An Overview of Instruction-Level Abstraction for OpenPiton L1.5 Cache

Hongce Zhang

January 2020

1 Design under Verification

OpenPiton is a tiled many-core architecture. It maintains global cache coherence among all processor cores. The cache hierarchy is comprised of a private L1 and L1.5 cache per core and a distributed, shared L2 cache. The OpenPiton many-core architecture implements directory based MESI coherence protocol. Point-to-point coherence messages are transmitted through the three physical channels of the network-on-chip (NoC1 to NoC3). The L1.5 cache serves as both the glue logic, transducing the processor's crossbar protocol to OpenPiton's NoC coherence packet formats, and a write-back layer, caching loads and stores from the write-through L1 cache.

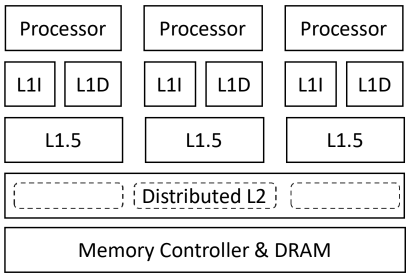


Figure 1: The Memory Hierarchy of OpenPiton

2 The ILA Model

The L1.5 cache receives commands from two interfaces: memory requests from the processor-side on the PCX interface (processor-cache crossbar), NoC packets from L2 cache on NoC2. It initiates NoC packet transmission on NoC1 and NOC3 and responds to the processor on the CPX (cache-processor crossbar) interface. Because L1.5 has two input command interfaces, we model it with two Instruction-Level Abstraction (ILA) for PCX and NoC2 respectively. We omit the BIST scan chain interface in modeling and verification. The interfaces of L1.5 are shown in Fig. 2.

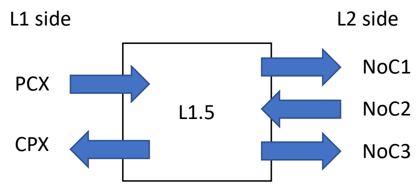
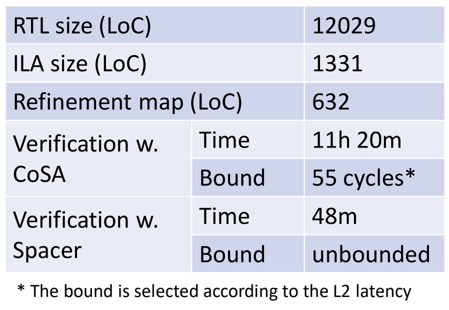


Figure 2: L1.5 I/O Interface

For the PCX interface, we model it with 10 instructions with 3 child-instructions. Child-instructions are for some commands (e.g., `LOAD_NC`) that could sequentially send multiple response packets on the same output interface. For the NOC2 interface, we model it with 6 instructions and 2 child-instructions. As there is no existing formal specification for L1.5, in the process of building the formal model, we decide that some design parameters (e.g., the associativity of L1.5) and features (e.g., the tracking algorithm of the least recently used cache line), are implementation details, as we would like to construct a more general ILA specification that can be used to verify L1.5 caches with various configurations. Therefore, in ILA we omit the specifications of such parameters and features but focus on the I/O packets. A summary of the ILA model is listed in Fig. 4 and Fig. 5.

3 ILA-based Verification

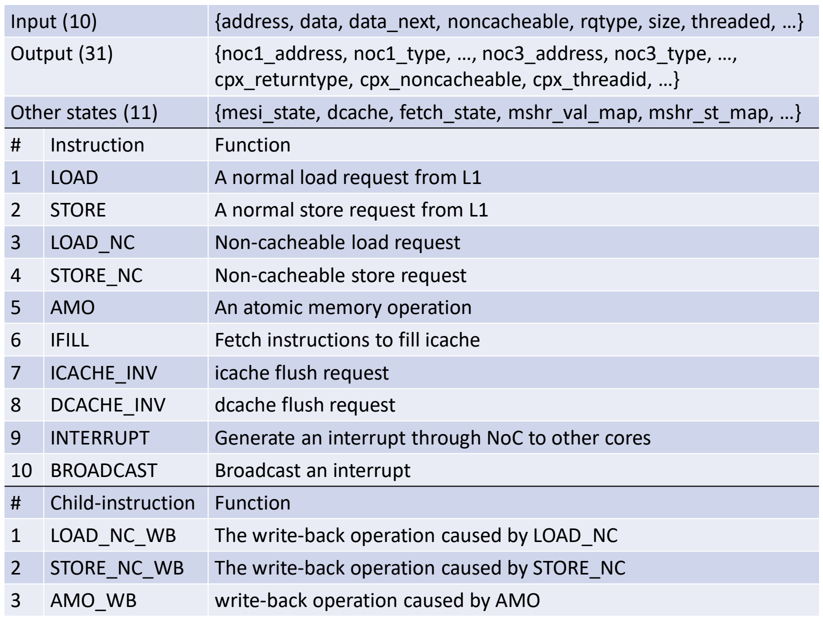


RTL size (LoC)		12029
ILA size (LoC)		1331
Refinement map (LoC)		632
Verification w. CoSA	Time	11h 20m
	Bound	55 cycles*
Verification w. Spacer	Time	48m
	Bound	unbounded

* The bound is selected according to the L2 latency

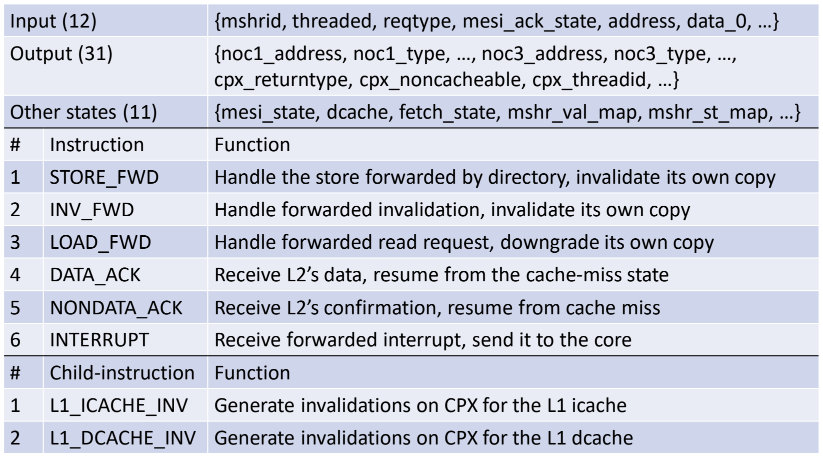
Figure 3: Statistics of ILA and the verification

As the ILA model is built according to the RTL description, the verification of ILA vs. RTL is in fact a check of whether the human-written ILA correctly abstracts the given RTL description. In the verification we exercised our newly-built interface to Spacer model checker (open-source, MIT license), whose model checking technique provides unbounded proof of correctness and the results showed a significant speed-up on this verification task compared to our previous model checking backend.



Input (10)		{address, data, data_next, noncacheable, rqtype, size, threaded, …}
Output (31)		{noc1_address, noc1_type, …, noc3_address, noc3_type, …, cpx_returntype, cpx_noncacheable, cpx_threadid, …}
Other states (11)		{mesi_state, dcache, fetch_state, mshr_val_map, mshr_st_map, …}
#	Instruction	Function
1	LOAD	A normal load request from L1
2	STORE	A normal store request from L1
3	LOAD_NC	Non-cacheable load request
4	STORE_NC	Non-cacheable store request
5	AMO	An atomic memory operation
6	IFILL	Fetch instructions to fill icache
7	ICACHE_INV	icache flush request
8	DCACHE_INV	dcache flush request
9	INTERRUPT	Generate an interrupt through NoC to other cores
10	BROADCAST	Broadcast an interrupt
#	Child-instruction	Function
1	LOAD_NC_WB	The write-back operation caused by LOAD_NC
2	STORE_NC_WB	The write-back operation caused by STORE_NC
3	AMO_WB	write-back operation caused by AMO

Figure 4: ILA Model for L1.5 PCX Interface



Input (12)		{mshrid, threaded, reqtype, mesi_ack_state, address, data_0, …}
Output (31)		{noc1_address, noc1_type, …, noc3_address, noc3_type, …, cpx_returntype, cpx_noncacheable, cpx_threadid, …}
Other states (11)		{mesi_state, dcache, fetch_state, mshr_val_map, mshr_st_map, …}
#	Instruction	Function
1	STORE_FWD	Handle the store forwarded by directory, invalidate its own copy
2	INV_FWD	Handle forwarded invalidation, invalidate its own copy
3	LOAD_FWD	Handle forwarded read request, downgrade its own copy
4	DATA_ACK	Receive L2's data, resume from the cache-miss state
5	NONDATA_ACK	Receive L2's confirmation, resume from cache miss
6	INTERRUPT	Receive forwarded interrupt, send it to the core
#	Child-instruction	Function
1	L1_ICACHE_INV	Generate invalidations on CPX for the L1 icache
2	L1_DCACHE_INV	Generate invalidations on CPX for the L1 dcache

Figure 5: ILA Model for L1.5 NOC2 Interface