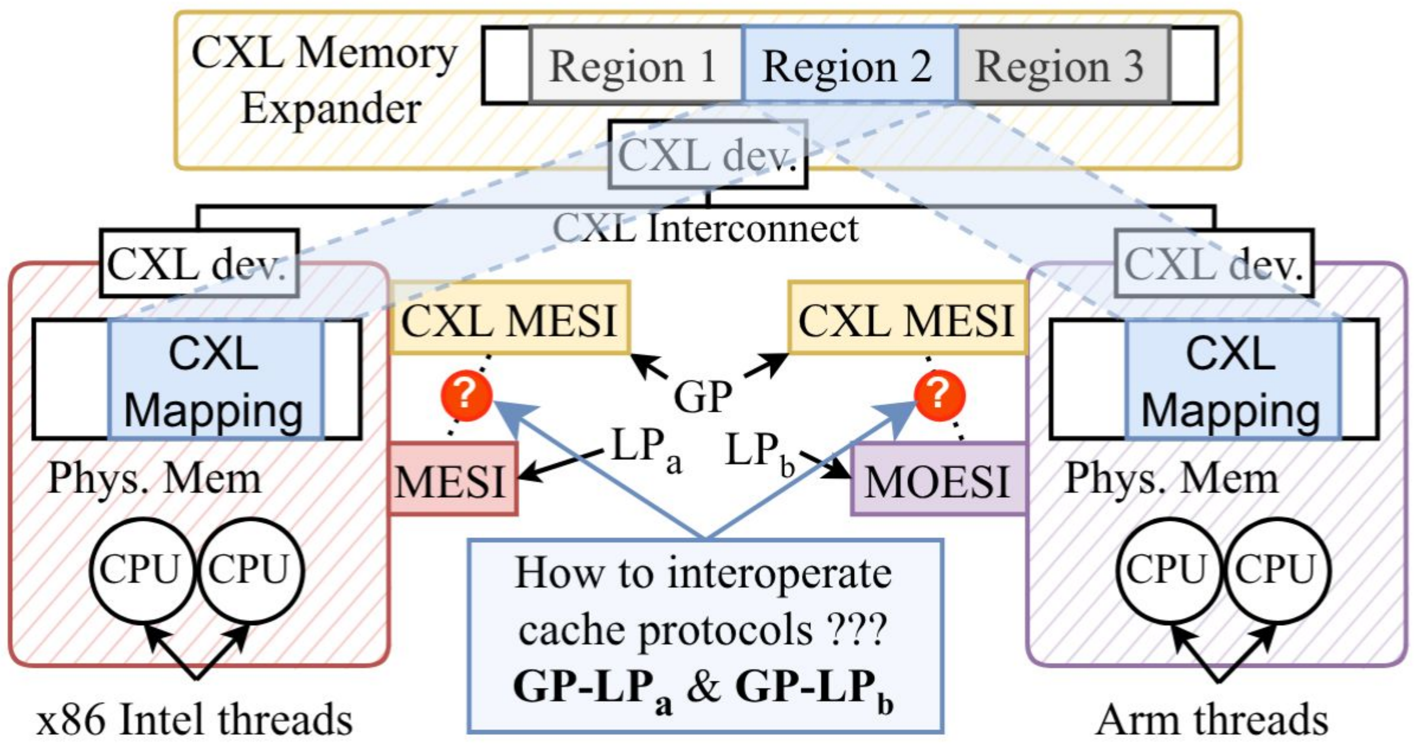
CXL-Bridge: CXL controllers for heterogeneous memory architectures

Priority Programme "Disruptive Memory Technologies" (SPP 2377)

PI: Prof. Dr.-Ing Pramod Bhatotia (TU Munich)

Deutsche Forschungsgemeinschaft

Motivation



CXL specifications lack **safety guarantees** for **heterogeneous** & **multi-host** shared memory

Problem statement

How to systematically integrate CXL distributed coherence into heterogeneous hardware architectures?

CXL Bridges

Push-button ***synthesis*** and ***verification*** of heterogeneous CXL coherence controllers

Performance

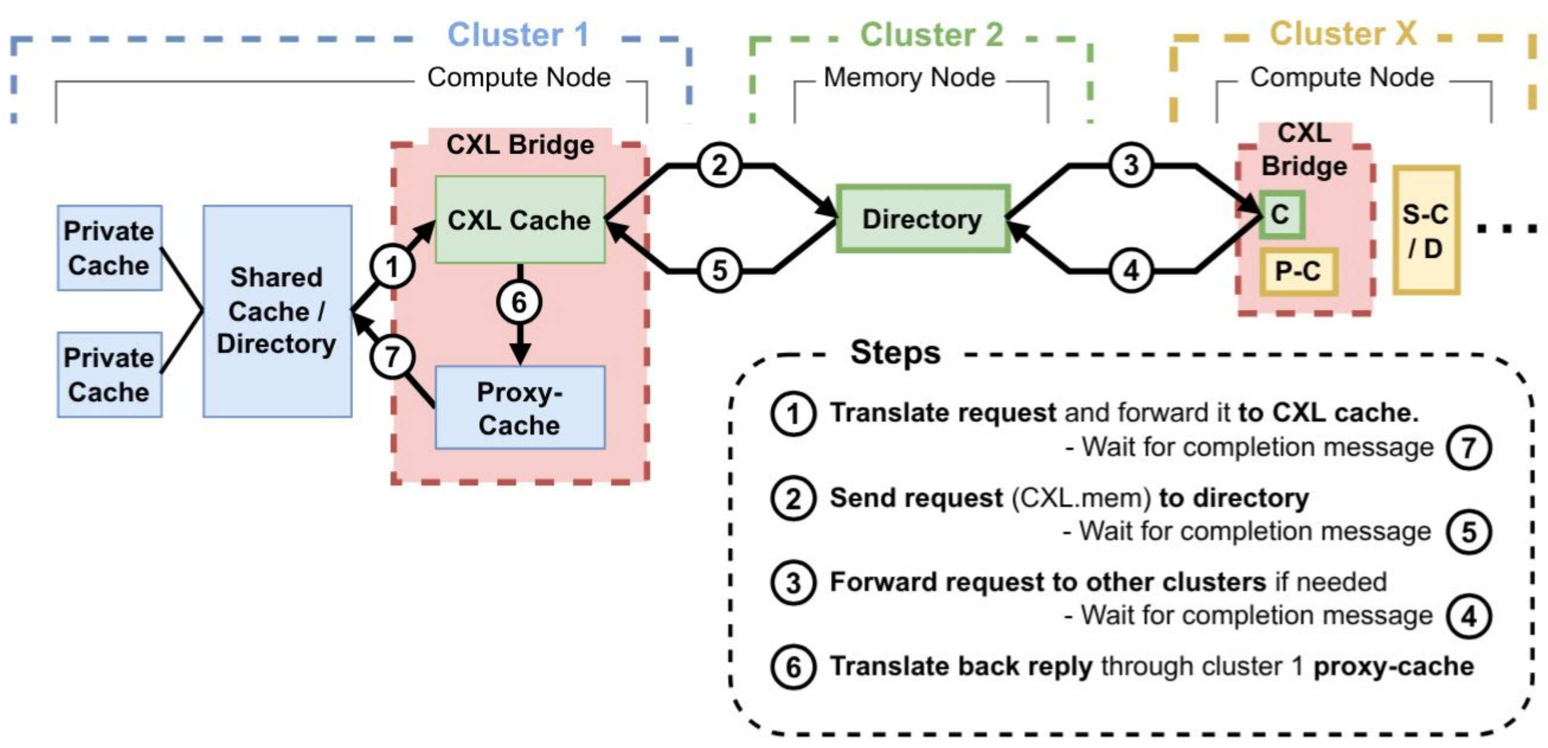
No coherence overheads
- Traffic (bandwidth)
- Network delays (latency)

Correctness

Formal verification
- Automated
- Correctness criteria

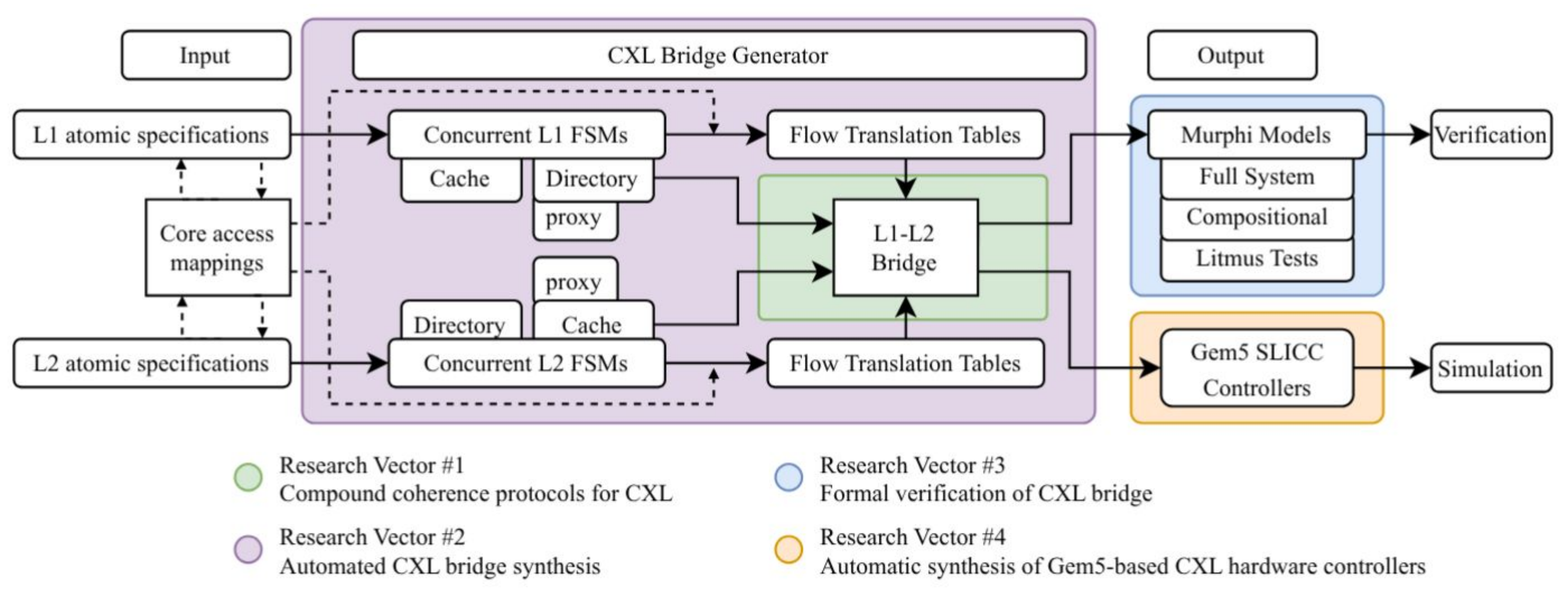
CXL Bridge Controllers

- **Approach:** Non-intrusively interoperate local (host-specific) protocols with `CXL.mem` global protocol
- **Problem:** Semantic gap between host protocols (requests/responses) and CXL specifications
- **Solution:** A new CXL Bridge abstraction
- **Key Ideas:** an "interface" controller
  - translate and propagate coherence transactions between protocols
  - dedicated controller with new state machine built from protocol specs



Automated Synthesis of CXL Bridge Controller Synthesis

- **Approach:** Universal methodology to construct CXL bridges
- **Problem:** Complexity of protocol compounding (multiplicative #states/#transitions), need to evolve fast with new specs
- **Solution:** Specification-driven synthesis of CXL Bridge
- **Key Ideas:**
  - Input machine readable specifications of both protocols at the interface
  - Derive bridge state machines from the protocol IRs
  - Output verification and simulation models with 2 codegen backends



CXL Heterogeneous Memory Consistency

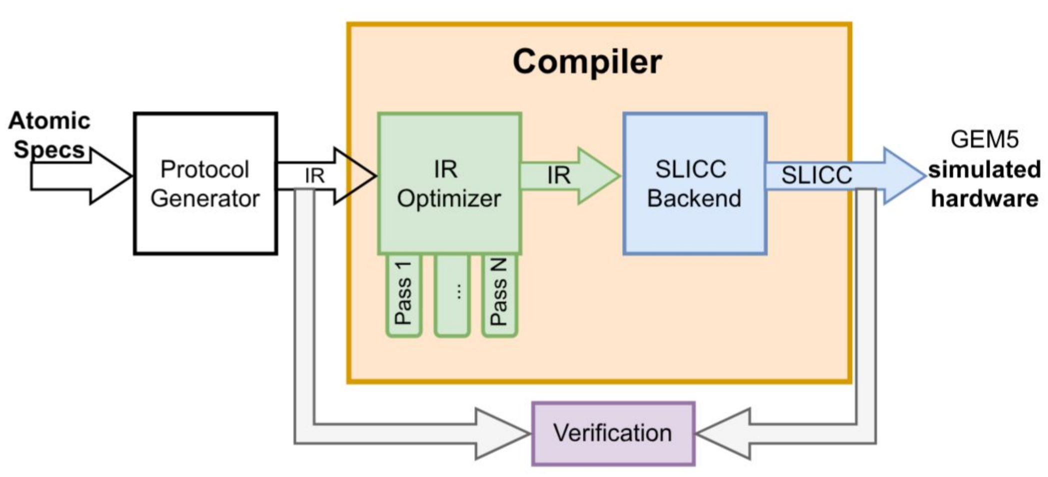
- **Approach:** A correctness criteria for CXL heterogeneous shared memory
- **Problem:** Formally define the Memory Consistency Model (MCM) of CXL bridges
  - *"What happens when concurrent threads access the same memory with different memory models? e.g., TSO & ARM"*
- **Solution:** Formally show that CXL bridges realize *Compound Memory Models*
  - Every node can view and use the shared memory as if it were its local (native) MCM
  - No code change (source or compiled)

T_1 (TSO)	T_2 (RC)	T_1 (RC)	T_2 (TSO)
st_1: `st #1` X;	ld_1: `ldar` r_1 Y;	st_1: `st #1` X;	ld_1: `ld` r_1 Y;
			m_1: `mfence`;
st_2: `st #1` Y;	ld_2: `ld` r_2 X;	st_2: `stlr #1` Y;	ld_2: `ld` r_2 X;
TSO producer, RC consumer		RC producer, TSO consumer	

Heterogeneous MP litmus test. In both, $st1 \to st2$ & $ld1 \to ld2$, thus, the outcome $\{r1 = 0, r2 = 1\}$ is disallowed.

Performance assessment: Gem5-based CXL Bridges

- **Approach:** Gem5-based simulations to assess performance of CXL bridges
- **Problem:** inflexible development of cache coherence controllers in Gem5 (SLICC DSL)
- **Solution:** A compilation framework to generate SLICC code from CXL Bridge IR



Formal Verification of Coherence Protocols

- **Verification properties** - from any state
  - Safety: no compound MCM violation (litmus tests)
  - Liveness: core eventually get any requested access
- **Approach:** Explicit state model checking
- **Problem:** state space explosion (OoM)
- **Solution:** compositional system models