Performance modeling and timing verification for DRAM memory subsystems

Thomas Schuster, Peter Prüller, Christian Sauer

cadence

accellera SYSTEMS INITIATIVE

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Motivation

- DRAM is part of almost every SoC package
- Shared amongst multiple processing elements through complex interconnect
- Very often performance bottleneck

- **Need exploration at System Level using fast simulation models to ensure latency and throughput requirements can be met**
- **Need appropriate Simulation IP / especially flexible reconfigurable DRAM controller TLM**

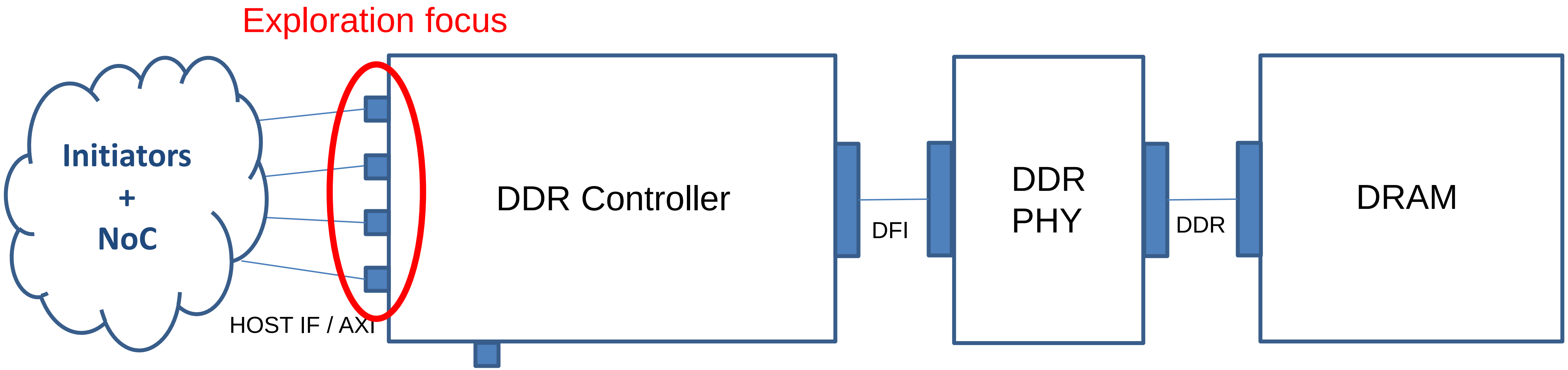
Outline

- Performance modeling approach
  - DRAM memory subsystem
  - TLM interfaces
  - Dataflow modeling
- Verification setups / simulation results
  - Simulation accuracy RTL vs. TLM
  - Latency analysis
  - Data path and fifo monitoring

DDR CONTROLLER PERFORMANCE MODELING

Modeling goals

DRAM memory subsystem

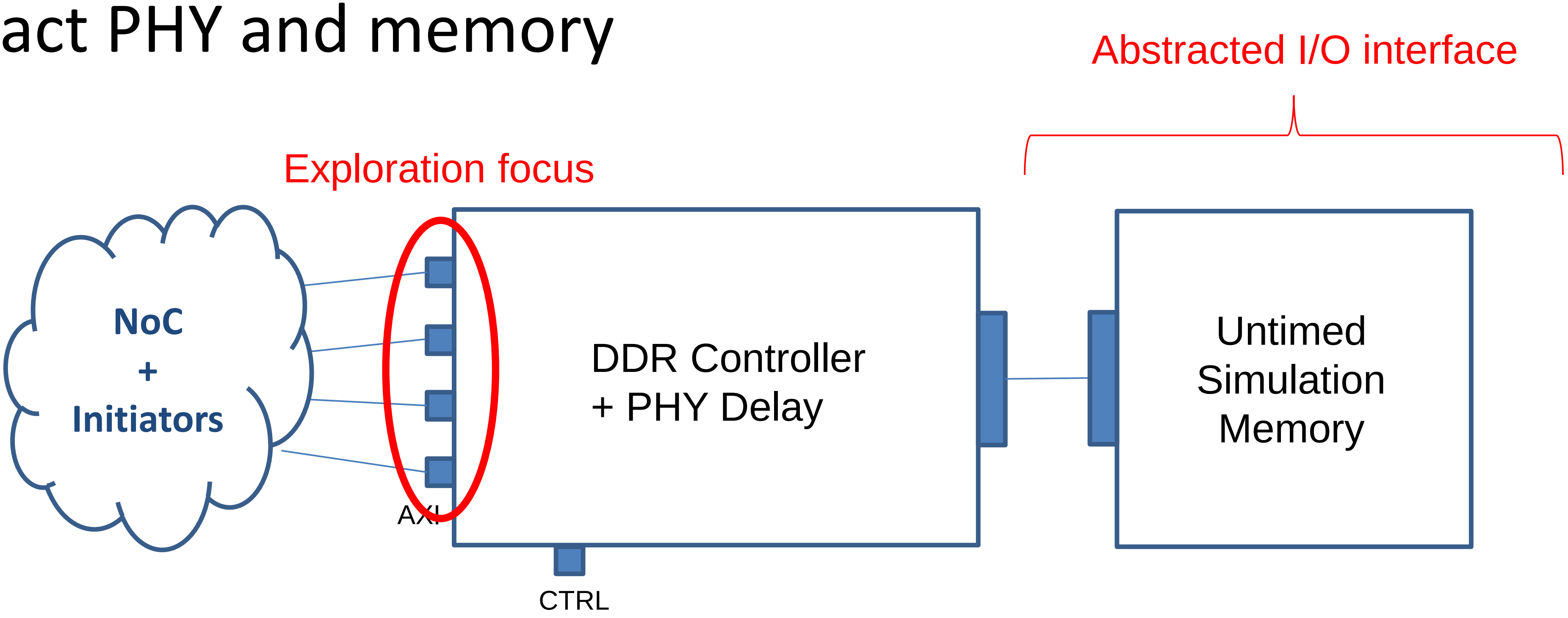


- Measurement of throughput and latency at the host interface in steady-state with max error ~15%

- Indicators for performance bottlenecks
- Significant speedup over RTL simulation
- Configurability / Reuse for different system configurations, memory types, etc.

Modeling approach

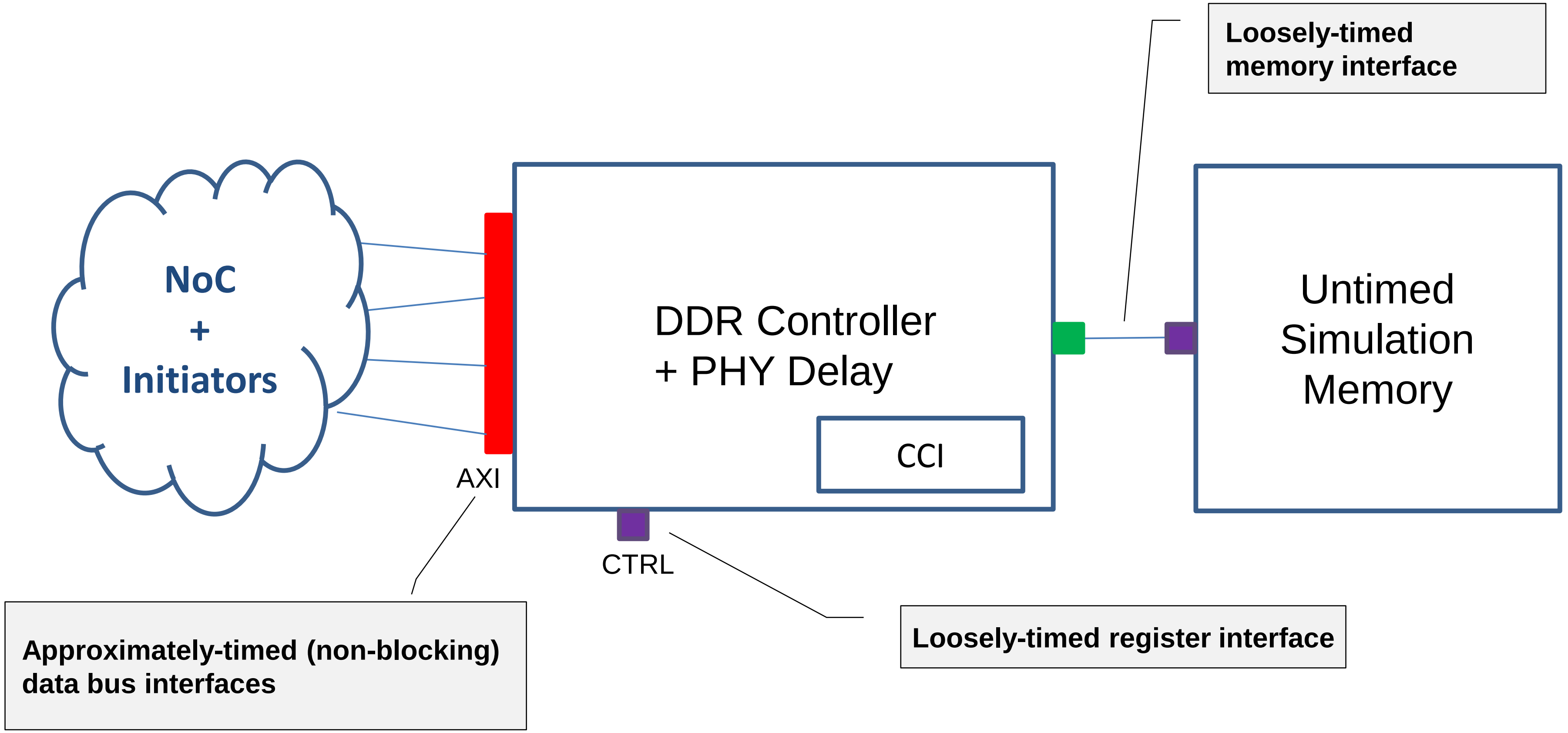
Abstract PHY and memory



- PHY Delays can be added by the controller model
  -> PHY model not needed in steady state
- DDR Controller is aware of DRAM type and layout
  -> data can be held in untimed simulation memory

Modeling approach

Interfaces as TLM2.0 sockets / CCI configuration



Loosely-timed memory interface

Approximately-timed (non-blocking) data bus interfaces

Loosely-timed register interface

- Ignorable payload extensions for side-band signals, e.g. Quality of Service
- TLM standard protocol + allowing reads and writes to be accepted in parallel (AXI)

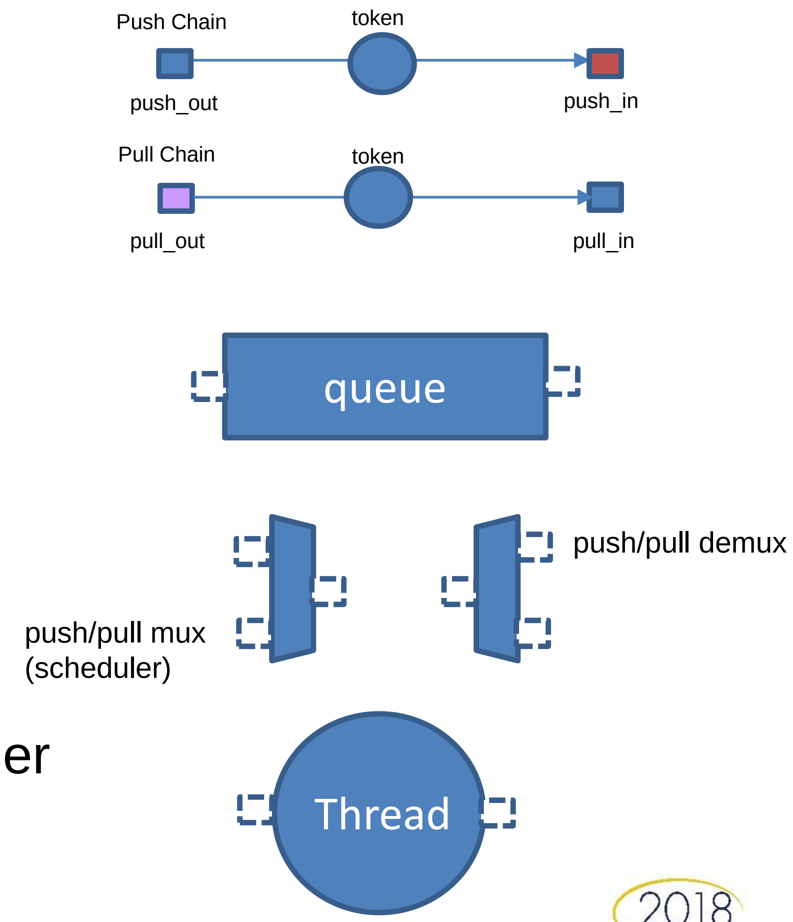
Modeling approach

Behaviour as SystemC dataflow

Library of essential components for modeling the controller:

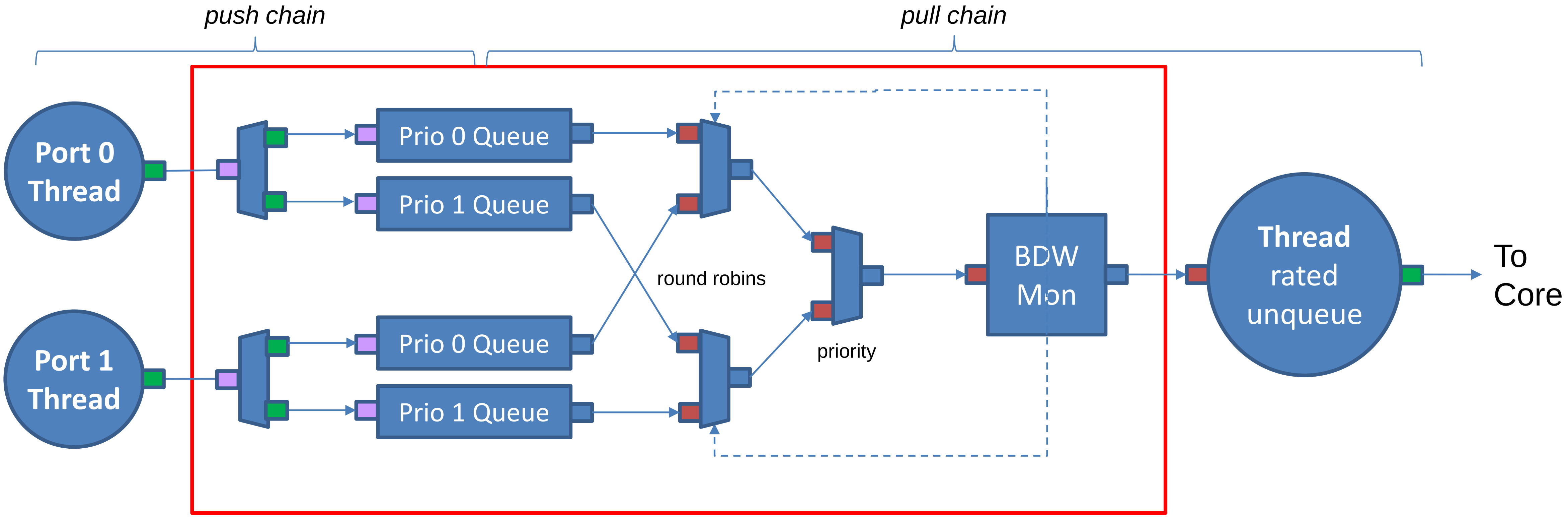
1. Token
   - Basic dataflow objects / ddr command
   - Contains reference to TLM2 generic payload
2. Push/pull ports
   - Interfaces to dataflow components
   - Pushing tokens downstream: out_push, in_push
   - Pull upstream tokens: in_pull, out_pull
3. Queues
4. Mux / Demux
5. Threads
   - Inserting or removing tokens to queues at a rate or on a trigger

Simplifies reuse and maintainance of models !



Dataflow library

Example multi-port arbiter / BW-aware priority round-robin



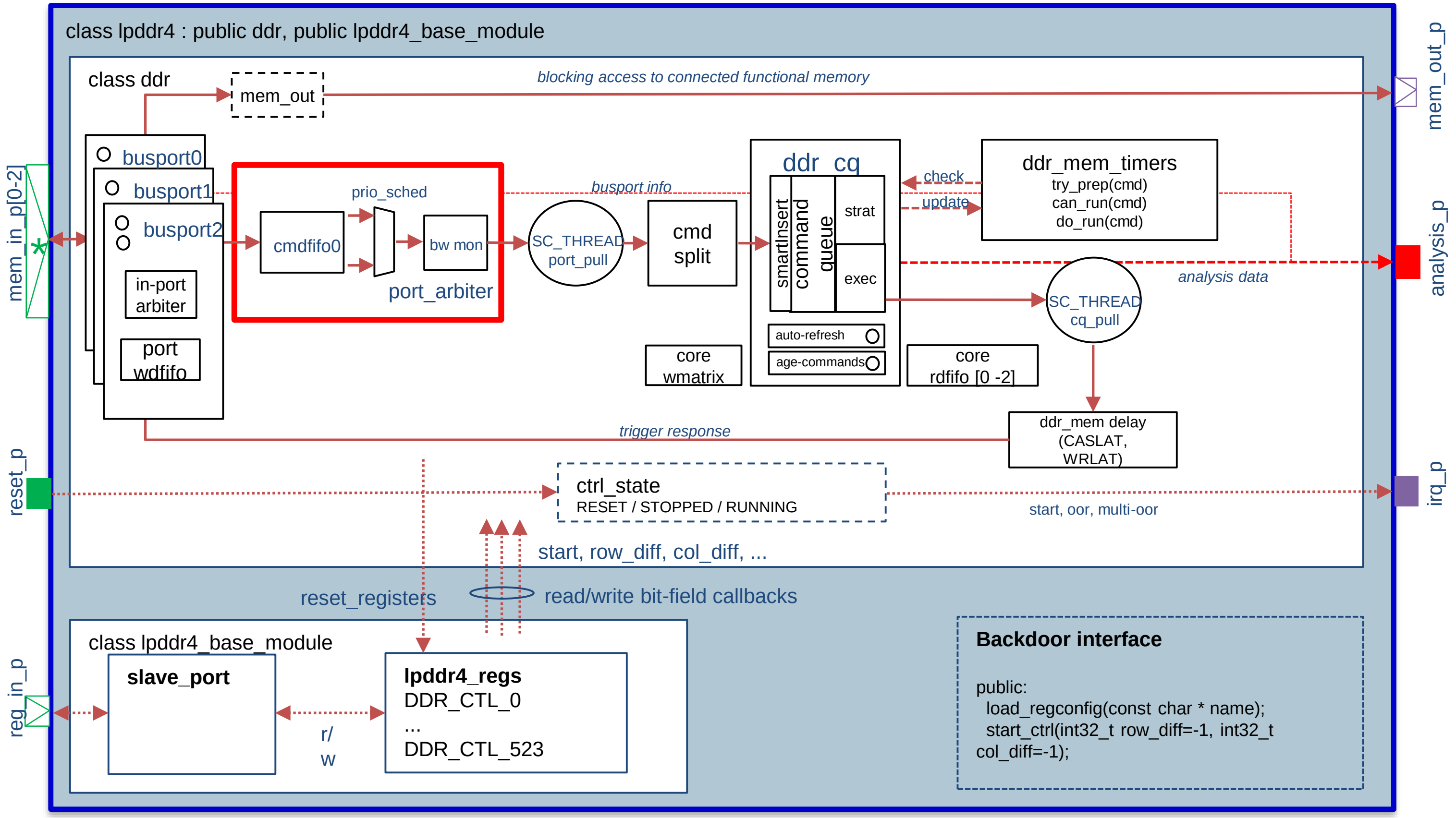
🟦 out_pull 🟩 out_push

🟥 in_pull 🟪 in_push

Tokens are pulled through arbitration by single unqueue thread!

DDR Controller dataflow model

Build on dataflow library



Streaming width of data sockets does not reflect RTL configuration (128bit/64bit/64bit). Set to 32bit for all interfaces on customer request.

☒ tlm_utils::simple_target_socket<32>

☒ tlm_utils::simple_initiator_socket<32>

* sc_vector<tlm_utils::simple_target_socket<32> * >

🟩 sc_in<bool>

🟪 sc_out<bool>

🟥 tlm::tlm_analysis_port<>

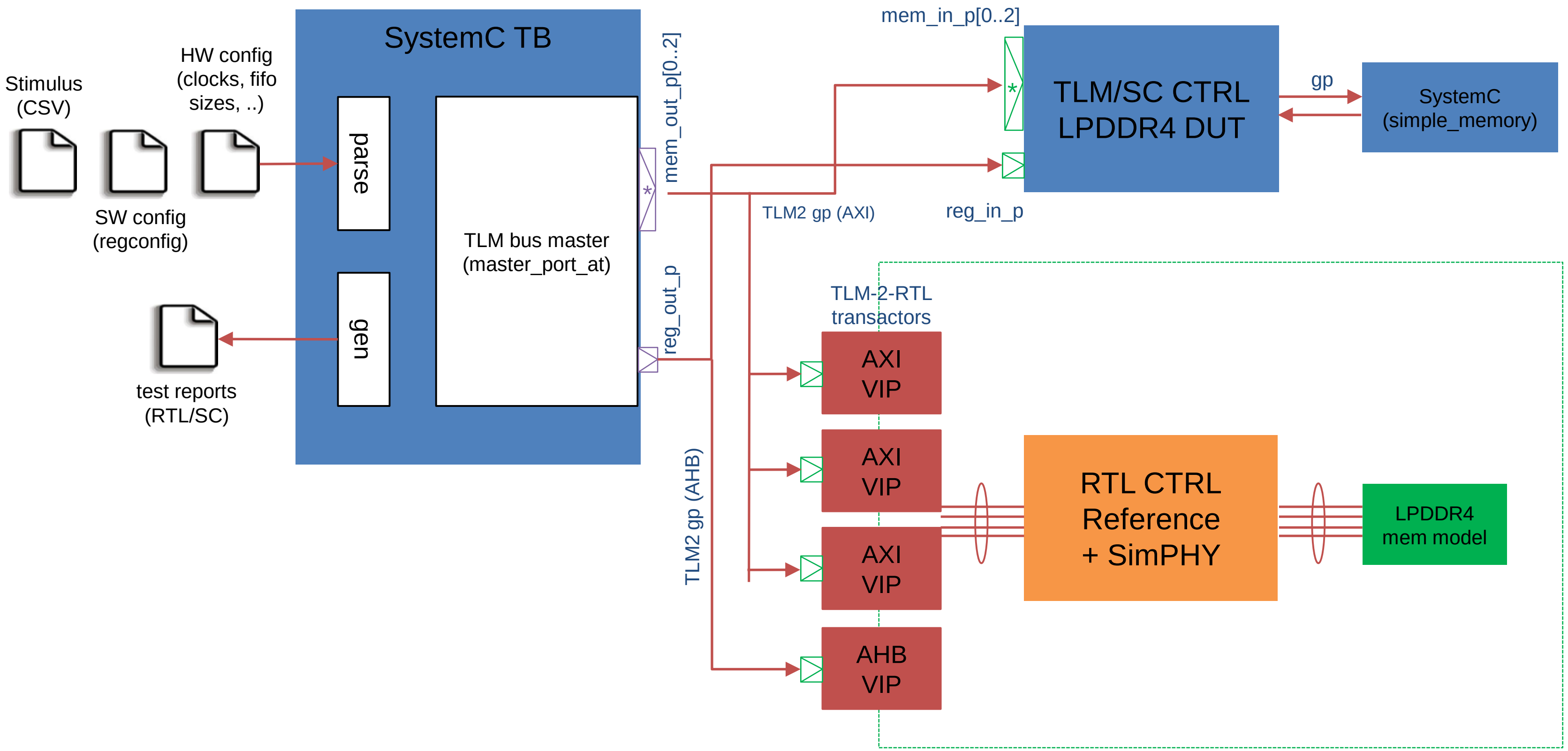
Dataflow:

- Bus transactions translated in one or multiple data flow tokens
- Tokens propagate through system based on backpressure from fifos, queues and memory timers
- Very few threads required *(can be stopped if no data or no space)*

TIMING VERIFICIATION

Timing verification setup

TLM/RTL co-simulation with commercial verification IP



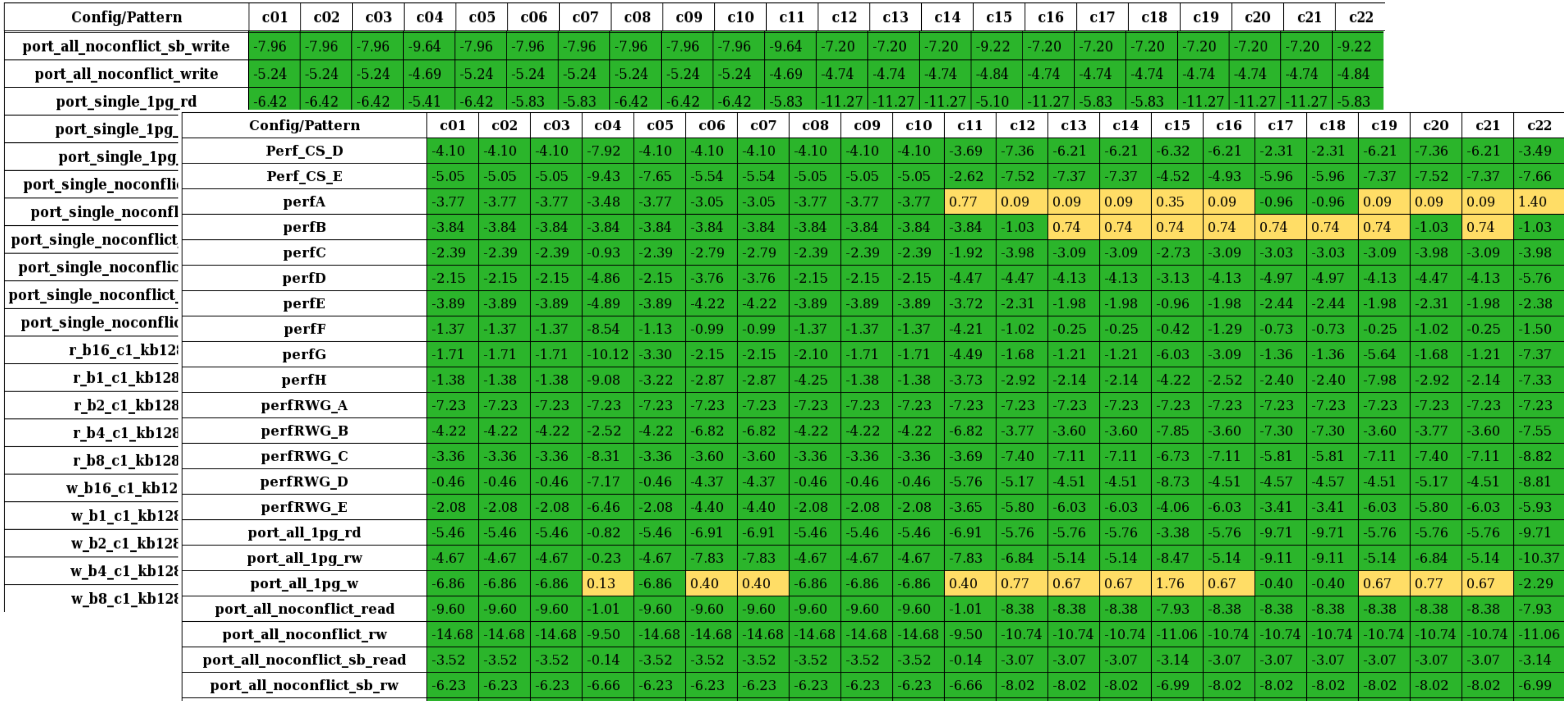
- Common test bench (SystemC) for TLM and RTL designs
- TLM/RTL co-simulation AMBA VIP

Simulation results

Throughput TLM vs. RTL in %

design configurations

test pattern



Config/Pattern	c01	c02	c03	c04	c05	c06	c07	c08	c09	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22
port_all_noconflict_sb_write	-7.96	-7.96	-7.96	-9.64	-7.96	-7.96	-7.96	-7.96	-7.96	-7.96	-9.64	-7.20	-7.20	-7.20	-9.22	-7.20	-7.20	-7.20	-7.20	-7.20	-7.20	-9.22
port_all_noconflict_write	-5.24	-5.24	-5.24	-4.69	-5.24	-5.24	-5.24	-5.24	-5.24	-5.24	-4.69	-4.74	-4.74	-4.74	-4.84	-4.74	-4.74	-4.74	-4.74	-4.74	-4.74	-4.84
port_single_1pg_rd	-6.42	-6.42	-6.42	-5.41	-6.42	-5.83	-5.83	-6.42	-6.42	-6.42	-5.83	-11.27	-11.27	-11.27	-5.10	-11.27	-5.83	-5.83	-11.27	-11.27	-11.27	-5.83
port_single_1pg_																						
port_single_1pg																						
port_single_noconfli																						
port_single_noconfl																						
port_single_noconflict																						
port_single_noconflic																						
port_single_noconflict_																						
port_single_noconflic																						
r_b16_c1_kb12																						
r_b1_c1_kb128																						
r_b2_c1_kb128																						
r_b4_c1_kb128																						
r_b8_c1_kb128																						
w_b16_c1_kb12																						
w_b1_c1_kb12																						
w_b2_c1_kb12																						
w_b4_c1_kb12																						
w_b8_c1_kb12																						

Config/Pattern	c01	c02	c03	c04	c05	c06	c07	c08	c09	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22
Perf_CS_D	-4.10	-4.10	-4.10	-7.92	-4.10	-4.10	-4.10	-4.10	-4.10	-4.10	-3.69	-7.36	-6.21	-6.21	-6.32	-6.21	-2.31	-2.31	-6.21	-7.36	-6.21	-3.49
Perf_CS_E	-5.05	-5.05	-5.05	-9.43	-7.65	-5.54	-5.54	-5.05	-5.05	-5.05	-2.62	-7.52	-7.37	-7.37	-4.52	-4.93	-5.96	-5.96	-7.37	-7.52	-7.37	-7.66
perfA	-3.77	-3.77	-3.77	-3.48	-3.77	-3.05	-3.05	-3.77	-3.77	-3.77	0.77	0.09	0.09	0.09	0.35	0.09	-0.96	-0.96	0.09	0.09	0.09	1.40
perfB	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-3.84	-1.03	0.74	0.74	0.74	0.74	0.74	0.74	0.74	-1.03	0.74	-1.03
perfC	-2.39	-2.39	-2.39	-0.93	-2.39	-2.79	-2.79	-2.39	-2.39	-2.39	-1.92	-3.98	-3.09	-3.09	-2.73	-3.09	-3.03	-3.03	-3.09	-3.98	-3.09	-3.98
perfD	-2.15	-2.15	-2.15	-4.86	-2.15	-3.76	-3.76	-2.15	-2.15	-2.15	-4.47	-4.47	-4.13	-4.13	-3.13	-4.13	-4.97	-4.97	-4.13	-4.47	-4.13	-5.76
perfE	-3.89	-3.89	-3.89	-4.89	-3.89	-4.22	-4.22	-3.89	-3.89	-3.89	-3.72	-2.31	-1.98	-1.98	-0.96	-1.98	-2.44	-2.44	-1.98	-2.31	-1.98	-2.38
perfF	-1.37	-1.37	-1.37	-8.54	-1.13	-0.99	-0.99	-1.37	-1.37	-1.37	-4.21	-1.02	-0.25	-0.25	-0.42	-1.29	-0.73	-0.73	-0.25	-1.02	-0.25	-1.50
perfG	-1.71	-1.71	-1.71	-10.12	-3.30	-2.15	-2.15	-2.10	-1.71	-1.71	-4.49	-1.68	-1.21	-1.21	-6.03	-3.09	-1.36	-1.36	-5.64	-1.68	-1.21	-7.37
perfH	-1.38	-1.38	-1.38	-9.08	-3.22	-2.87	-2.87	-4.25	-1.38	-1.38	-3.73	-2.92	-2.14	-2.14	-4.22	-2.52	-2.40	-2.40	-7.98	-2.92	-2.14	-7.33
perfRWG_A	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23	-7.23
perfRWG_B	-4.22	-4.22	-4.22	-2.52	-4.22	-6.82	-6.82	-4.22	-4.22	-4.22	-6.82	-3.77	-3.60	-3.60	-7.85	-3.60	-7.30	-7.30	-3.60	-3.77	-3.60	-7.55
perfRWG_C	-3.36	-3.36	-3.36	-8.31	-3.36	-3.60	-3.60	-3.36	-3.36	-3.36	-3.69	-7.40	-7.11	-7.11	-6.73	-7.11	-5.81	-5.81	-7.11	-7.40	-7.11	-8.82
perfRWG_D	-0.46	-0.46	-0.46	-7.17	-0.46	-4.37	-4.37	-0.46	-0.46	-0.46	-5.76	-5.17	-4.51	-4.51	-8.73	-4.51	-4.57	-4.57	-4.51	-5.17	-4.51	-8.81
perfRWG_E	-2.08	-2.08	-2.08	-6.46	-2.08	-4.40	-4.40	-2.08	-2.08	-2.08	-3.65	-5.80	-6.03	-6.03	-4.06	-6.03	-3.41	-3.41	-6.03	-5.80	-6.03	-5.93
port_all_1pg_rd	-5.46	-5.46	-5.46	-0.82	-5.46	-6.91	-6.91	-5.46	-5.46	-5.46	-6.91	-5.76	-5.76	-5.76	-3.38	-5.76	-9.71	-9.71	-5.76	-5.76	-5.76	-9.71
port_all_1pg_rw	-4.67	-4.67	-4.67	-0.23	-4.67	-7.83	-7.83	-4.67	-4.67	-4.67	-7.83	-6.84	-5.14	-5.14	-8.47	-5.14	-9.11	-9.11	-5.14	-6.84	-5.14	-10.37
port_all_1pg_w	-6.86	-6.86	-6.86	0.13	-6.86	0.40	0.40	-6.86	-6.86	-6.86	0.40	0.77	0.67	0.67	1.76	0.67	-0.40	-0.40	0.67	0.77	0.67	-2.29
port_all_noconflict_read	-9.60	-9.60	-9.60	-1.01	-9.60	-9.60	-9.60	-9.60	-9.60	-9.60	-1.01	-8.38	-8.38	-8.38	-7.93	-8.38	-8.38	-8.38	-8.38	-8.38	-8.38	-7.93
port_all_noconflict_rw	-14.68	-14.68	-14.68	-9.50	-14.68	-14.68	-14.68	-14.68	-14.68	-14.68	-9.50	-10.74	-10.74	-10.74	-11.06	-10.74	-10.74	-10.74	-10.74	-10.74	-10.74	-11.06
port_all_noconflict_sb_read	-3.52	-3.52	-3.52	-0.14	-3.52	-3.52	-3.52	-3.52	-3.52	-3.52	-0.14	-3.07	-3.07	-3.07	-3.14	-3.07	-3.07	-3.07	-3.07	-3.07	-3.07	-3.14
port_all_noconflict_sb_rw	-6.23	-6.23	-6.23	-6.66	-6.23	-6.23	-6.23	-6.23	-6.23	-6.23	-6.66	-8.02	-8.02	-8.02	-6.99	-8.02	-8.02	-8.02	-8.02	-8.02	-8.02	-6.99

🟩 SystemC 0-15% slower than RTL (target range)

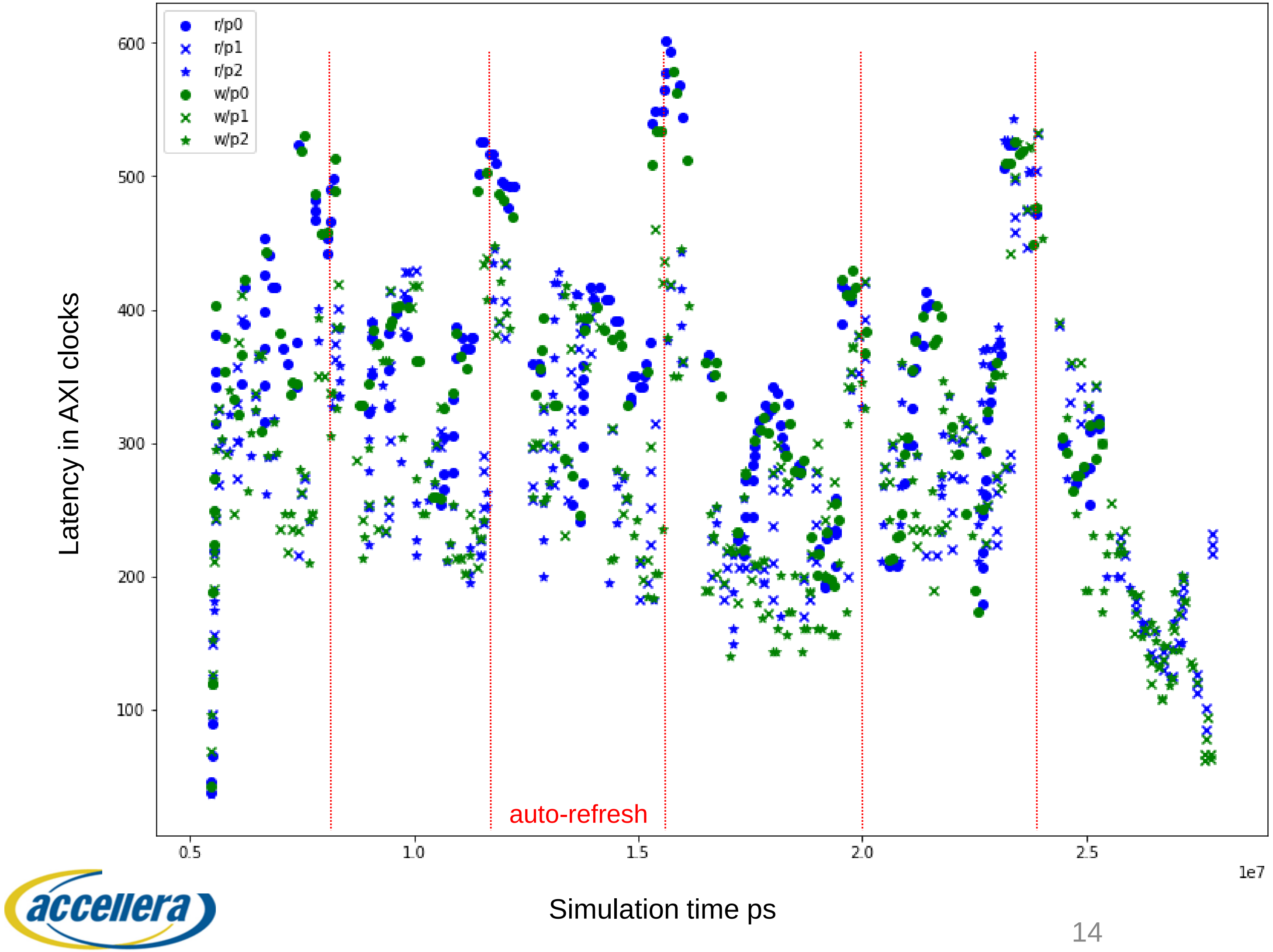
🟨 SystemC 0-15% faster than RTL (needs fixing)

🟥 Error > 15% (needs fixing)

Goal: TLM should always predict pessimistically (0 – 15% slower than RTL)

Simulation results

Latency over time

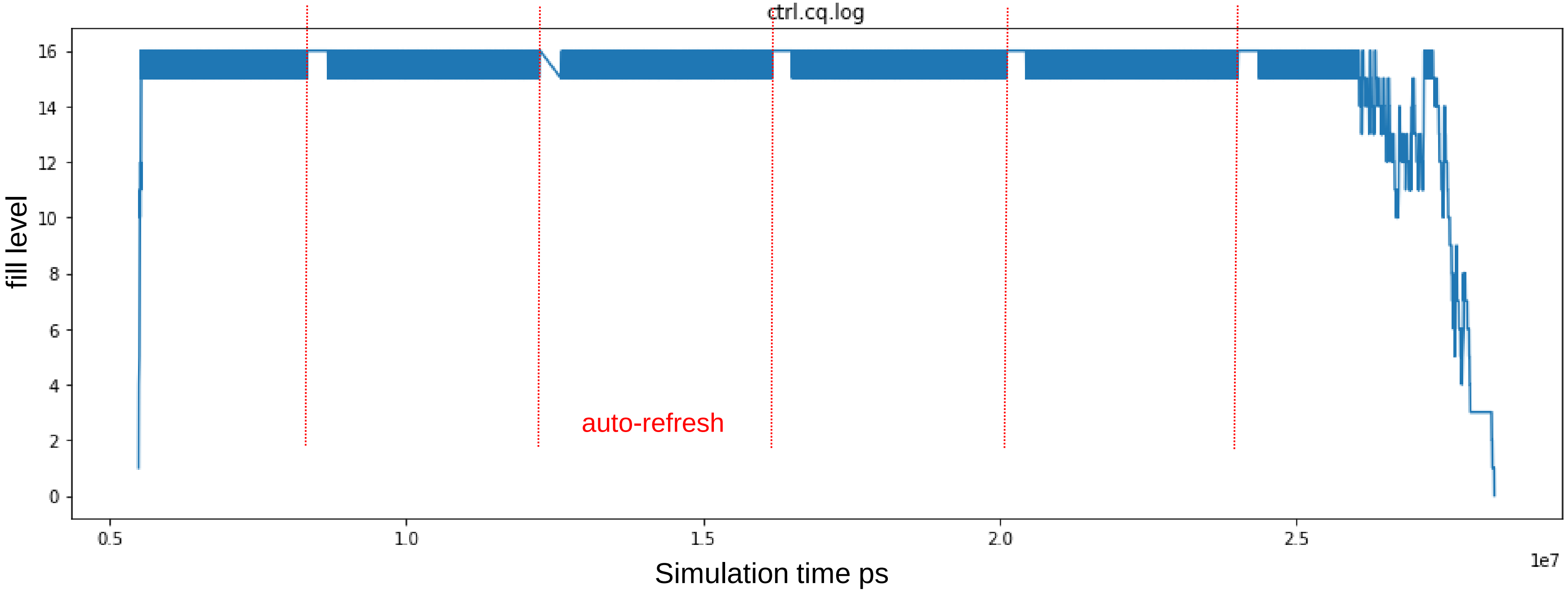


Example: Random r/w traffic on three ports

- Latency of reads and writes from different ports
- Low latency @ simulation begin
- High latency after queue has filled up (random traffic with low locality causes many bank activations)
- Auto-refreshes temporary stop execution (latency peaks)

Simulation results

Controller command queue

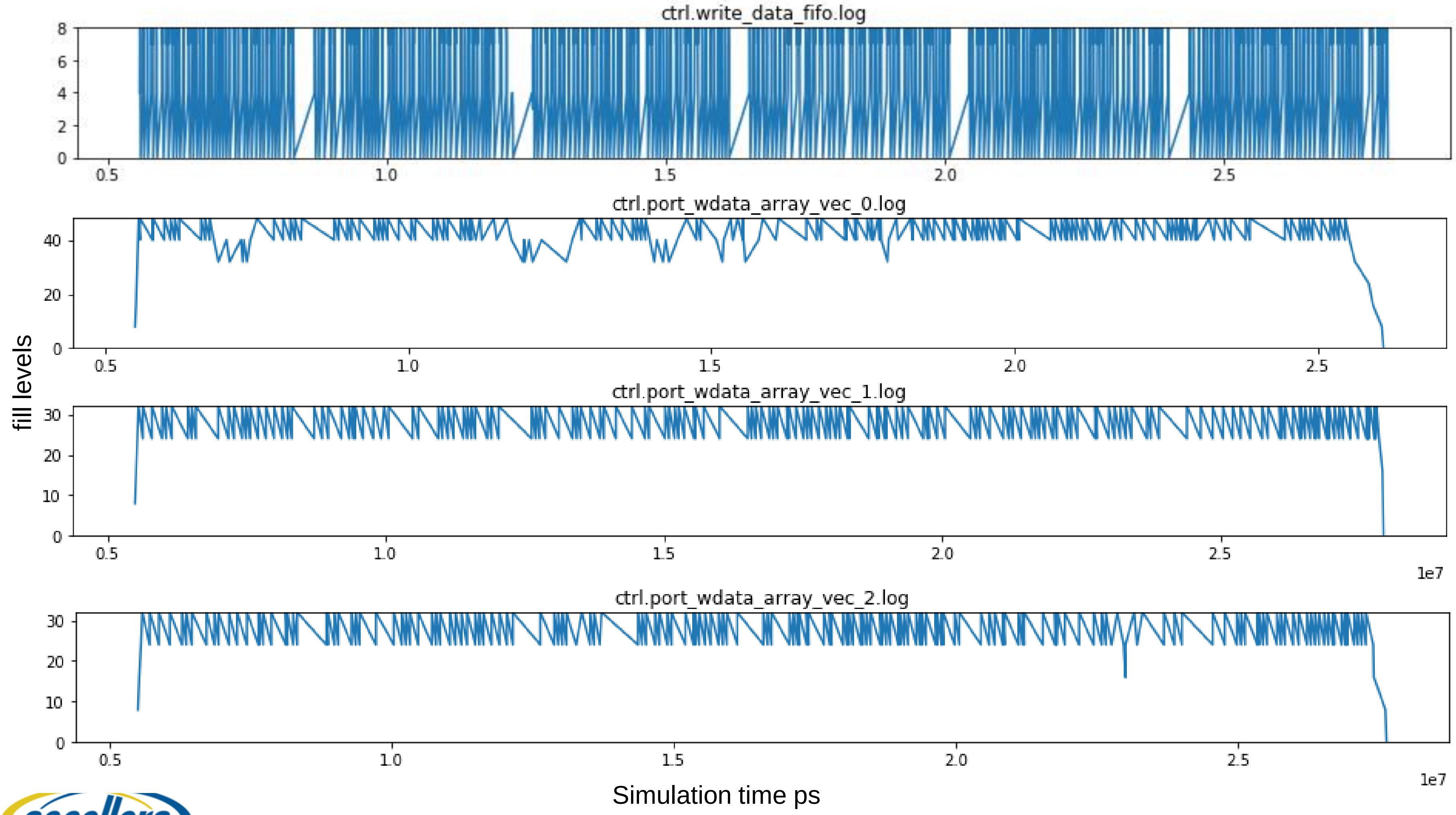


Observations:

- Command queue quickly fills up and remains full
- Back-pressure from core command queue)

Simulation results

Write data path fifos

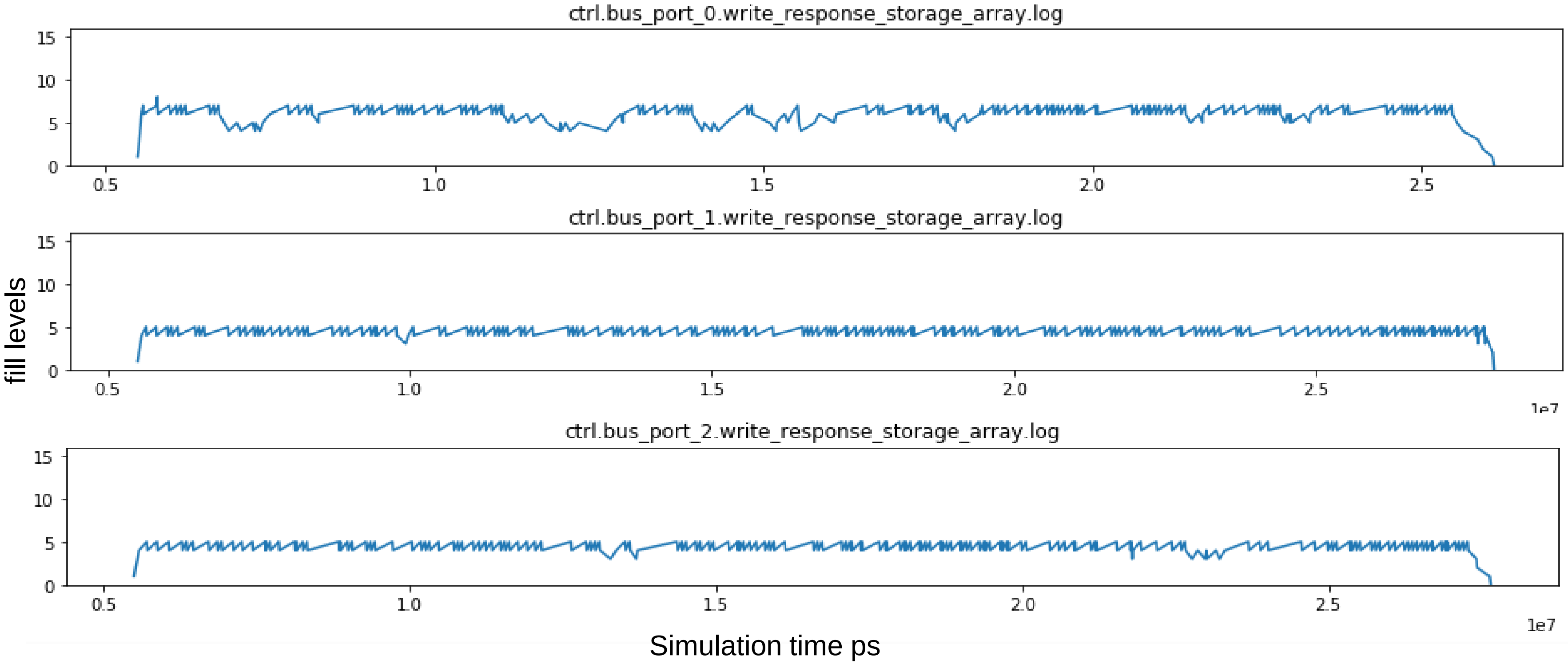


Observation:

- Core write data fifo is to small (only 8 entries)
- Execution of writes slowed down / stalls back into bus port

Simulation results

Write response fifos

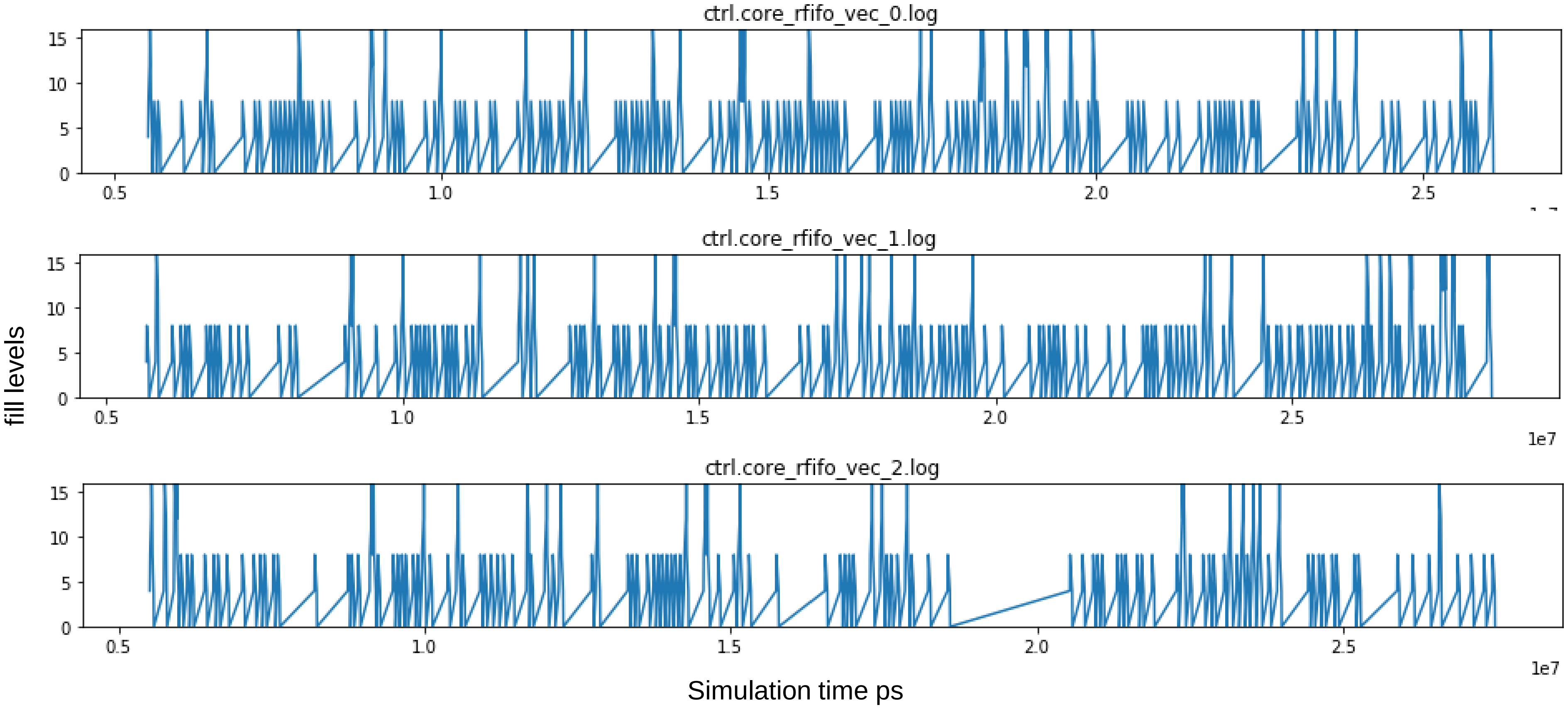


Observation:

- No stalls on write response channel
- Number of writes to be accepted at a time (outstanding write responses) limited by capacity of write data fifos (previous slid)

Simulation results

Read data path fifos



Observation:

- Very view stalls on read data path
- Fifo size appears sufficient
  (for given traffic)

Conclusion

- Performance exploration of SoC can be leverages using an Approximately-timed DDR CTRL TLM
- Early understanding of performance bottle-necks leads to better quality of results
- Accuracy goals throughput / latency within reach

Thanks !

Questions ?