Shining a Light on Full FPGA and ASIC Performance

Adam Sherer, Account Technical Executive, Cadence Design Systems

STAC Fall 2022

cādence®

Agenda

- Performance in low-latency trading systems

- Simulation-based performance analysis

- Using assertions for performance analysis

- System level performance analysis

- Call to action

cādence®

Shining a Light on Full FPGA and ASIC Performance

- FPGA and ASIC HFT accelerators are architected and coded for functionality and performance

- It works!!! … but not fast enough in all conditions

- Performance anomalies often found in lab or live
  - Occur due to core cases and/or system saturation
  - Visibility inside FGPAs is limited for debug
  - For ASIC, analysis must be done pre-silicon

- Verification shines for visibility and analysis
  - Block – order of algorithms and data paths
  - Interfaces – data transport efficiency
  - Full system – memory and on chip bus latency
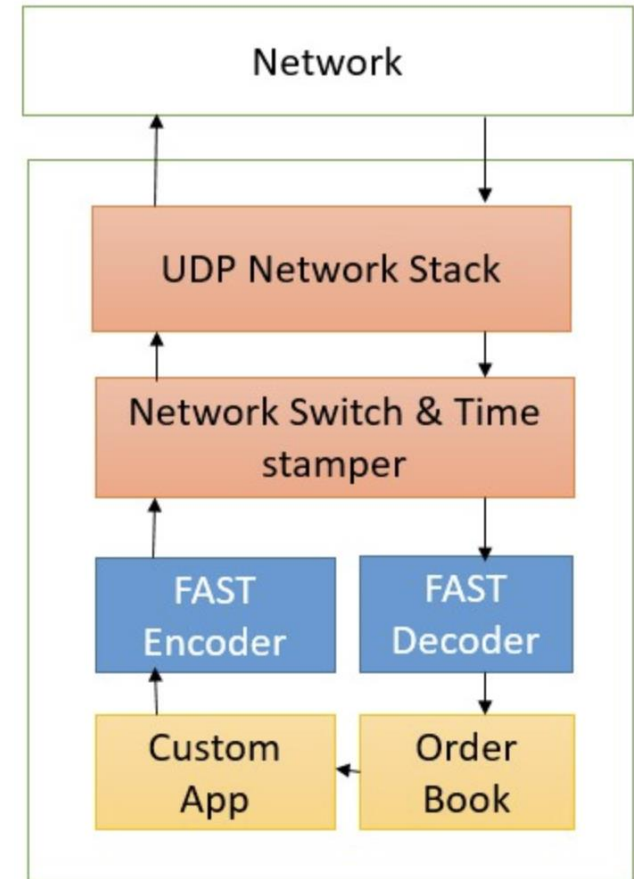
Source: "The Architecture of HFT System", Tejasvi Shiv, June 7, 2021, FPGAs for Stock Market Trading: https://medium.com/fpgas-for-stock-market-trading/the-architecture-of-hft-system-713e64604a61

cādence®

Designing Hardware Blocks for Performance

- Know the order of your algorithms in hardware
    - Individual blocks may run fast enough but become bottlenecks when system scales

- Maximize combinatorial logic
    - Unroll loops to reduce clocks for computation
    - Pre-calculate invariants to reduce computation in loops and/or data path
    - Create testbench monitors to observe combinatorial performance
    - Add clocking domains to reduce clock skew issues

- Perform all price and time processing in fixed point (integer) notation (assumed to be common practice today)
    - Utilize incremental vs full value calculations to reduce computational area

cādence®

Analyze Regression for Performance Issues and Opportunities

Modules (blocks)

```
-----------------------------------------------------------
Most Active Modules (behavioral)
-----------------------------------------------------------
%hits #hits  #inst  name
 6.8 96670 [16049] worklib.dsp_delay:v (file: /tmp/myproject/dsp/rtl/dsp_delay.v line: 29)
 3.9 55715 [  703] worklib.gen_mux_ohot:v (file: /tmp/myproject/gen/rtl/gen_mux_ohot.v line: 30)
 3.4 49232 [ 1163] worklib.inv_tim_adj:sv (file: /tmp/myproject/bench/models/inv_tim_adj.sv line: 5)
 3.3 47636 [16952] worklib.dsp_convert:v (file: /tmp/myproject/dsp/rtl/dsp_convert.v line: 56)
 3.3 47408 [ 3704] worklib.dsp_add:v (file: /tmp/myproject/dsp/rtl/dsp_add.v line: 64)
 1.9 27181 [ 1680] worklib.gen_counter_ripple_v2:v (file: /tmp/myproject/gen/rtl/gen_counter_ripple_v2
```

Total time per module at line in file

Streams (ex. `always` stmts)

```
-----------------------------------------------------------
Stream Counts (1428894 hits total)
-----------------------------------------------------------
%hits #hits  #inst  name
 4.0 57416 [16049] Always stmt (file: /tmp/myproject/dsp/rtl/dsp_delay.v, line: 65 in worklib.dsp_delay [module
 3.8 53586 [  703] Continuous Assignment (file: /tmp/myproject/gen/rtl/gen_mux_ohot.v, line: 53 in worklib.gen
 2.0 29182 [16952] Always stmt (file: /tmp/myproject/dsp/rtl/dsp_convert.v, line: 326 in worklib.dsp_convert [m
 1.9 27688 [16049] Continuous Assignment (file: /tmp/myproject/dsp/rtl/dsp_delay.v, line: 76 in worklib.dsp_de
 1.8 25784 [ 3704] Continuous Assignment (file: /tmp/myproject/dsp/rtl/dsp_add.v, line: 184 in worklib.dsp_add
```

Total hits per `always` at line in file

Merged Profiles
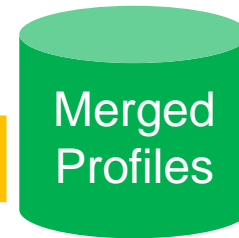
Profiles for each regression test
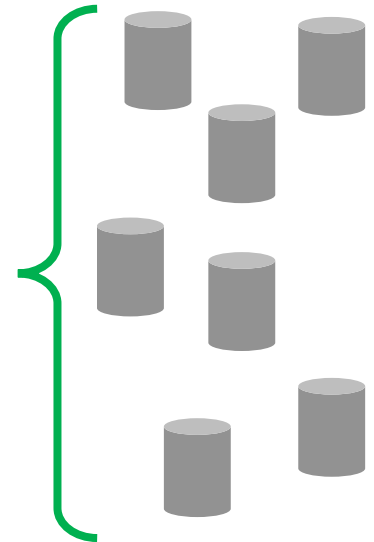
- Performance issues may only manifest in regression

- Most active code (hit rate) may be different for different tests

- Target most active code to optimize performance

cādence®

Digger Deeper with Single Run Profiling

- Rerun tests for instance-level performance details

- Determine if issue is in all or individual instances

- Modify code, reprofile, rerun regression

Examine instances vs. modules

Examine contributing subblocks

Add Assertions to Monitor and Prove Performance

- **Shine the light on critical paths that must complete within a max clock count**
  - Critical datapaths
  - Resource request/grant combinations
  - Asynchronous requests delay critical path

$rose(req) |=> ## [M:N] $rose(*ack*)

rising edge of "ack" must occur between M and N clocks after risking edge of "req"

- **Shine the light on streaming invariants**
  - Resource must never livelock/deadlock

Best done with advanced lint checking

- **Simulation is good, formal is better**
  - Simulation depends on stimulus for all cases
  - Formal proves all cases without stimulus

cādence®

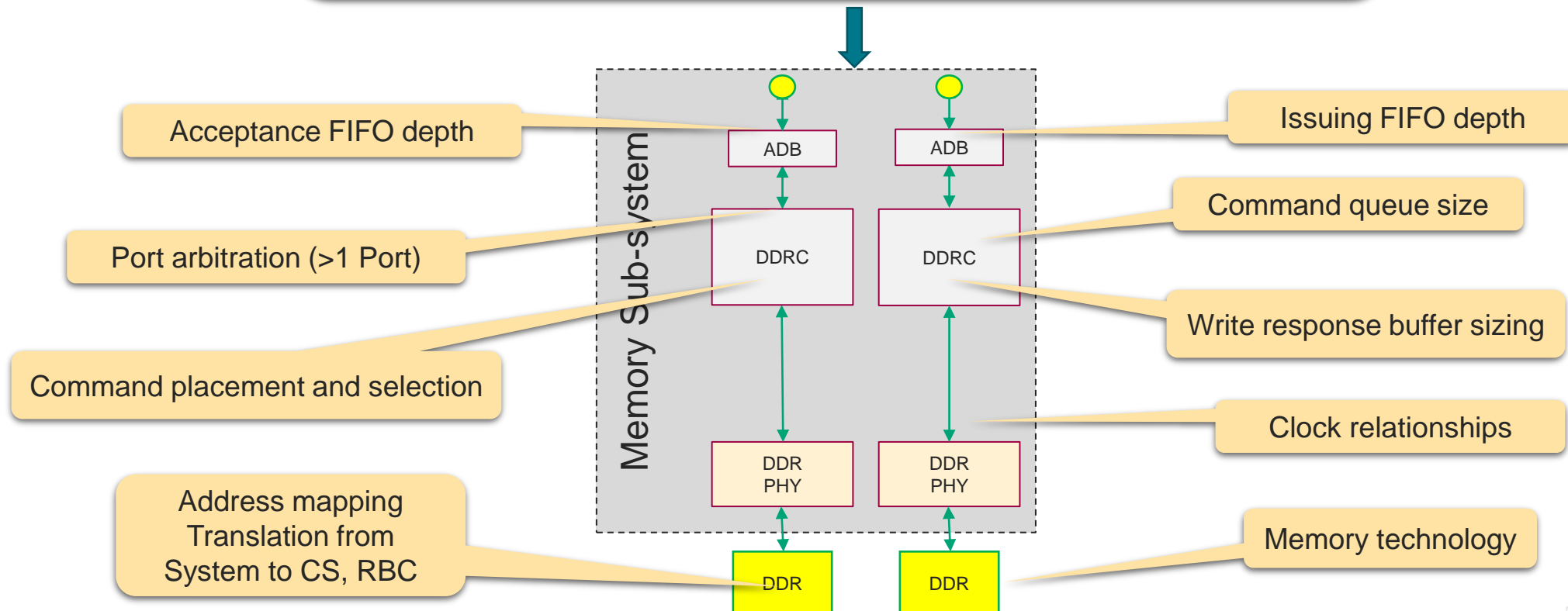# Examine Data Transport Efficiency Between System Layers

- Layering system distributes tasks to multiple designers

- Interfaces assure clear communication among teams

- Inefficiency can occur if sequential layers recreate intermediate calculations

- Refactor interfaces if intermediate calculations are shared

cādence®

Memory Subsystem Performance Challenges
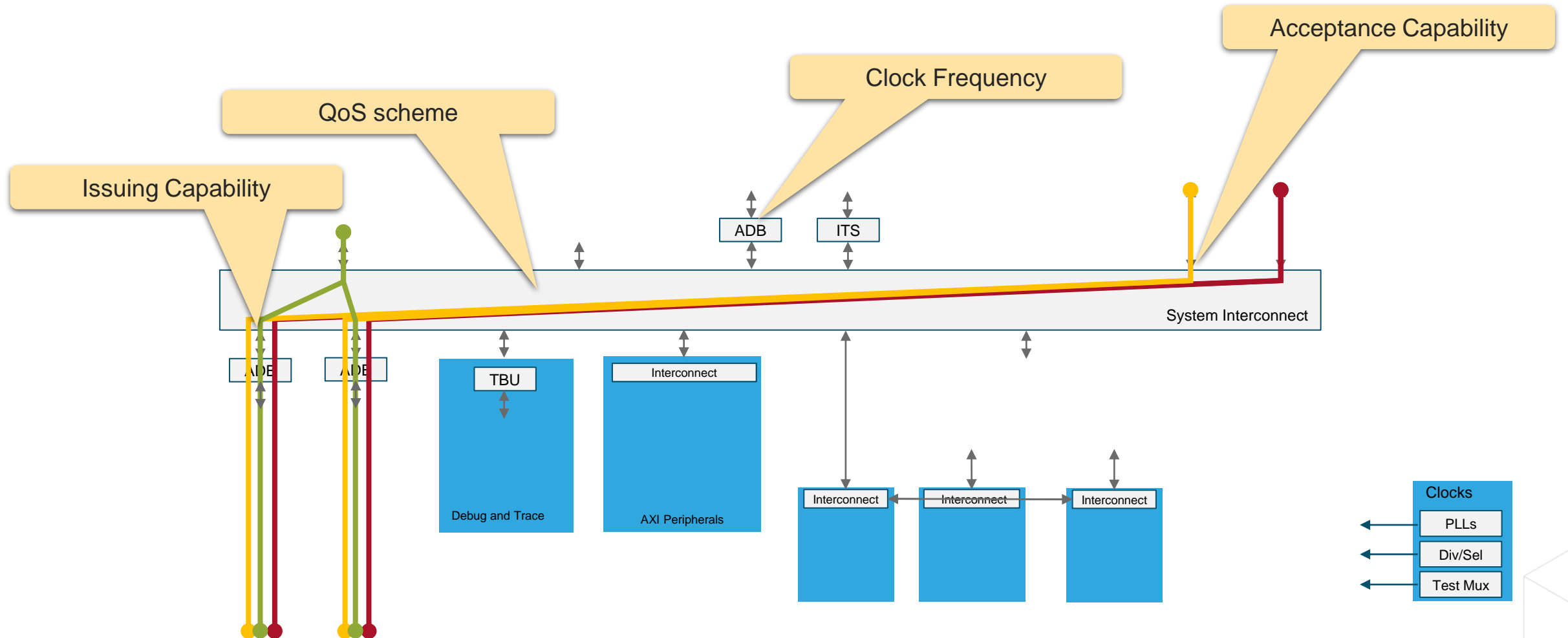## Degradation Causes

System
- Inefficient ID reuse - resulting ID collision
- Traffic causes read-modify-write memory access
- Load balancing
- Quality of Service (QoS)
- Cacheable and Bufferable transaction attributes (posted / non-posted traffic)

Memory Sub-system

Acceptance FIFO depth

Issuing FIFO depth

ADB

ADB

Command queue size

Port arbitration (>1 Port)

DDRC

DDRC

Write response buffer sizing

Command placement and selection

Clock relationships

DDR PHY

DDR PHY

Address mapping
Translation from
System to CS, RBC

Memory technology

DDR

DDR

cādence®

Interconnect Performance Challenges
Degradation Causes

Issuing Capability

QoS scheme

Clock Frequency

Acceptance Capability

ADB

ITS

System Interconnect

ADB

ADB

TBU

Interconnect

Debug and Trace

AXI Peripherals

Interconnect

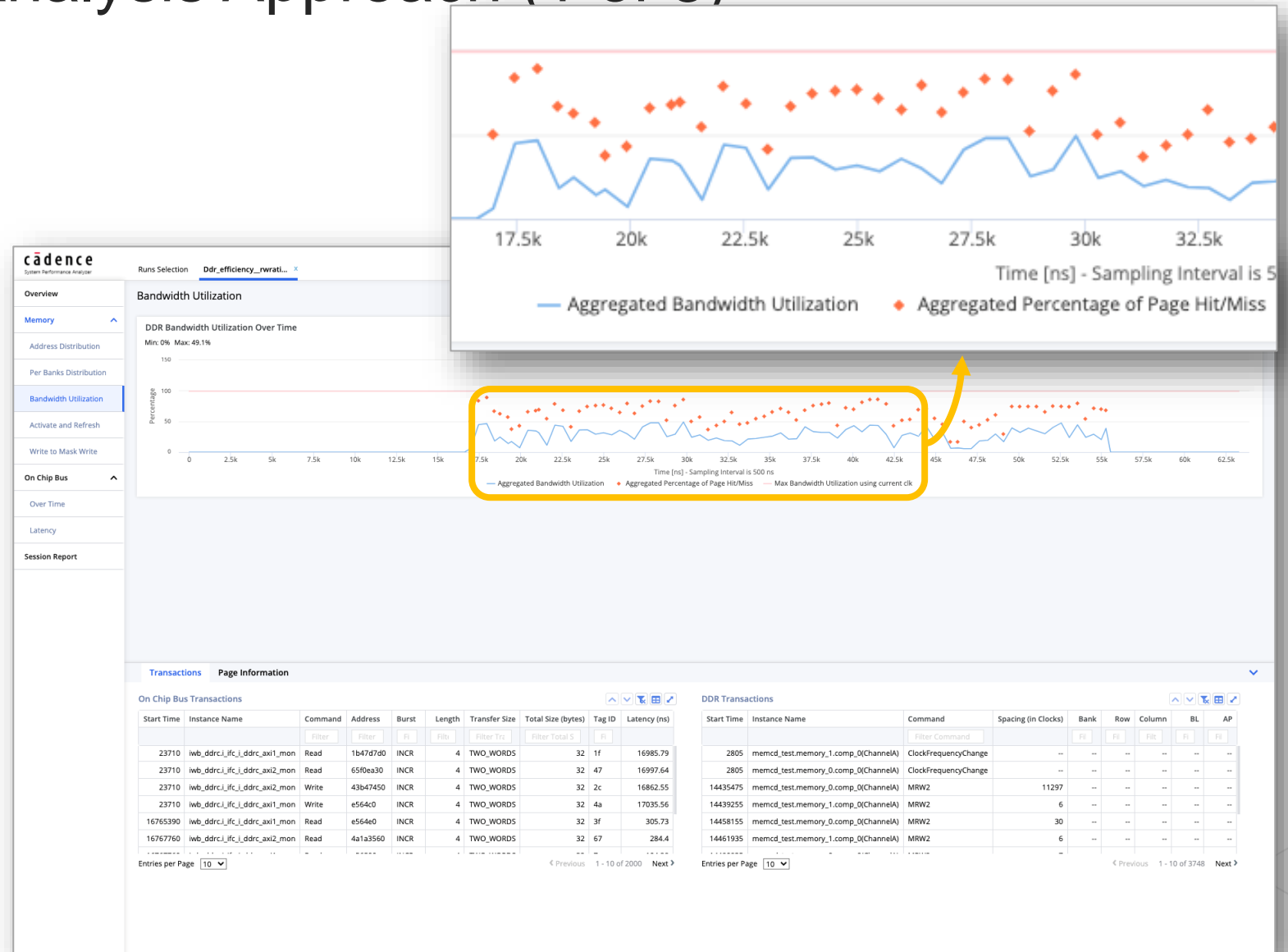Interconnect

Interconnect

Clocks

PLLs

Div/Sel

Test Mux

cādence®

System Performance Analysis Approach (1 of 3)
Memory DDR utilization

- With page hit/miss indication

- Max utilization line indicates the system potential for the current clock frequency

- Table of DDR commands list all commands in the current time window

System Performance Analysis Approach (2 of 3)
## On chip bus over time analysis

- Quickly understand the relationship between:
  - Bandwidth over time
  - Latency over time
  - Outstanding transaction over time

- Allows bottlenecks to be identified and investigated
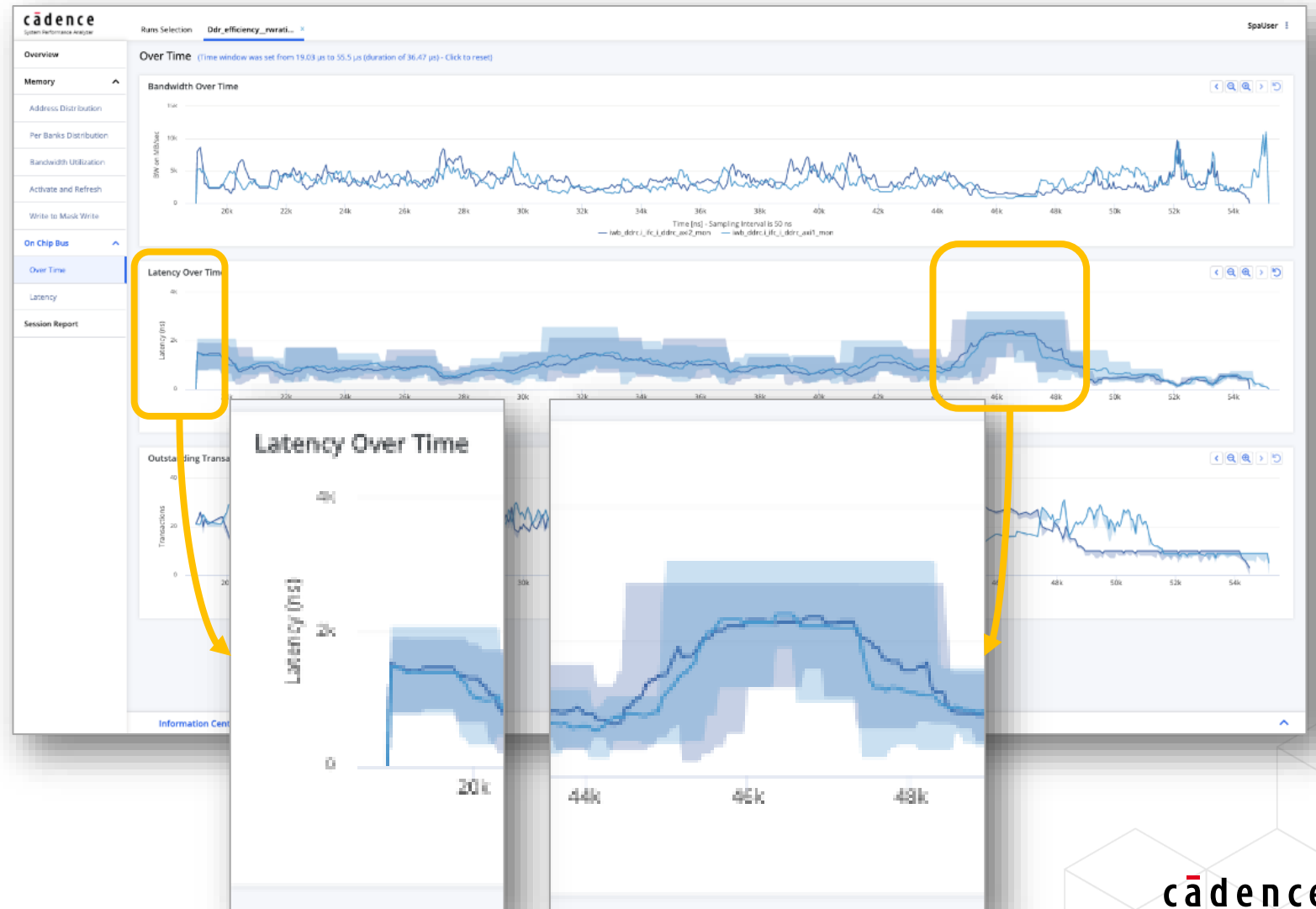
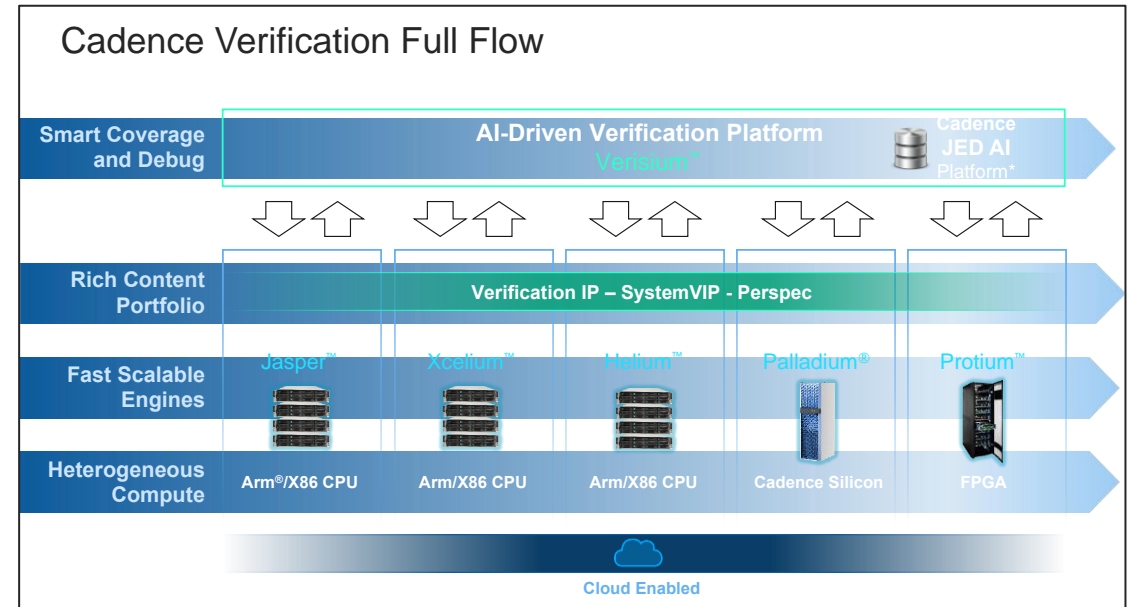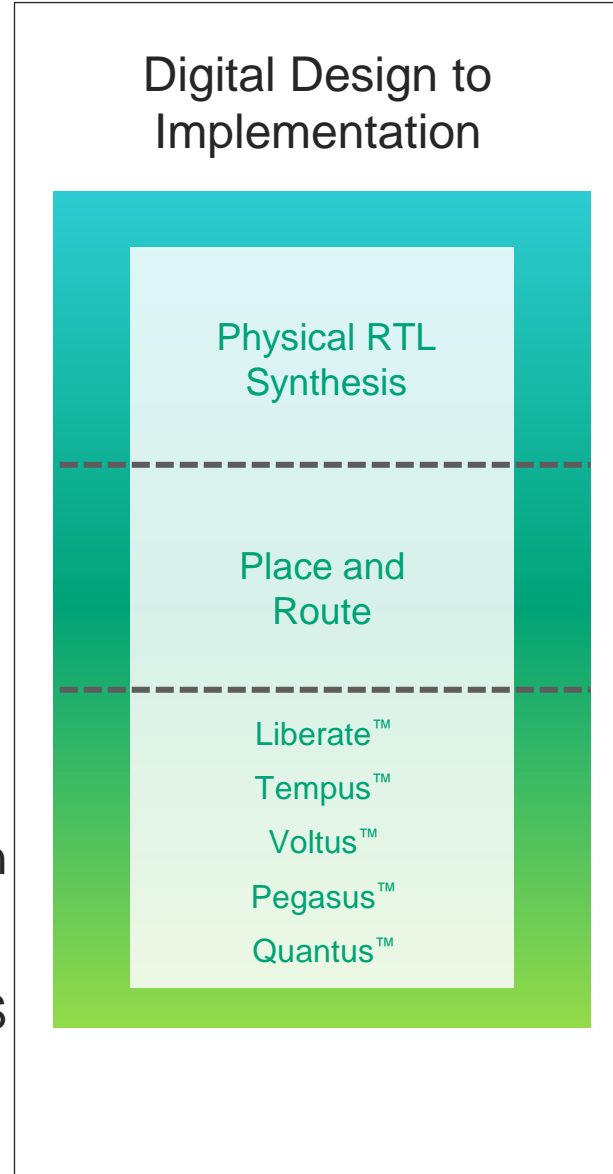# System Performance Analysis Approach (3 of 3)
On chip bus latency analysis

- Quickly identify outlier transactions with high latency and investigate the time period when they occur

- In all three analysis examples UVM or similar testbench messaging and/or assertions/checkers should be used to identify suspicious tests for analysis

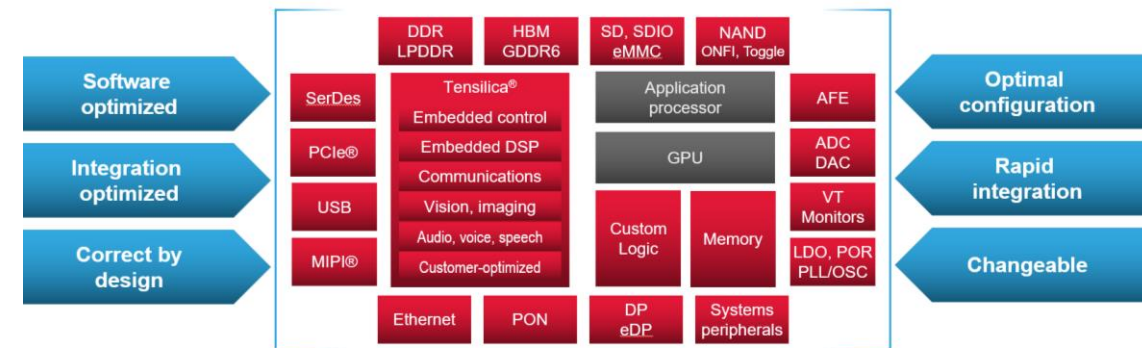Correlate latency outliers for further analysis
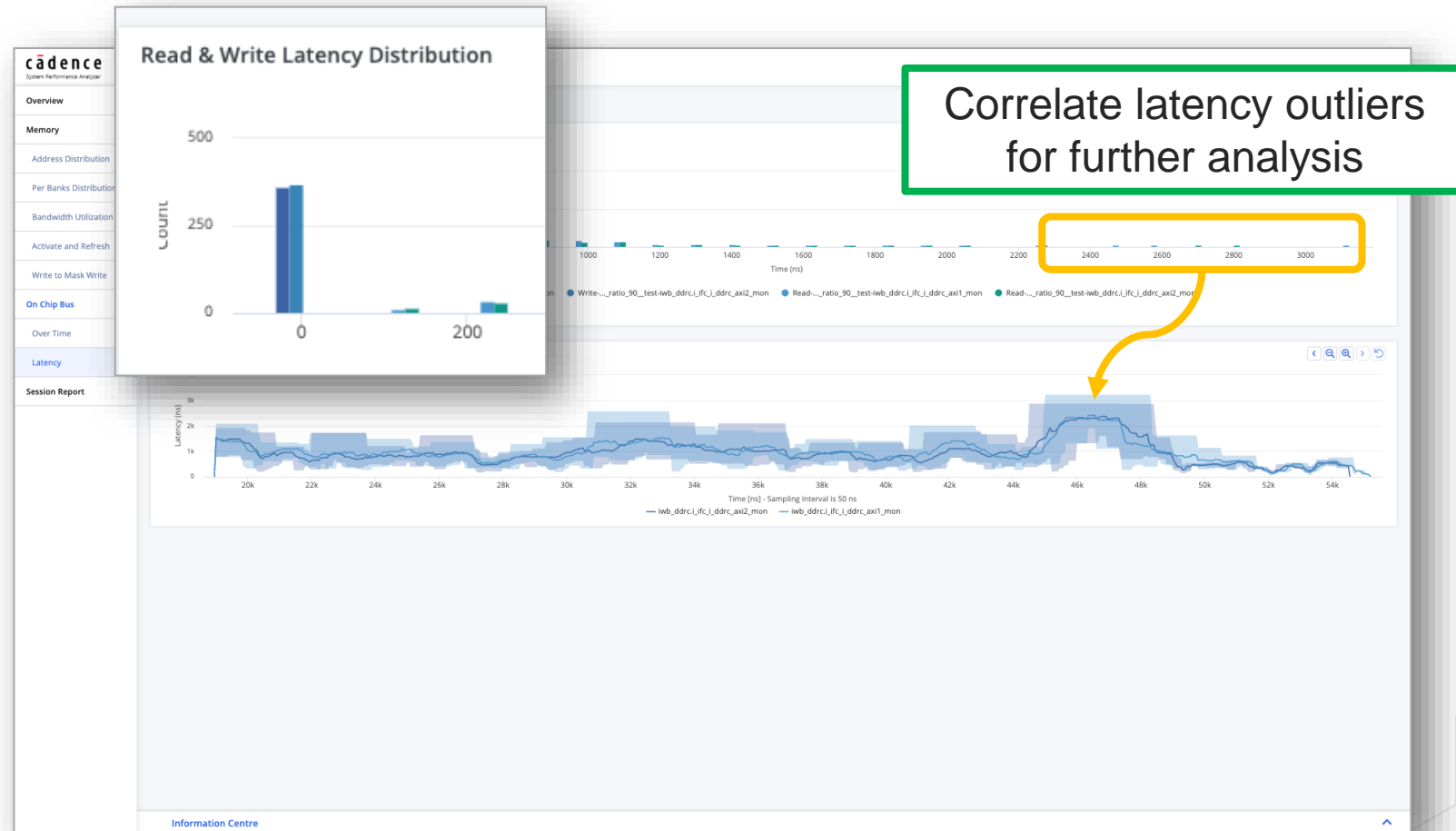
cādence®

Cadence: Your ASIC/FPGA Partner

- Verification solution: Apply objective analysis to improve FPGA and prepare for ASIC

- ASIC solution: Broadly adopted in high-speed comms and mission-critical applications

- Tensilica® IP: Proven processor technology used in autonomous drive and other high-reliability apps

- High-performance IP: Proven in leading comms systems

- Services: Expert RTL to GDS design services

Digital Design to Implementation

Physical RTL Synthesis

Place and Route

Liberate™
Tempus™
Voltus™
Pegasus™
Quantus™

Cadence Verification Full Flow

Cadence IP Solutions
Silicon-proven in advanced nodes

Cadence extensive Design IP, Verification IP (VIP), Tensilica® IP, and memory models to ensure complex SoC designs correctly on first pass

cādence®

Call to Action: Shine A Strong Light on Performance

- Data, data, data!
  - Performance analysis requires simulation cycles to generate data…
  - … except where formal analysis can be applied

- Profile, profile, profile!
  - Functionality and performance go together for HFT
  - Functionality without performance can be a competitive disadvantage
  - Performance without functionality can be an even higher risk!

- Start simple and build up
  - Add assertions – these will help debug as you rerun suspected lab sequences in simulation
  - Make profiling a design review task – ask designers to comment on performance analysis
  - Build to formal methods and system performance analysis ➔ especially for ASIC!

cādence®