中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
鹏城实验室
Peng Cheng Laboratory
中国开放指令生态（RISC-V）联盟
China RISC-V Alliance

TL-Test: Cache Coherence Verification Framework for XiangShan

王凯帆 蔺嘉炜 张传奇

2022/8/26

大纲

- 处理器缓存验证相关背景
- 已有工作基础
- TL-Test：基于 TileLink 总线的多场景缓存验证框架
- TL-Test 在香山处理器中的验证场景
- 未来展望与开放问题

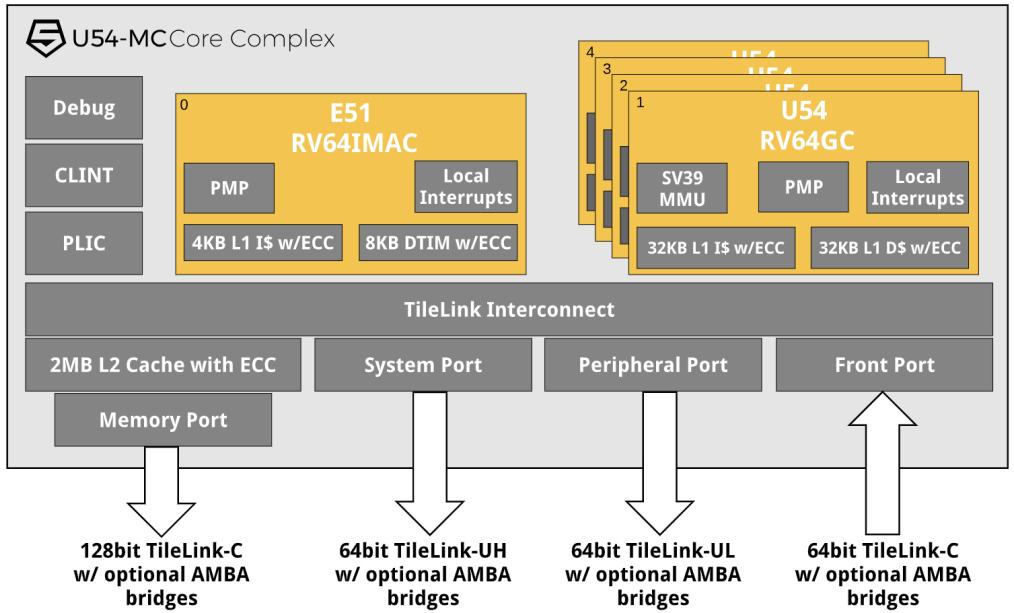
大纲

- **处理器缓存验证相关背景**
- 已有工作基础
- TL-Test：基于 TileLink 总线的多场景缓存验证框架
- TL-Test 在香山处理器中的验证场景
- 未来展望与开放问题

趋势：RISC-V 平台缓存验证需求强烈

- **RISC-V 计算平台复杂度升级**
  - 多核、多级缓存、异构
  - 芯片设计进入 SoC 时代

- **缓存系统的复杂度不断攀升**
  - 高并发度的硬件实现、复杂的一致性协议
  - 拥有最多 Corner Case 的部件之一

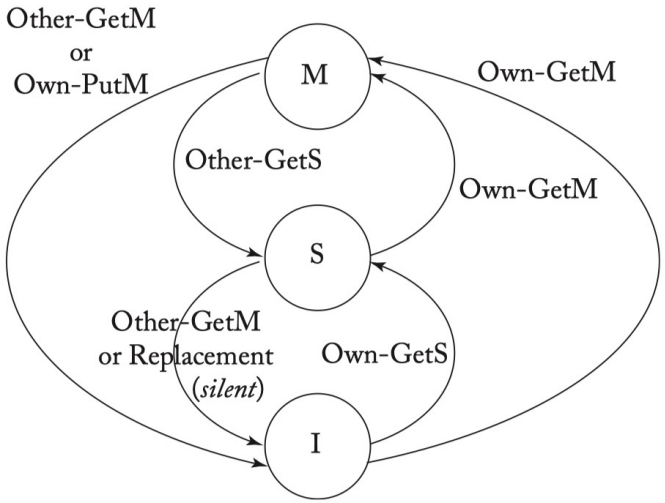


示例：U54-MC Core Complex 架构图

➢ **缓存一致性验证问题越来越重要**

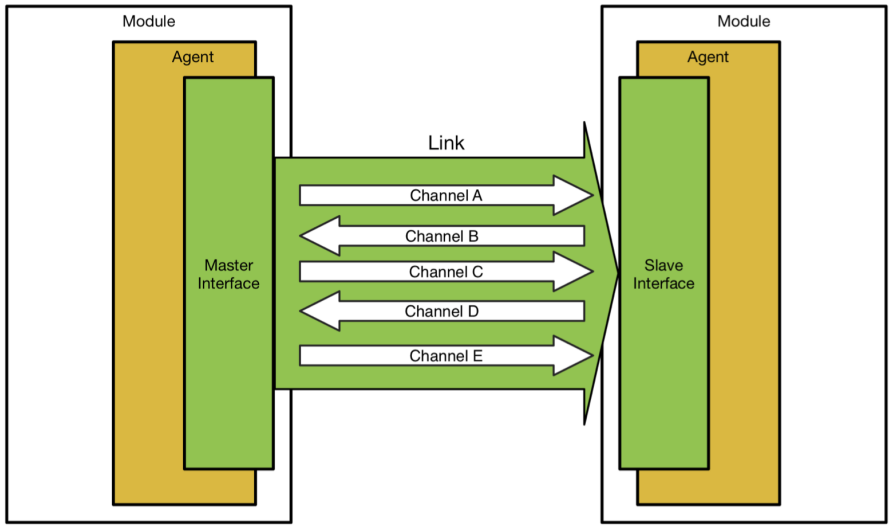
缓存一致性基础：协议&总线

- **一致性协议**
  - **定义单个缓存块的权限状态与转移关系**
  - MSI，MESI，Dragon，etc.
  - 从协议层面保证共享数据的一致性



一致性协议示例：MSI

- **互联总线**
  - **定义各种缓存拓扑结构下，实现跨层信息传输与状态转换的标准**
  - ACE，CHI，TileLink，etc.
  - 总线标准本身的正确性与完备性可由形式化方法验证

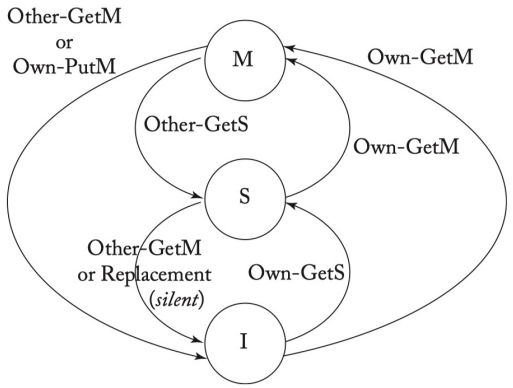


互联总线示例：TileLink

缓存验证的内容

- **有了完备的一致性协议和互联总线就足够了嘛？**
  - Probably Not!

➢ 逻辑状态转移图 VS 实现状态转移表（以 MSI 为例）



States	Processor Core Events			Bus Events						
				Own Transaction				Transactions for Other Cores		
	Load	Store	Replacement	Own-GetS	Own-GetM	Own-PutM	Data	Other-GetS	Other-GetM	Other-PutM
I	Issue GetS /IS^D	Issue GetM /IM^D								
IS^D	Stall Load	Stall Store	Stall Evict				Copy data into cache, load hit /S	(A)	(A)	(A)
IM^D	Stall Load	Stall Store	Stall Evict				Copy data into cache, store hit /M	(A)	(A)	(A)
S	Load hit	Issue GetM /SM^D	-/I						-/I	
SM^D	Load hit	Stall Store	Stall Evict				Copy data into cache, store hit /M	(A)	(A)	(A)
M	Load hit	Store Hit	Issue PutM, send Data to memory /I					Send Data to req and memory /S	Send Data to req /I	

图源：《A Primer on Memory Consistency and Cache Coherence》

- **实际缓存验证还要解决的问题**
  - 缓存的硬件实现与状态转移表一致
  - 缓存对外的接口是否严格满足互联总线标准

……

大纲

- 处理器缓存验证相关背景
- **已有工作基础**
- TL-Test：基于 TileLink 总线的多场景缓存验证框架
- TL-Test 在香山处理器中的验证场景
- 未来展望与开放问题

（回顾）Agent Faker：TL-C一致性Cache的软件测试框架

张传奇 @第一届 RISC-V 中国峰会

针对支持一致性操作的TL-C Cache模块，进行**单元级**测试验证

测试要点：

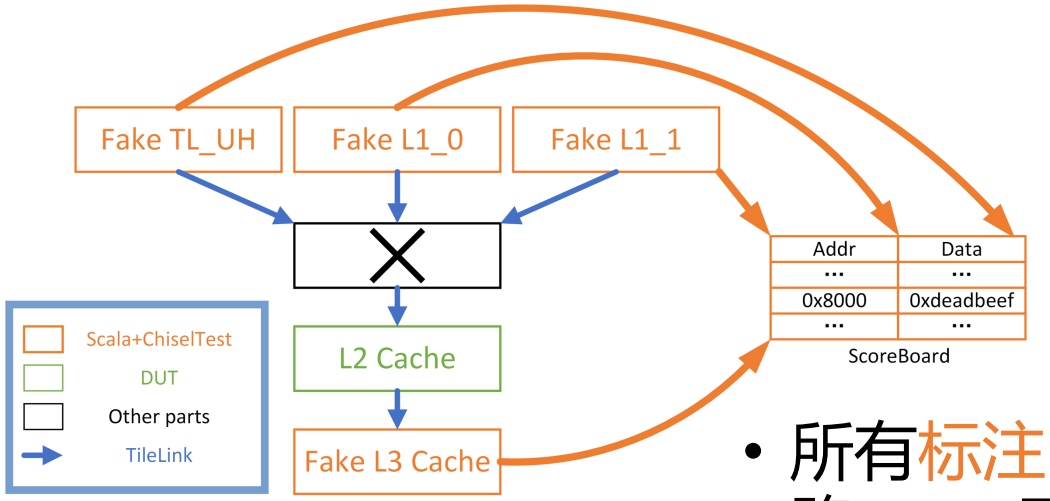
- 满足 Cache 的数据一致性
- 满足 TileLink 规定的总线行为规范

利用 ChiselTest 进行测试框架的搭建

- 控制 Cache 模块单元的IO
- 随机生成符合 TileLink 总线协议的请求
- 对收到的信号解析后进行数据校验

（回顾）Agent Faker：TL-C一致性Cache的软件测试框架

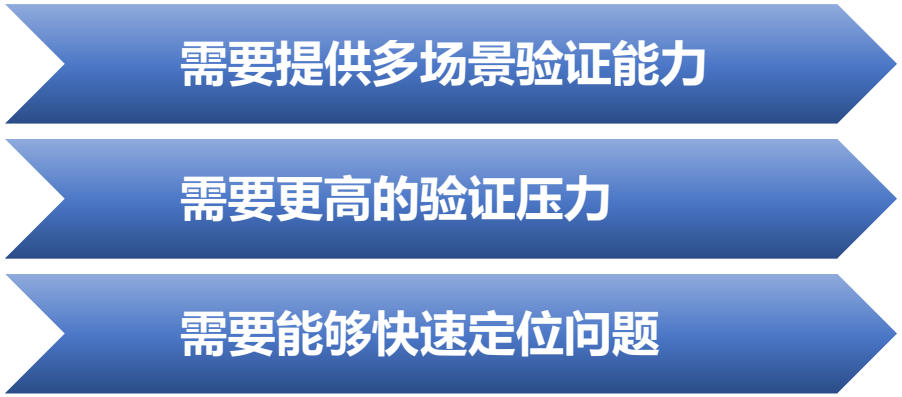
张传奇 @第一届 RISC-V 中国峰会



- 所有标注Fake的模块，都是通过ChiselTest库来进行电路(DUT+已充分验证的辅助模块)IO的操纵，同时通过Scala编写的软件模拟TileLink Agent的各种行为
- 随机生成地址池，从这个固定范围中选取地址进行激励，这样可以更好地校验一致性情景
- 在Scala维护一个计分板记录全局正确数据

香山南湖架构验证的新要求

- **香山南湖架构首次以双核进行流片，使用了新开发的 Non-inclusive Cache**
  - 对多核的验证要求提高



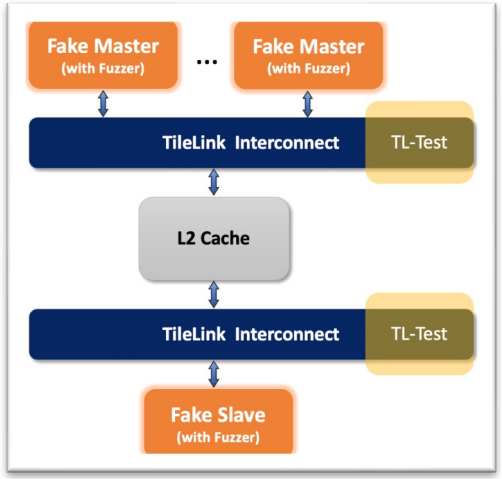
- **Agent Faker 暴露出局限性**
  - 测试场景单一，仅针对单元测试
  - 基于 ChiselTest 开发，运行速度缓慢，测试压力小
  - 出错修复代价高，Debug 困难

大纲

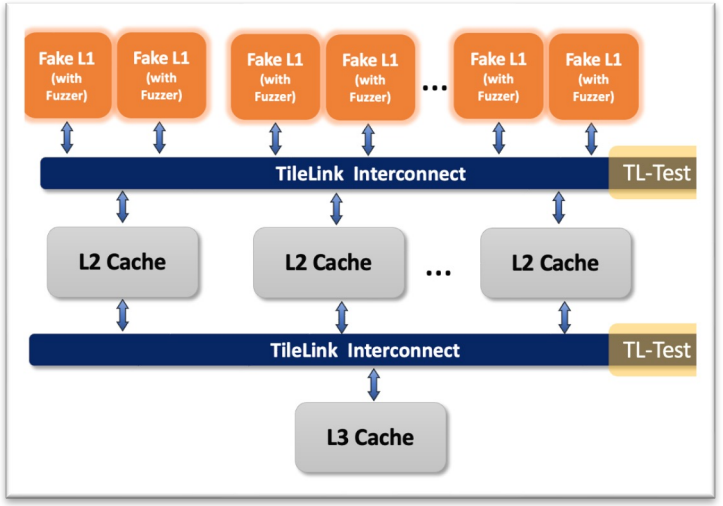
- 处理器缓存验证相关背景
- 已有工作基础
- **TL-Test：基于 TileLink 总线的多场景缓存验证框架**
- TL-Test 在香山处理器中的验证场景
- 未来展望与开放问题

TL-Test：基于 TileLink 总线的多场景缓存验证框架

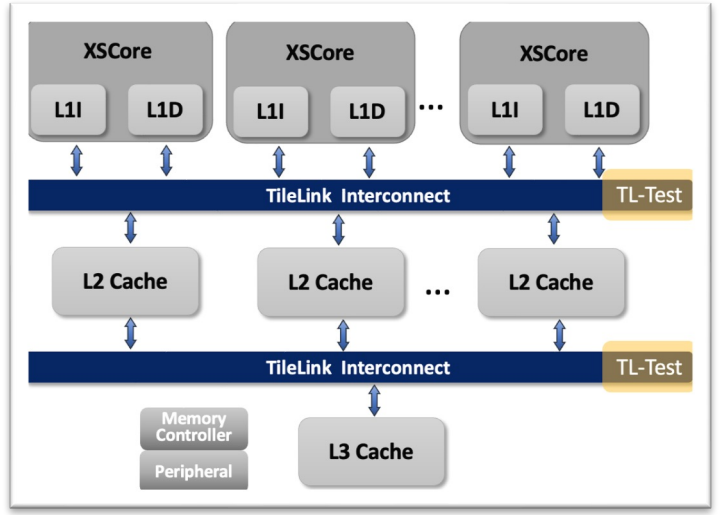
- **核心思想：基于总线抽象出验证平面基础设施**
  - **一套代码**，根据所需验证场景敏捷搭建验证框架



① 单元级验证



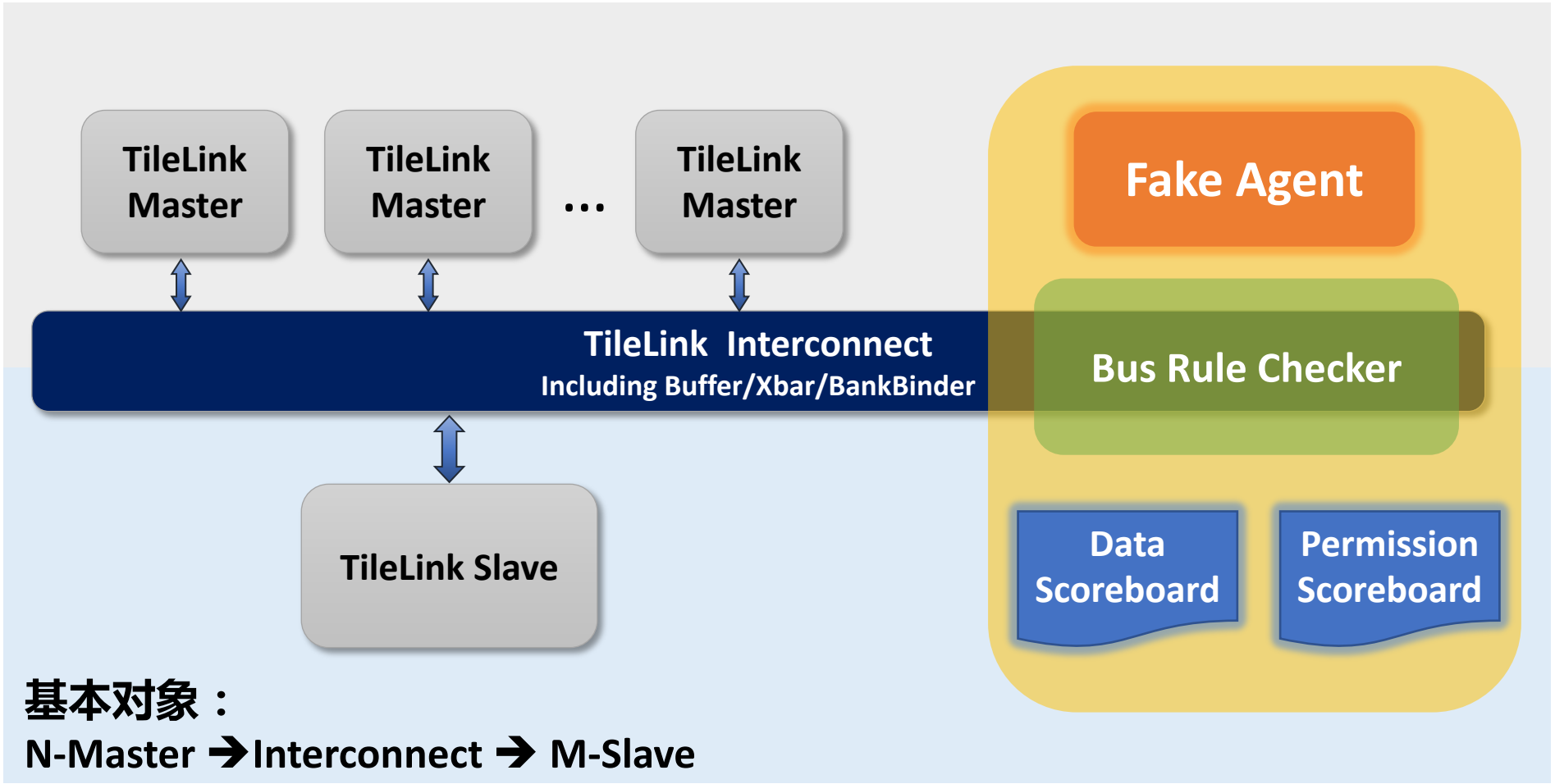
② 子系统级验证



③ SoC 级验证

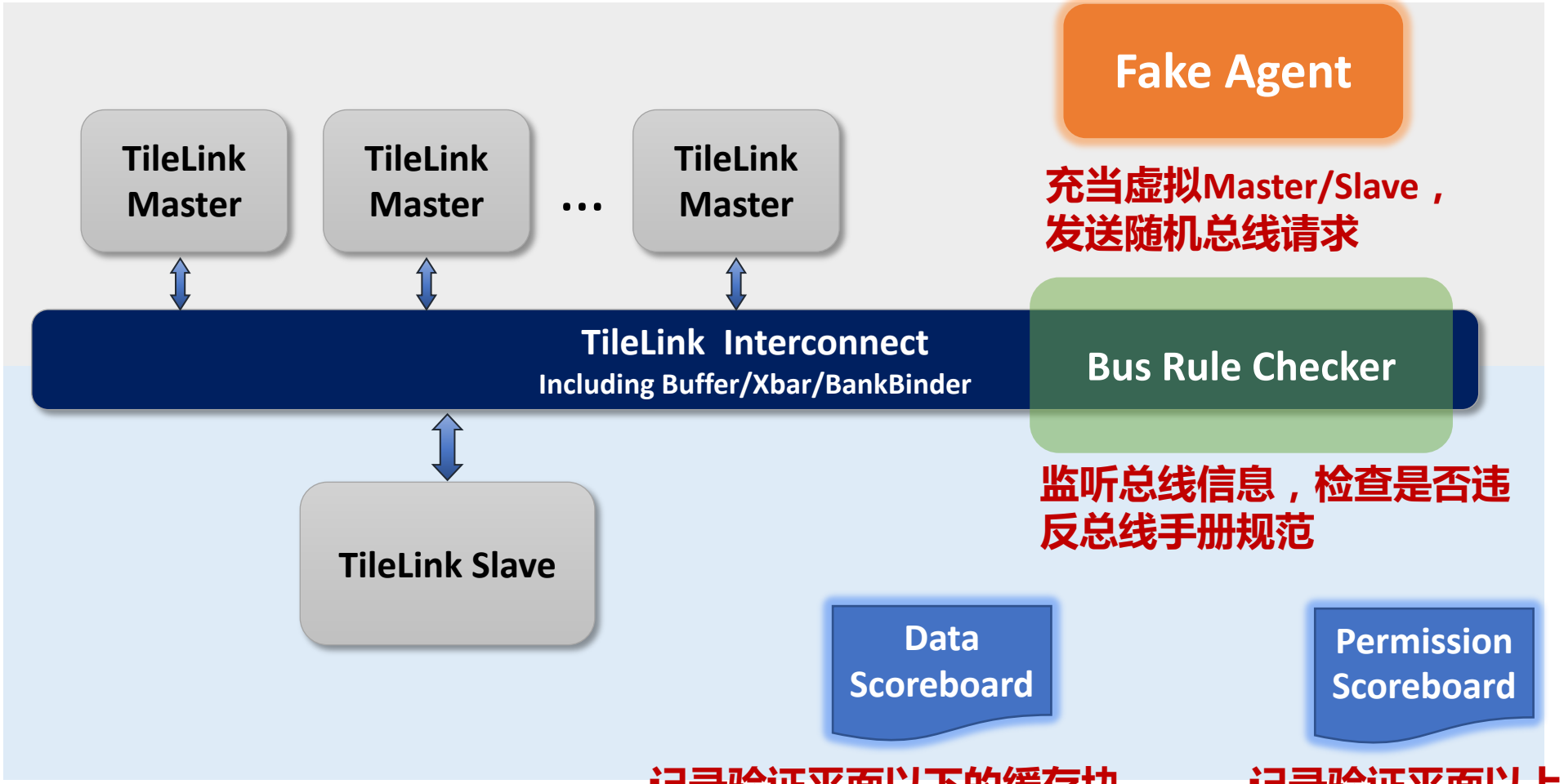
- **使用 C++ 编写，相比 Chisel-Tester2 验证速度提升 16.5X+**

验证平面基础设施



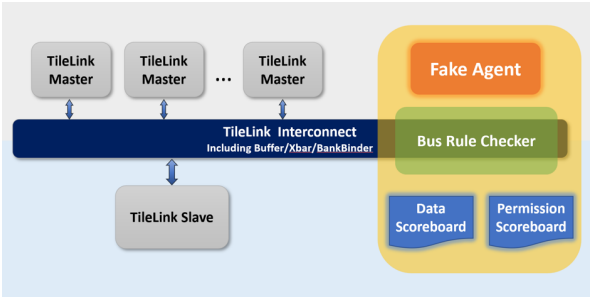
验证平面

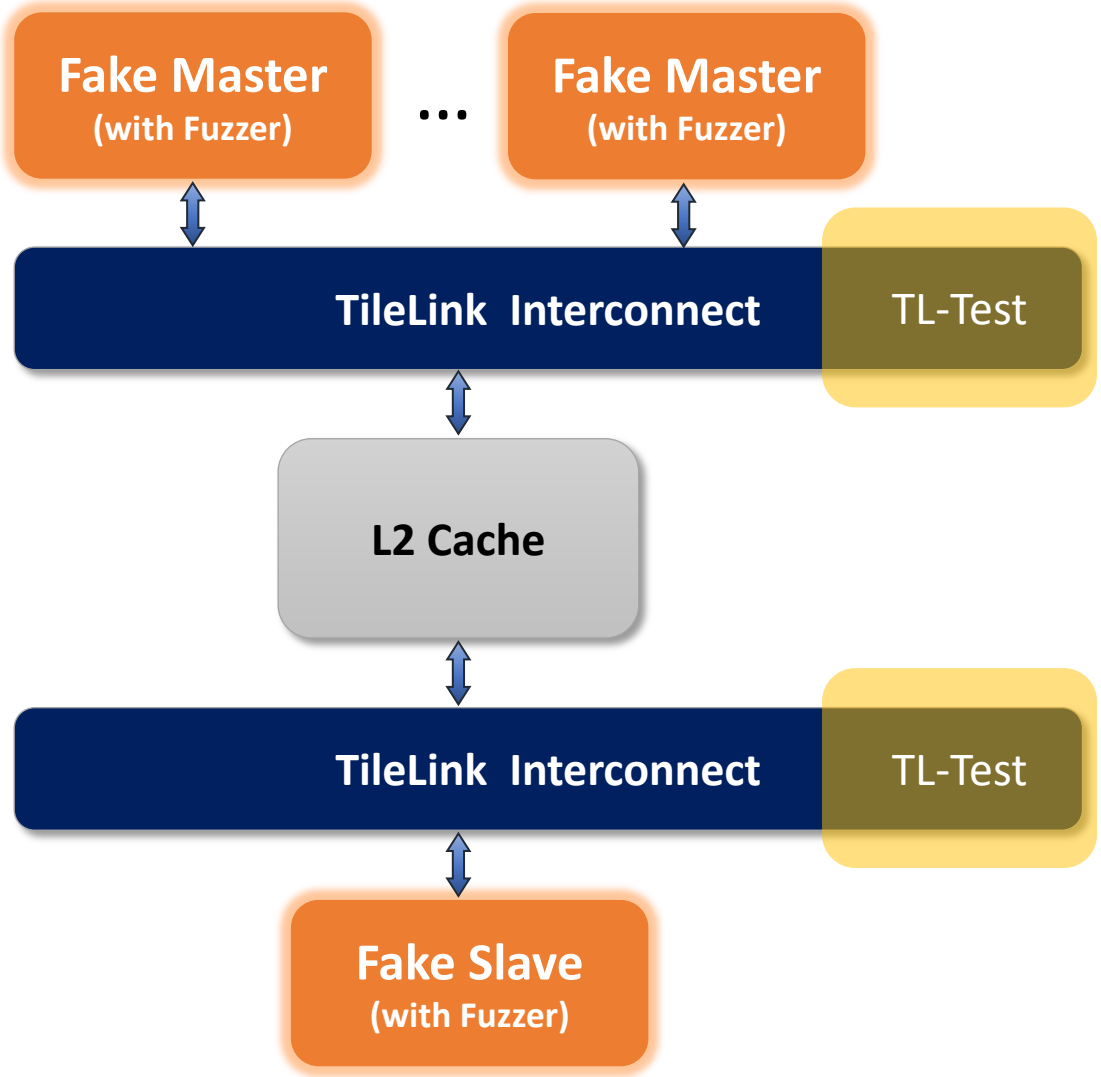
验证平面基础设施



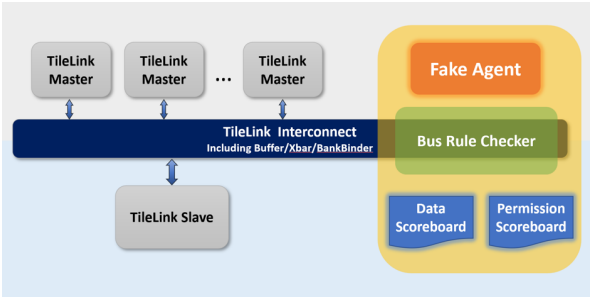
验证平面

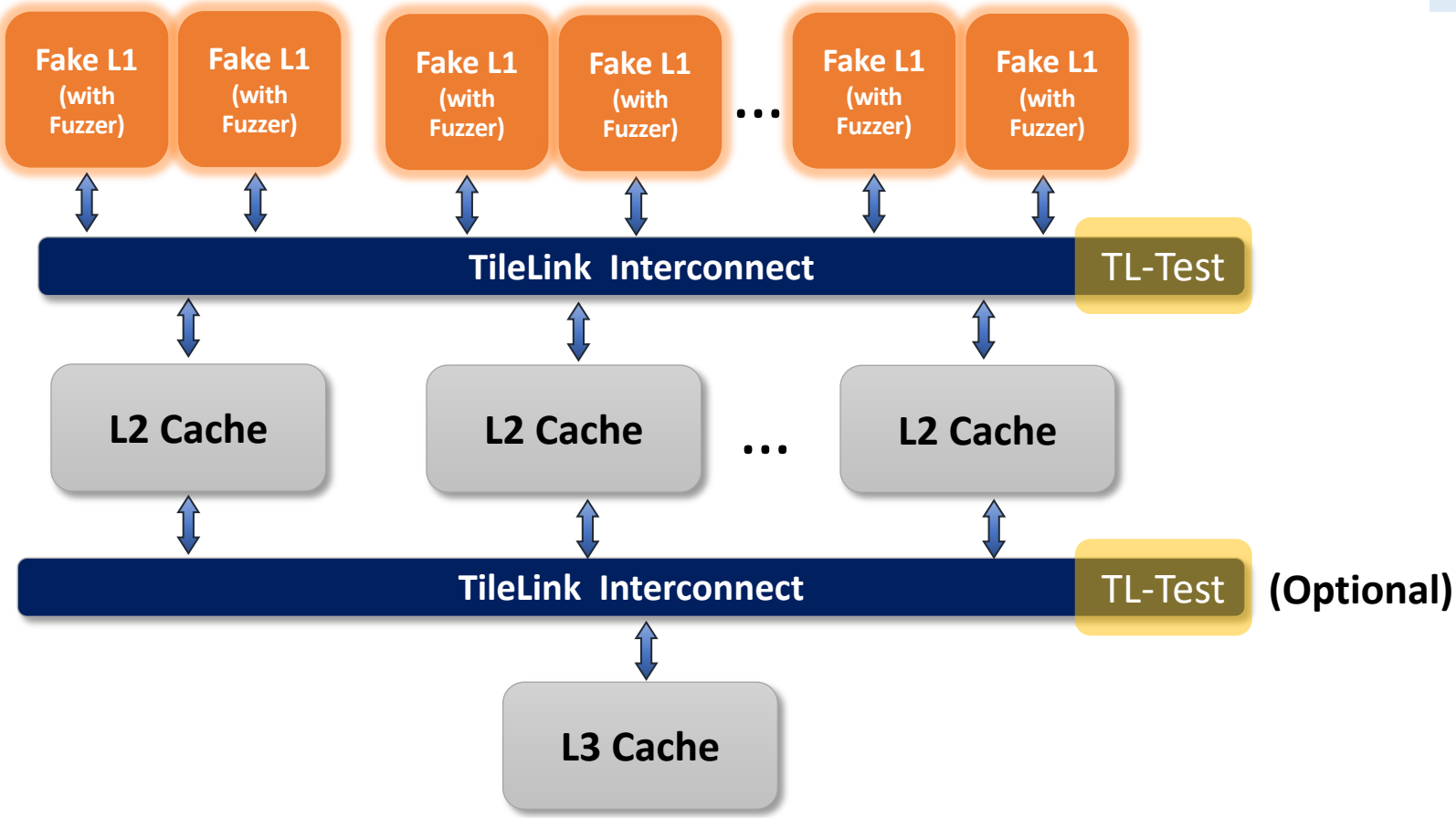
场景①：单元级验证


TileLink Master
TileLink Master
...
TileLink Master
Fake Agent
TileLink Interconnect
Including Buffer/Xbar/BankBinder
Bus Rule Checker
TileLink Slave
Data Scoreboard
Permission Scoreboard

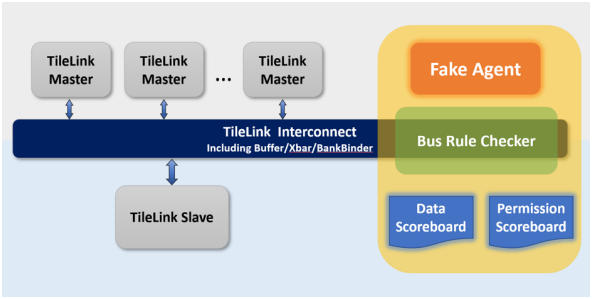

Fake Master
(with Fuzzer)
...
Fake Master
(with Fuzzer)
TileLink Interconnect
TL-Test
L2 Cache
TileLink Interconnect
TL-Test
Fake Slave
(with Fuzzer)

场景②：子系统级验证


TileLink Master
TileLink Master
TileLink Master
Fake Agent
TileLink Interconnect
Including Buffer/Xbar/BankBinder
Bus Rule Checker
TileLink Slave
Data Scoreboard
Permission Scoreboard


Fake L1 (with Fuzzer)
Fake L1 (with Fuzzer)
Fake L1 (with Fuzzer)
Fake L1 (with Fuzzer)
Fake L1 (with Fuzzer)
Fake L1 (with Fuzzer)
TileLink Interconnect
TL-Test
L2 Cache
L2 Cache
L2 Cache
TileLink Interconnect
TL-Test
(Optional)
L3 Cache

场景③：SoC 级验证



编译仿真运行

多线程 Workload

XSCore L1I L1D

XSCore L1I L1D

XSCore L1I L1D

TileLink Interconnect TL-Test

L2 Cache

L2 Cache

L2 Cache

TileLink Interconnect TL-Test (Optional)

Memory Controller

Peripheral

L3 Cache

提升验证压力

- **Fuzzer 部件是提升验证压力的核心**
  - 生成带约束的随机总线请求
  - 合理定义约束来到达更多的 Corner Case

额外的约束定义	目的
➔允许 Opcode 设置为 Get/Put	支持随机发送非一致性 Tilelink 总线请求
➔允许 Opcode 设置为 Prefetch	支持随机发送预取请求
➔限制请求地址空间大小	提升各缓存块的状态转移次数
➔允许发送不同 Color Bits 的请求	验证香山缓存 Cache Anti-Alias 的实现

提升调试效率

- 设置 Monitor 监听验证平面的总线请求
- 汇总成持久化的总线数据库
  - 基于 ChiselDB 技术
  - 将运行过程中的总线事务记录到 Sqlite3 数据库中
  - 出错检索 & 性能分析

蔺嘉炜 @CCC 2022.8.27

时间戳	监控器名称	通道	Opcode	权限升降	状态转换信息	SourceID	SinkID	地址	数据			
116854134	L2_L1I_0	A	AcquireBlock	Grow	NtoB	1	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
116854146	L2_L1I_0	D	GrantData	Cap	toT	1	16	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
116854147	L2_L1I_0	D	GrantData	Cap	toT	1	16	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191840	L2_L1I_0	C	ReleaseData	Shrink	TtoN	0	0	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
123191841	L2_L1I_0	B	Probe	Cap	toN	0	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
123191841	L2_L1I_0	C	ReleaseData	Shrink	TtoN	0	0	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191848	L2_L1I_0	D	ReleaseAck	Cap	toT	0	31	803d3680	0000000020fab05b	0000000020fab45b	0000000020fab85b	0000000020fabc5b
123191851	L3_L2_0	C	ReleaseData	Shrink	TtoN	31	0	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
123191852	L3_L2_0	C	ReleaseData	Shrink	TtoN	31	0	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191862	L3_L2_0	D	ReleaseAck	Cap	toT	31	16	803d3680	86a6f7043583dcd5	f0ef856a865a8762	3583b7654481e17f	865a86a68762f704
123191863	L2_L1I_0	C	ProbeAck	Shrink	TtoN	0	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
123191878	L3_L2_0	C	Release	Report	NtoN	30	0	803d3680	5597bf494981aa09	8522cc2585930000	f84ef15dda0f00ef	00194703b775e84a

示例：数据库中一组违反一致性的请求序列

提升调试效率

- 设置 Monitor 监听验证平面的总线请求
- 汇总成持久化的总线数据库
  - 基于 ChiselDB 技术
  - 将运行过程中的总线事务记录到 Sqlite3 数据库中
  - 出错检索 & 性能分析

蔺嘉炜 @CCC 2022.8.27

时间戳	监控器名称	通道	Opcode	权限升降	状态转换信息	SourceID	SinkID	地址	数据			
116854134	L2_L1I_0	A	AcquireBlock	Grow	NtoB	1	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
116854146	L2_L1I_0	D	GrantData	Cap	toT	1	16	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
116854147	L2_L1I_0	D	GrantData	Cap	toT	1	16	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191840	L2_L1I_0	C	ReleaseData	Shrink	TtoN	0	0	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
123191841	L2_L1I_0	B	Probe	Cap	toN	0	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
123191841	L2_L1I_0	C	ReleaseData	Shrink	TtoN	0	0	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191848	L2_L1I_0	D	ReleaseAck	Cap	toT	0	31	803d3680	0000000020fab05b	0000000020fab45b	0000000020fab85b	0000000020fabc5b
123191851	L3_L2_0	C	ReleaseData	Shrink	TtoN	31	0	803d3680	fa843783f9043903	06090063f8f43c23	f8f4382300093783	378300093023c3bd
123191852	L3_L2_0	C	ReleaseData	Shrink	TtoN	31	0	803d3680	03e32e07bb030089	f0ef8526c881f49b	bb0300893783bbdf	855a010925832e07
123191862	L3_L2_0	D	ReleaseAck	Cap	toT	31	16	803d3680	86a6f7043583dcd5	f0ef856a865a8762	3583b7654481e17f	865a86a68762f704
123191863	L2_L1I_0	C	ProbeAck	Shrink	TtoN	0	0	803d3680	0000000000000000	0000000000000000	0000000000000000	0000000000000000
123191878	L3_L2_0	C	Release	Report	NtoN	30	0	803d3680	5597bf494981aa09	8522cc2585930000	f84ef15dda0f00ef	00194703b775e84a

示例：数据库中一组违反一致性的请求序列

小结：香山处理器缓存模块验证效果

- 集成系统测试跑通所有 SPEC CPU 程序片段后，利用 TL-Test 验证框架额外发现的 Bug 总结

Bug 类型描述	Bug 数量
复杂情况下的 Dir 状态维护错误	8
Through机制与嵌套机制相关	7
一致性策略错误	5
复杂情况下的 Refill Buffer 错误	4
Anti-alias 机制与 Through 机制干扰	2
Probe Helper 机制与 Through 机制干扰	1
时序修改引发的错拍功能 Bug	4

合计：31

- **一个月内修复，成功在 FPGA 上跑通 SPEC CPU 2006 所有程序点以及多线程并发程序**

大纲

- 处理器缓存验证相关背景
- 已有工作基础
- TL-Test：基于 TileLink 总线的多场景缓存验证框架
- TL-Test 在香山处理器中的验证场景
- **未来展望与开放问题**

未来展望

- **更好地描述验证对象**
  - 自动生成 DUT 与验证框架的连接逻辑
  - 敏捷验证：让验证框架自适应匹配不断变动的 DUT 的结构
- **更好地描述验证规则**
  - 形式化描述一致性要求和互联总线规则的检查
  - 解放人工，支持更多的互联总线（ ACE, CHI, etc. ）
- **更好地描述激励**
  - 用形式化的方法刻画激励
  - 持久化激励，累积激励向量用于回归验证

开放问题

➢ “TL-Test As A Platform”

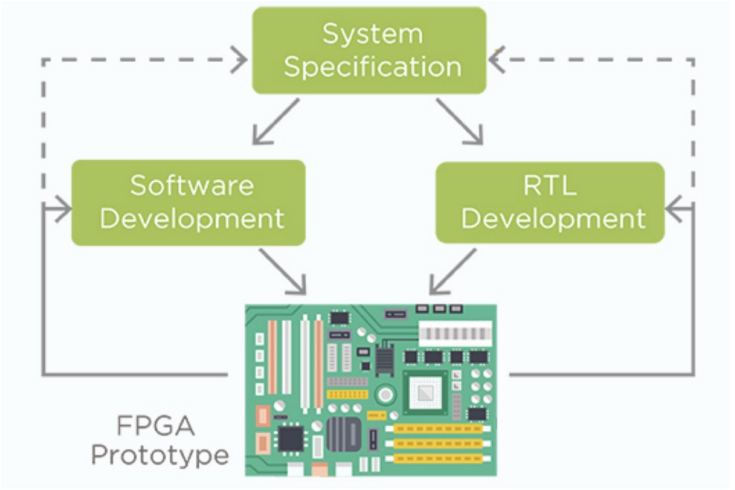
- TL-Test 仅是一个多场景缓存验证框架/平台

解决的问题
➔ 快速搭建缓存验证流程
➔ 缓存数据一致性的检查
➔ 基于 TileLink 总线的权限转移正确性检查
➔ 单元级、子系统级、SoC级多场景覆盖

- 验证领域永恒的话题：**激励 & 覆盖率**
  - 高级的激励生成技术——Coverage Guided Fuzzing，Portable Stimulus Standard，etc？
  - 断言覆盖点、功能覆盖点的自动插入？

开放问题

➢ TL-Test 验证场景的扩充：FPGA 原型验证



- FPGA 原型验证的调试是很大的挑战
  - **Easy** to know：对 or 错
  - **Hard** to know：错在哪里
- 近年来，高级 ILA、状态倾卸重载等 FPGA 信息交互方式不断发展
  - 为原型验证的调试带来了可能
- **TL-Test 可以作为一种总线请求重放器**
  - 复现原型验证场景下的总线事务
  - 借助现有基础在仿真条件下查错纠错

谢谢！
请各位批评指教！