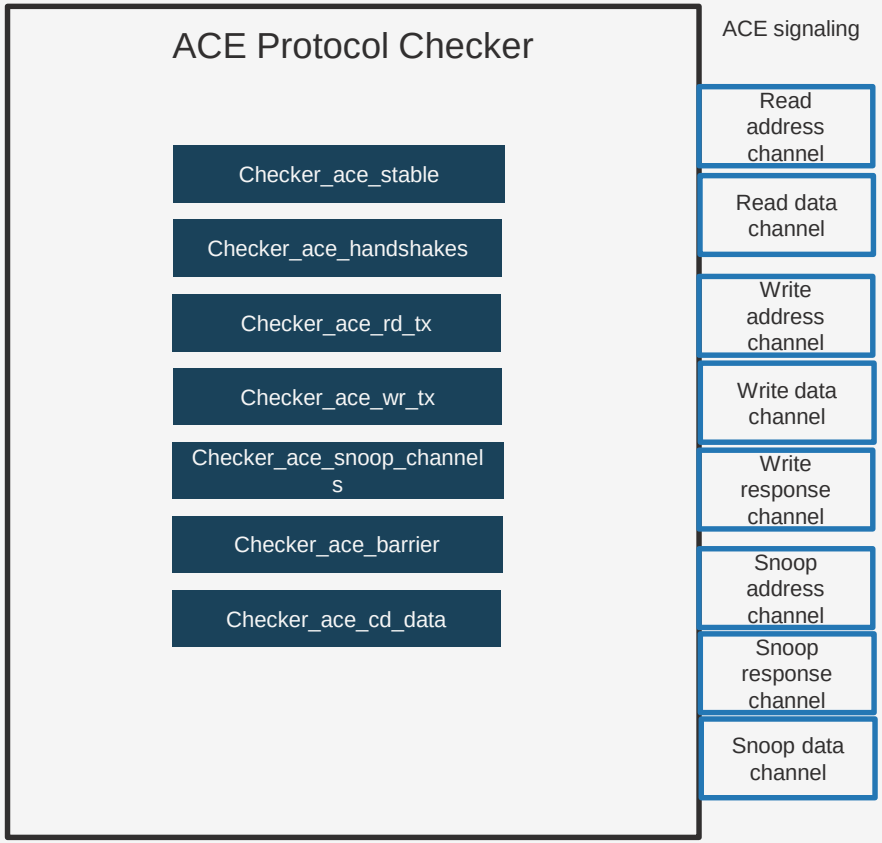
POSH TECH DETAIL – TLM ACE CACHE



Key Architectural Points

- Contains an internal cache with cache lines in one of the five states defined by the ACE protocol (invalid, unique clean, unique dirty, shared clean and shared dirty)
- Receives TLM generic payload from a master and generates ACE transactions if the required cache lines are not found/ or not in the correct state for serving the transaction
- Receives and acts on snoop transactions from an ACE interconnect

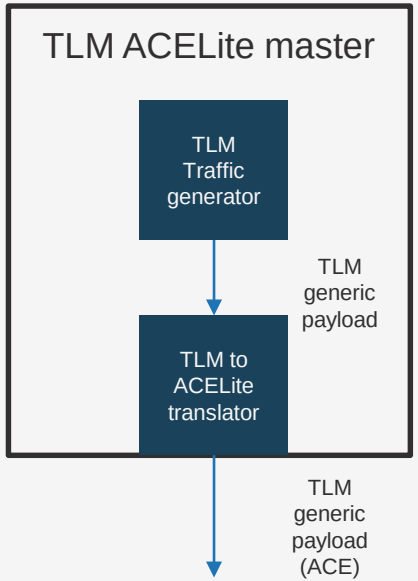
POSH TECH DETAIL – ACE PROTOCOL CHECKER



>> Key Architectural Points

- Monitors and verifies ACE signaling
- Consists of several internal checkers
  - Each internal checker performs one or several checks on the ACE signaling
- A configuration enables and disables individual checks to be performed
- Reports errors through systemc's SC_REPORT_ERROR
- Examples of checks:
  - Check that signals are stable in any channel while valid is asserted but ready is low
  - Check that rlast, rack, wlast and wack are generated correctly
  - Verifies signalling on the ar and r channels
  - Verifies signalling on the aw and w channels
  - Verifies signaling on the ac, cr and cd channels
  - Check responses on the r and b channel
  - Verifies that expected handshake signals are detected after receiving awvalid, arvalid and acvalid signals

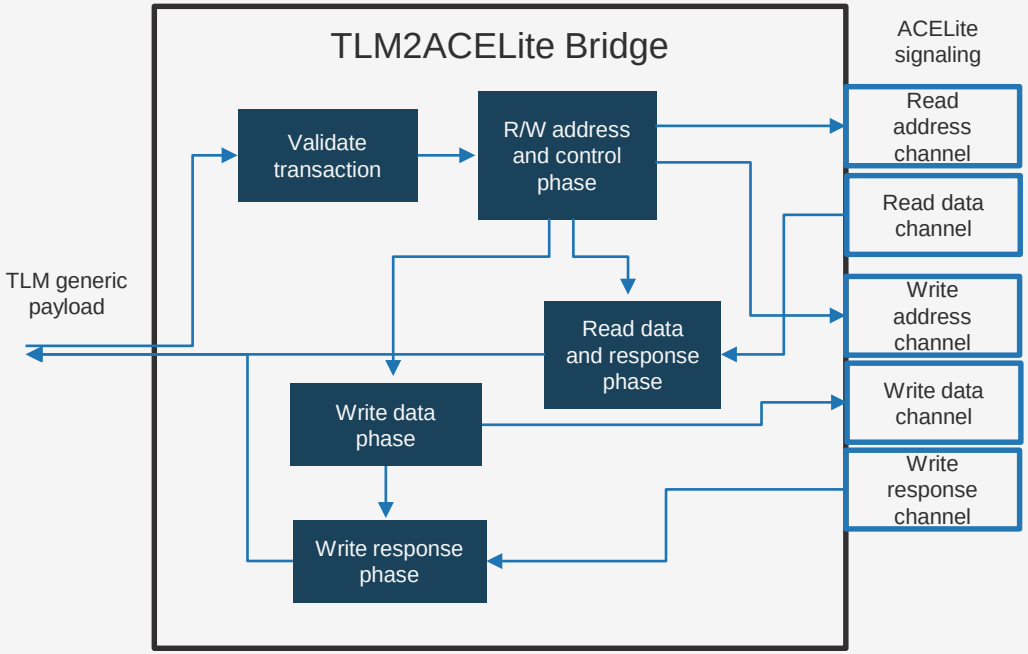
POSH TECH DETAIL – TLM ACELITE MASTER



>> Key Architectural Points

- Contains a TLM traffic generator generating TLM generic payloads based on traffic descriptions
- Contains a TLM to ACELite translator translating the TLM generic payloads to ACELite transactions described in TLM (a generic payload with a generic extension containing ACELite attributes)
- Example of ACELite transactions being generated:
  - WriteUnique
  - ReadOnce

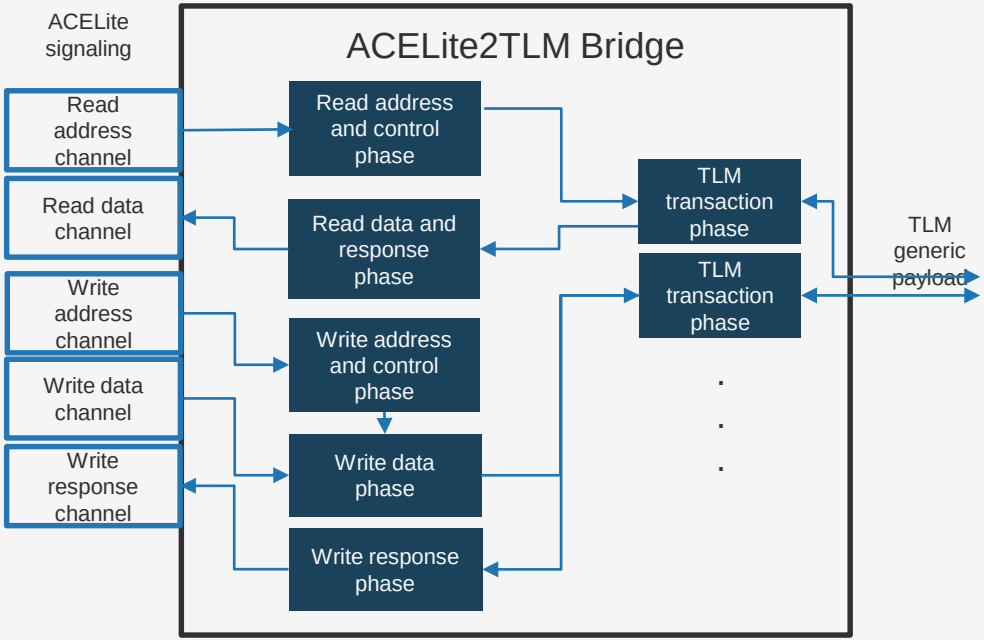
POSH TECH DETAIL – TLM 2.0 TO ACELITE BRIDGE



Key Architectural Points

- Translates TLM generic payloads into ACELite transactions
- ACE Protocol specific details are carried through TLM generic payload extensions (for example: snoop type, domain, barrier)
- The phases of the transactions are run in parallel for supporting multiple outstanding transactions
- Examples of translated ACELite transactions:
  - Non snooping transactions
  - Coherent transactions
  - Cache maintenance transactions

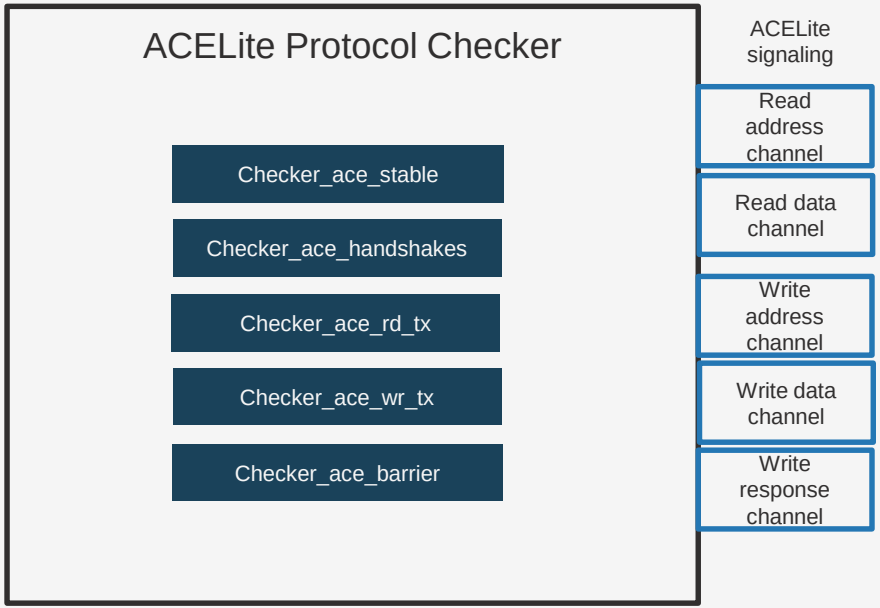
POSH TECH DETAIL – ACE TO TLM 2.0 BRIDGE



Key Architectural Points

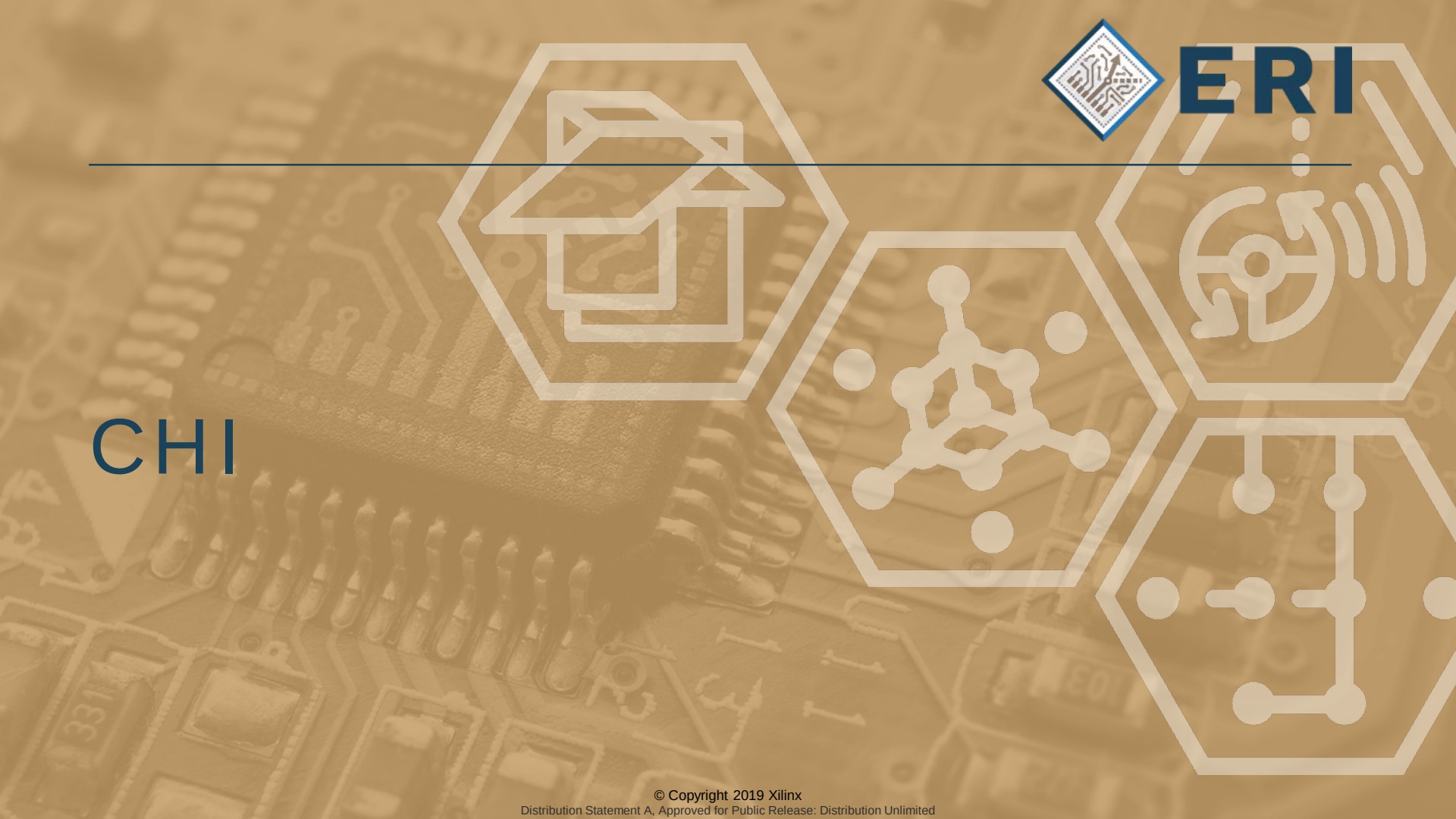
- Translates ACELite transactions into TLM generic payloads
- ACE Protocol specific details are carried through TLM generic payload extensions (for example: snoop type, domain, barrier)
- The phases of the transactions are run in parallel for supporting multiple outstanding transactions
- Examples of translated ACELite transactions:
  - Non snooping transactions
  - Coherent transactions
  - Cache maintenance transactions

POSH TECH DETAIL – ACELITE PROTOCOL CHECKER

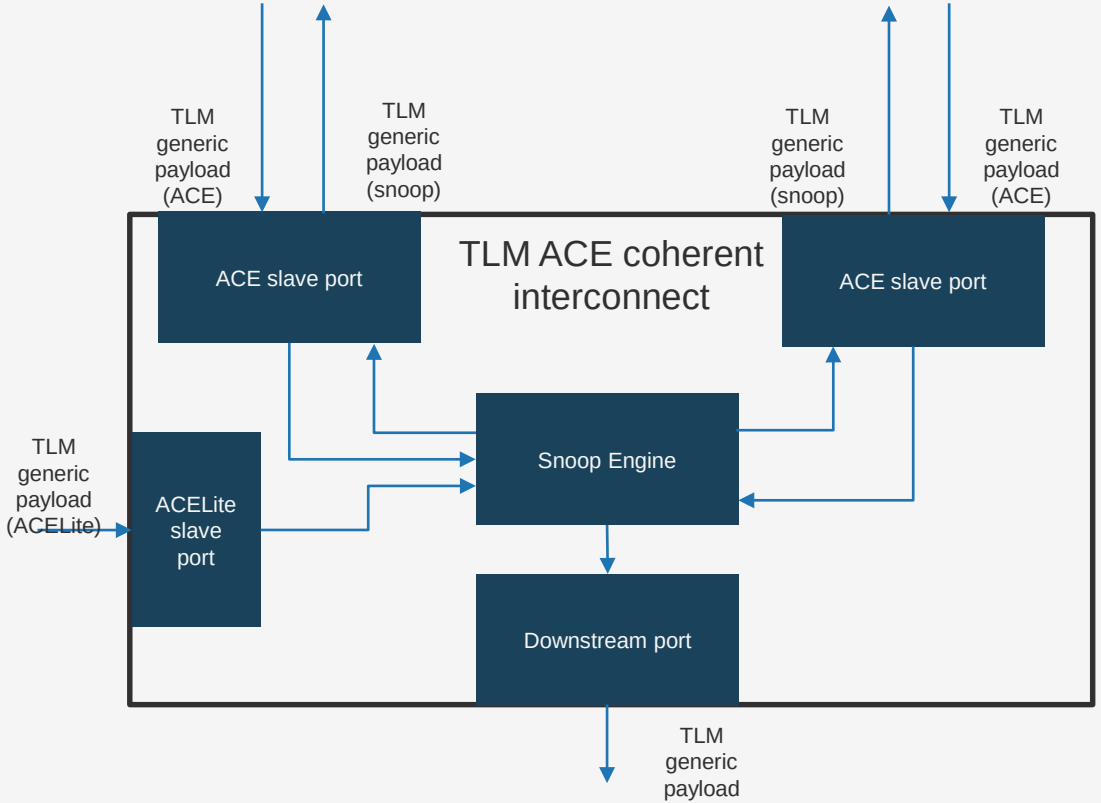


>> Key Architectural Points

- Monitors and verifies ACELite signaling
- Consists of several internal checkers
  - Each internal checker performs one or several checks on the ACE signaling
- A configuration enables and disables individual checks to be performed
- Reports errors through systemc's SC_REPORT_ERROR
- Examples of checks:
  - Check that signals are stable in any channel while valid is asserted but ready is low
  - Verifies signalling on the ar and r channels
  - Verifies signalling on the aw and w channels
  - Checks responses on the r and b channel
  - Verifies that expected handshake signals are detected after receiving awvalid, and arvalid signals



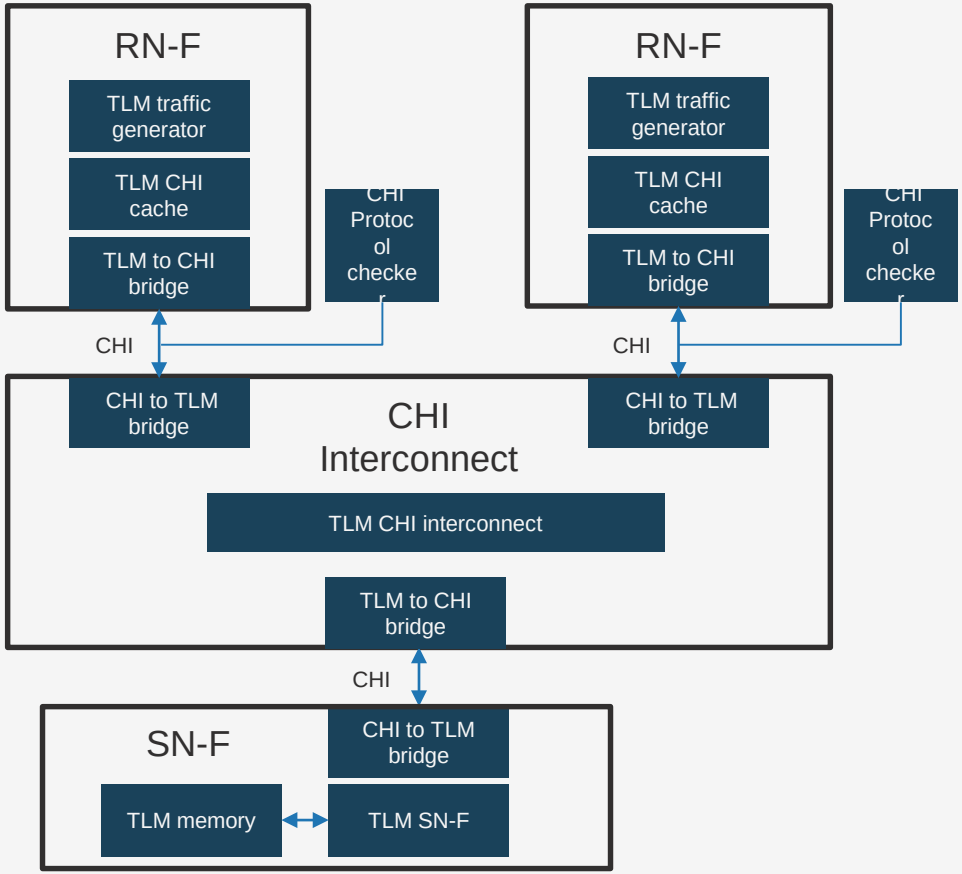
POSH TECH DETAIL – TLM CHI INTERCONNECT



Key Architectural Points

- Consists of ACE and ACELite slave ports, a snoop engine and a downstream port
- The ACE slave ports receive TLM generic payloads with ACE transactions and forwards them to the snoop engine
- ACELite slave ports receives TLM generic payloads with ACELite transactions and forwards them to the snoop engine
- The snoop engine handles snooping of masters (through the ACE slave ports). If it is not a snooping transaction it is passed directly to the downstream port

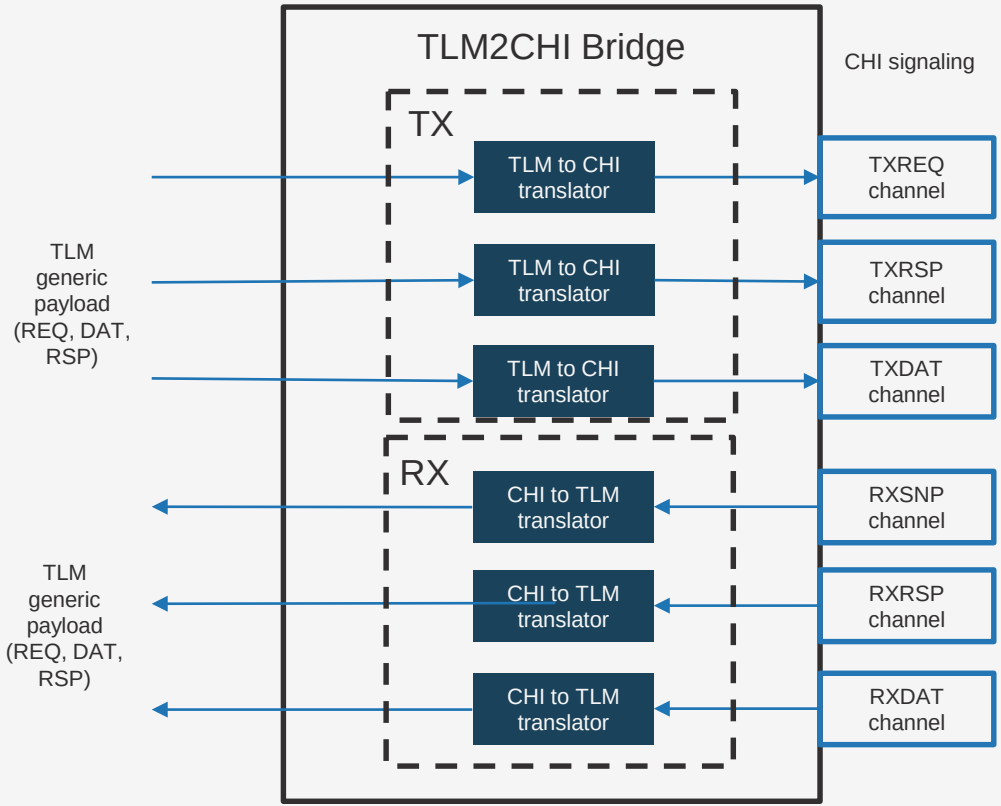
POSH TECH DETAIL – CHI TRAFFIC GENERATION



>> **Key Architectural Points**

- Request nodes (RN-F) generating CHI transactions
- An CHI interconnect (HN-F, MN) generating downstream and snoop transactions

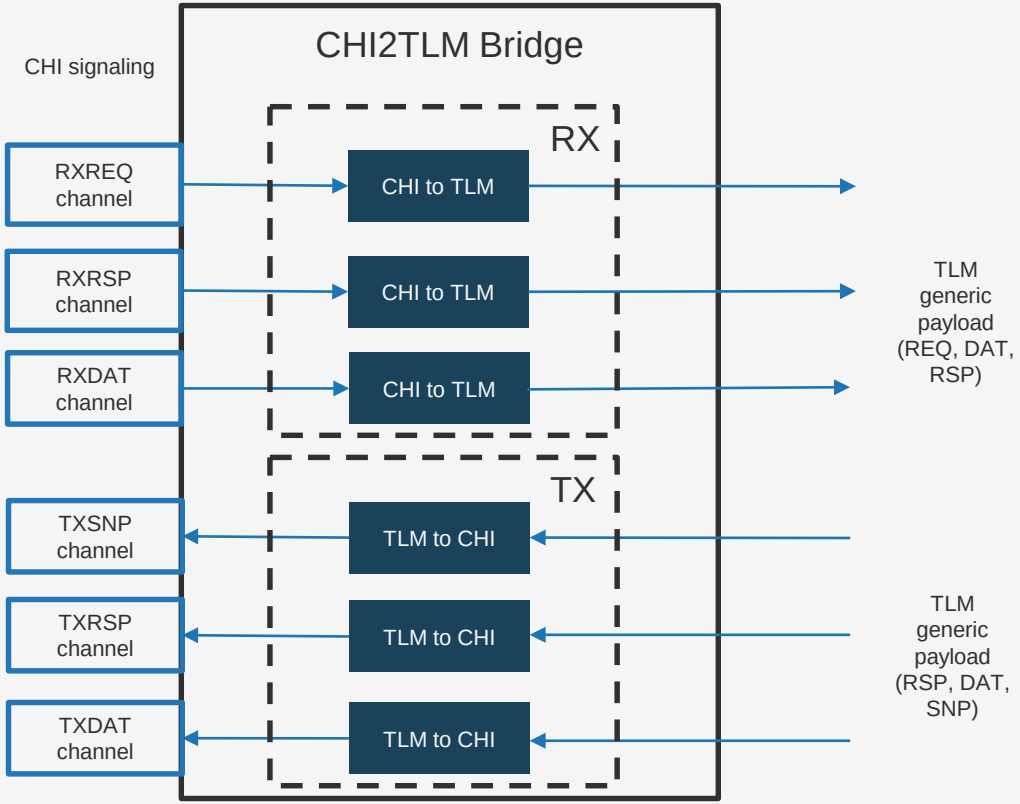
POSH TECH DETAIL – TLM 2.0 CHI BRIDGE



Key Architectural Points

- Translates TLM generic payloads into CHI flits on the outbound link
- Translates CHI flits into TLM generic payloads on the inbound link
- CHI Protocol specific details are carried through a TLM generic payload extensions (for example: TgtID, SrcID, Opcode)
- Translators contain queues for supporting multiple outstanding transactions
- Examples of translated CHI flits on the outbound link:
  - Read, dataless, write, atomic requests
  - Data flits containing data for write transactions or snoop requests
  - Requester response flits (for example carrying CompAck)
- Examples of translated CHI flits on the inbound link:
  - Snoop request flits
  - Data flits for read transactions
  - Completer response flits (for example carrying Comp, CompDBIDResp)

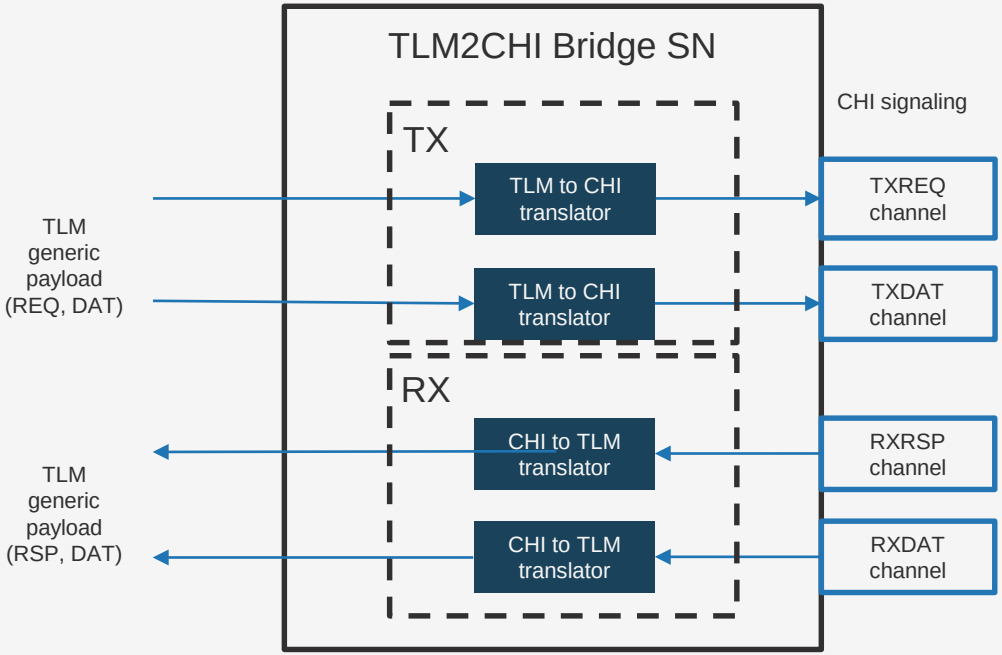
POSH TECH DETAIL – CHI TLM 2.0 BRIDGE



Key Architectural Points

- Translates CHI flits into TLM generic payloads on the inbound link
- Translates TLM generic payloads into CHI flits on the outbound link
- CHI Protocol specific details are carried through a TLM generic payload extensions (for example: TgtID, SrcID, Opcode)
- Translators contain queues for supporting multiple outstanding transactions
- Examples of translated CHI flits on the outbound link:
  - Snoop request flits
  - Data flits for read transactions
  - Completer response flits (for example carrying Comp, CompDBIDResp)
- Examples of translated CHI flits on the inbound link:
  - Read, dataless, write, atomic requests
  - Data flits containing data for write transactions or snoop requests
  - Requester response flits (for example carrying CompAck)

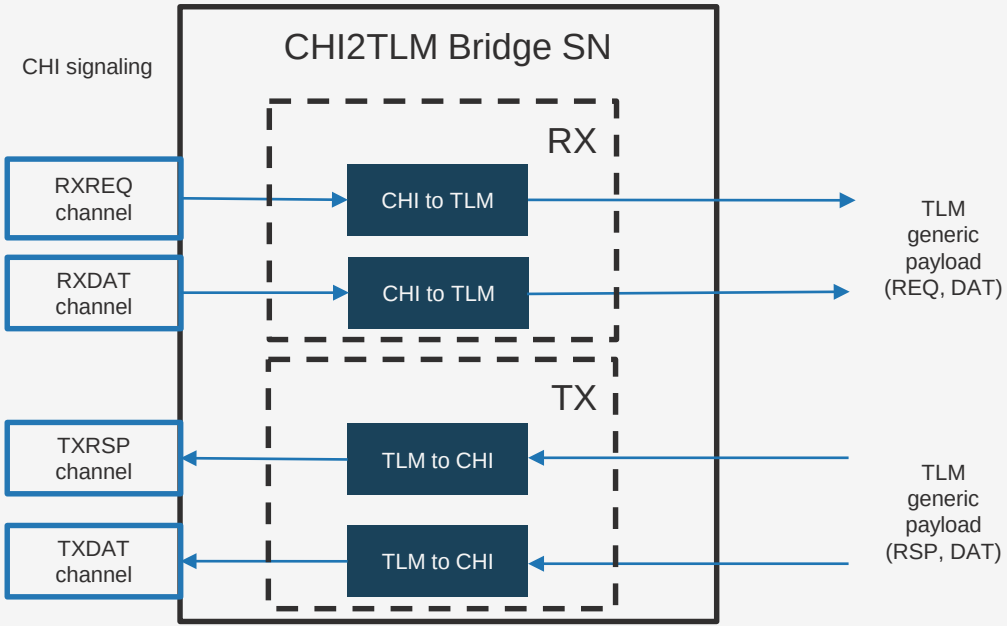
POSH TECH DETAIL – TLM 2.0 CHI BRIDGE SN



Key Architectural Points

- Translates TLM generic payloads into CHI flits on the outbound link
- Translates CHI flits into TLM generic payloads on the inbound link
- CHI Protocol specific details are carried through a TLM generic payload extensions (for example: TgtID, SrcID, Opcode)
- Translators contain queues for supporting multiple outstanding transactions
- Examples of translated CHI flits on the outbound link:
  - Read, dataless, write, atomic requests
  - Data flits containing data for write transactions
- Examples of translated CHI flits on the inbound link:
  - Data flits for read transactions
  - Completer response flits (for example carrying CompDBIDResp)

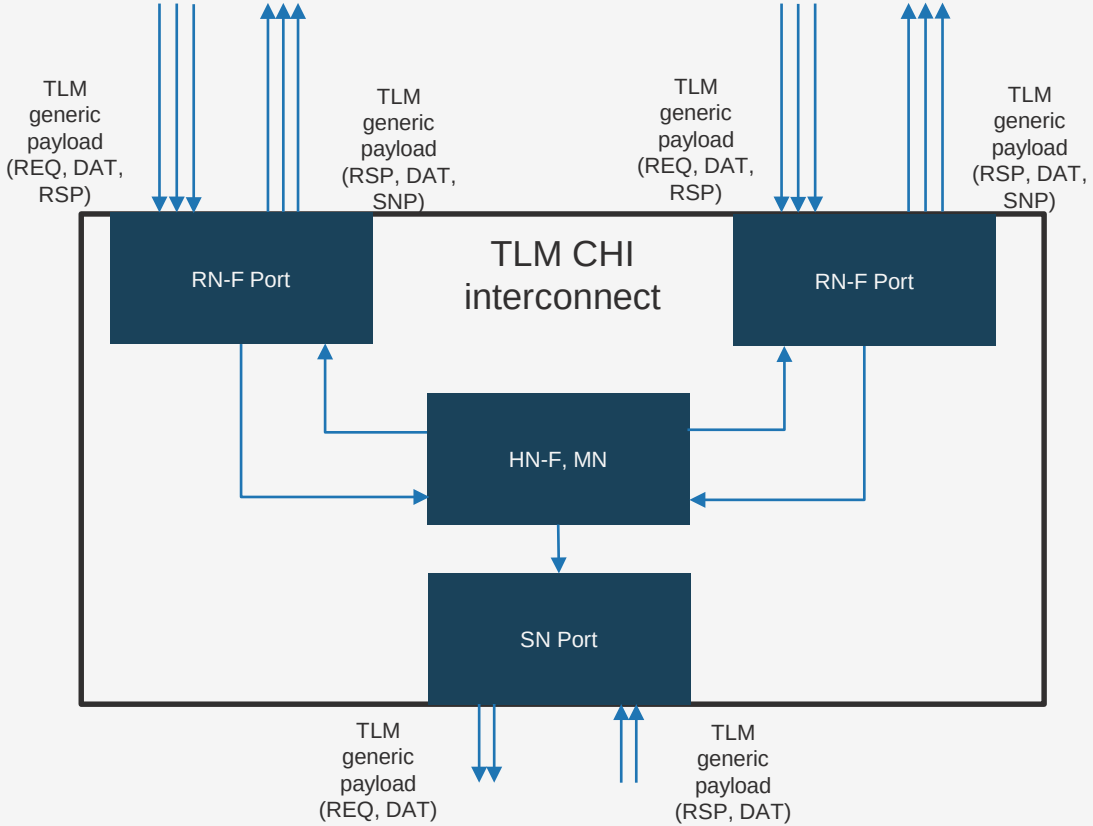
POSH TECH DETAIL – CHI TLM 2.0 BRIDGE SN



Key Architectural Points

- Translates CHI flits into TLM generic payloads on the inbound link
- Translates TLM generic payloads into CHI flits on the outbound link
- CHI Protocol specific details are carried through a TLM generic payload extensions (for example: TgtID, SrcID, Opcode)
- Translators contain queues for supporting multiple outstanding transactions
- Examples of translated CHI flits on the inbound link:
  - Read, dataless, write, atomic requests
  - Data flits containing data for write transactions
- Examples of translated CHI flits on the outbound link:
  - Data flits containing data for read transactions
  - Completer response flits (for example carrying CompDBIDResp)

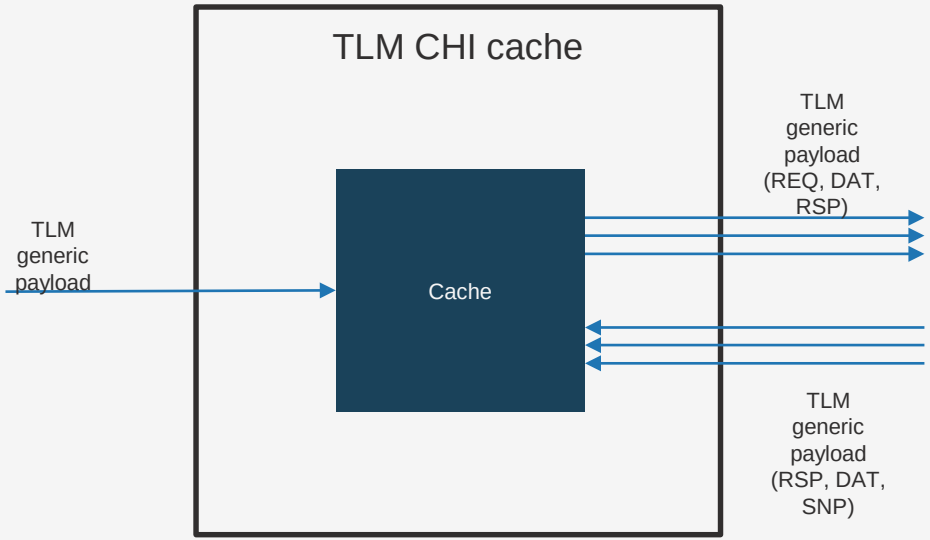
POSH TECH DETAIL – TLM CHI INTERCONNECT



>> **Key Architectural Points**

- Consists of RN-F ports, an SN port, a HN-F and an MN
- The RN-F ports receive TLM generic payloads describing CHI request, dat and response flits and forwards them to the HN-F (DVM transactions are forwarded to the MN)
- The HN-F snoops the required RN-Fs (through the RN-F ports). When required the HN-F issues a transaction towards the Slave Node (through the SN port)
- DVM transactions received on the RN-F ports are forwarded to and processed by the MN

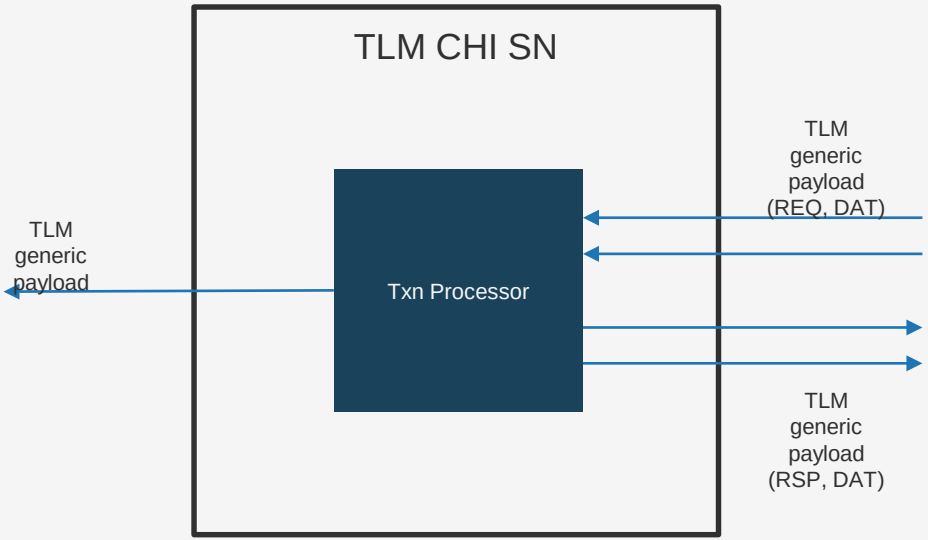
POSH TECH DETAIL – TLM CHI CACHE



Key Architectural Points

- Contains an internal cache with cache lines in one of the states defined by the CHI protocol (invalid, unique clean, unique clean empty, unique dirty, unique dirty partial, shared clean and shared dirty)
- Receives TLM generic payloads and generates CHI transactions (TLM generic payloads containing an CHI attributes extension) if the required cache lines are not found / or not in the correct state for serving the transaction
- Receives and acts on snoop transactions from an CHI interconnect

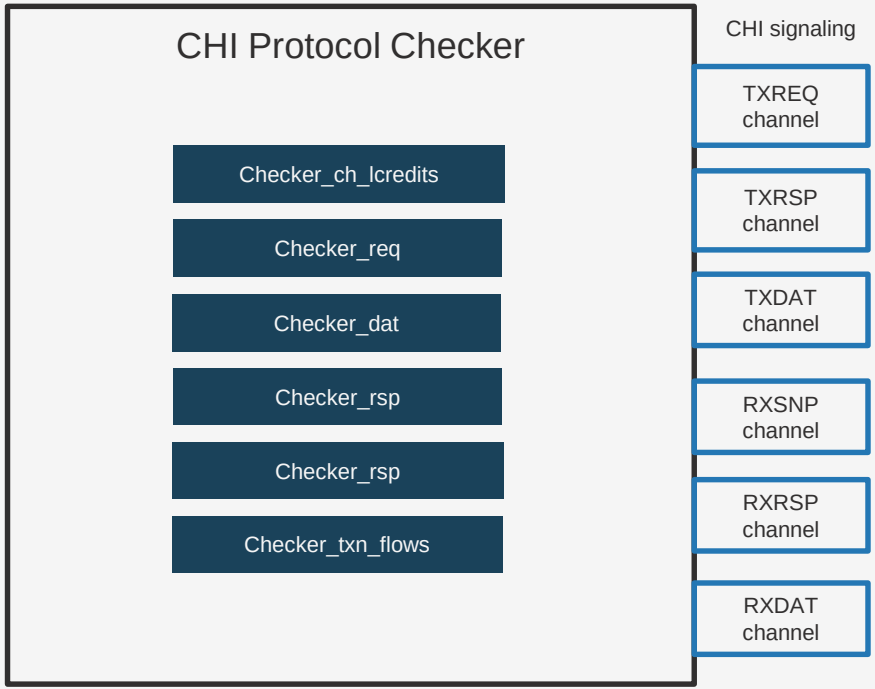
POSH TECH DETAIL – TLM CHI SN



Key Architectural Points

- Receives TLM generic payloads describing CHI requests from an ICN
- Operates as an SN-F on CHI request and generates TLM generic payloads towards a connected TLM module (memory) and or generic payloads towards the ICN describing CHI dat / responses as required by the request
- Example of processed CHI requests:
  - Non-snoopable read, write and atomic requests

POSH TECH DETAIL – CHI PROTOCOL CHECKER



>> Key Architectural Points

- Monitors and verifies CHI signaling
- Consists of several internal checkers
  - Each internal checker performs one or several checks on the CHI signaling
- A configuration enables and disables individual checks to be performed
- Reports errors through systemc's SC_REPORT_ERROR
- Examples of checks:
  - Checks outstanding link credits on channels
  - Checks request, data, response and snoop flits
  - Checks transactions flows

ERI ELECTRONICS RESURGENCE INITIATIVE

SUMMIT

2019 | DETROIT, MI | JULY 15-17

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.