VP Performance Optimization

Rocco Jonack
Juan Lara Ambel

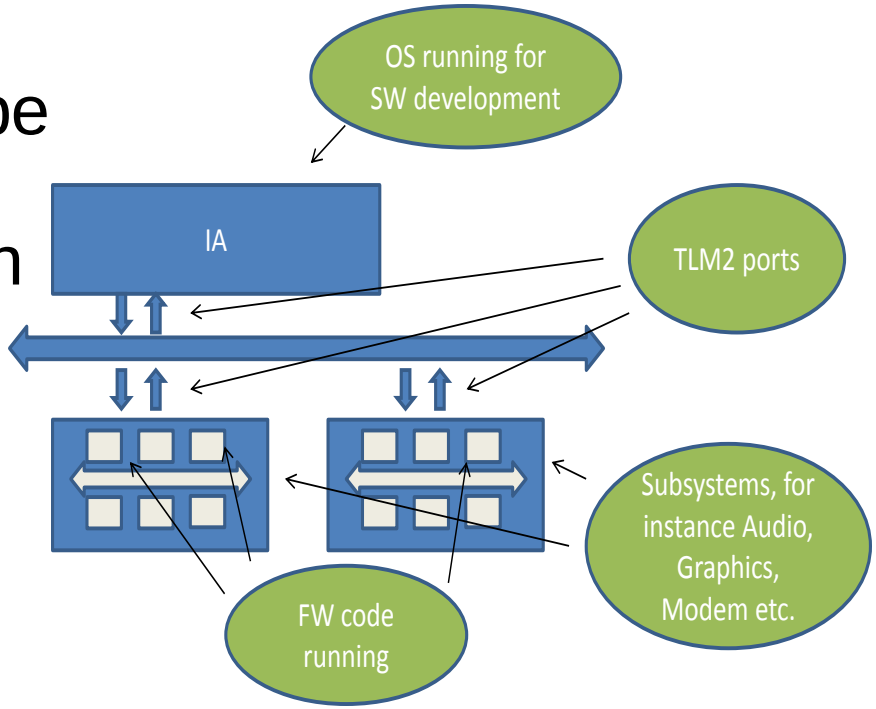
Intel

Contents

- Overview
  - VP development
- Runtime optimization for VP
  - TLM2 LT
  - Temporal decoupling
  - Profiling
  - Native execution
- Performance in heterogeneous SoC models.
  - Measurement scenarios
  - Results

VP Overview

- Virtual Platforms are software models of hardware architecture on which software and firmware can be executed
- Many factors have impact on VP creation
  - Functional correctness
  - Debuggability/visibility
  - Usage model
  - Execution speed
- VP often crosses different components and modeling styles

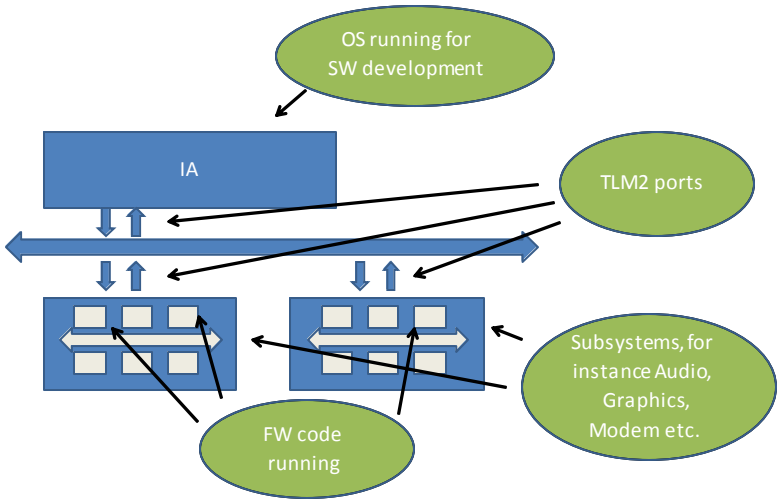


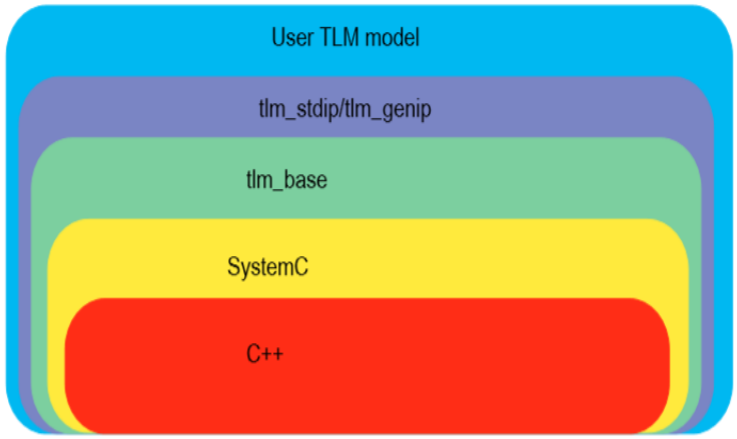
VP runtime optimizations

- Usage of transaction level modeling reduces runtime and modeling effort
- Using quantums consistently allows trade-off between timing accuracy and runtime
- Profiling allows analysis and focus on hotspots
- Using native execution on host machine or through virtualization can improve runtime dramatically

TLM2 modeling

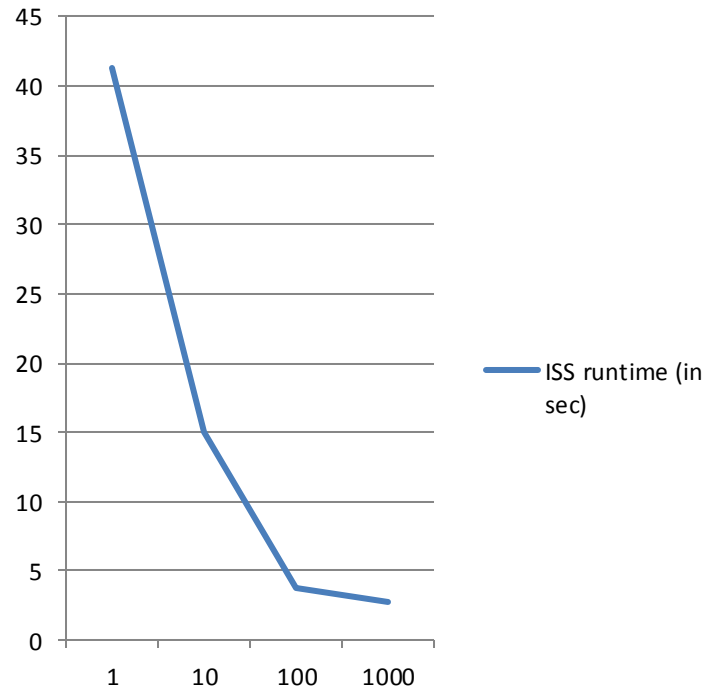
- Using TLM2 modeling allows a standardized modeling approach
  - Using LT modeling style reduces event overhead
  - Reuse of existing components whether 3rd party or inhouse
  - Many existing models are tested for simulation performance
  - ISCTLM extends the available models with preverified components for VP modeling





Quantum usage

- Temporal decoupling avoids switching between simulation threads
  - Initiator threads are running temporarily „out of sync“ with respect to simulation time
- Fewer simulation events, but also fewer synchronization points
- Different environements have different implementations of temporal decoupling
  - Keep synchronization time consistent
  - TLM2 provides coherent infrastructure which defines models like a global time keeper etc.

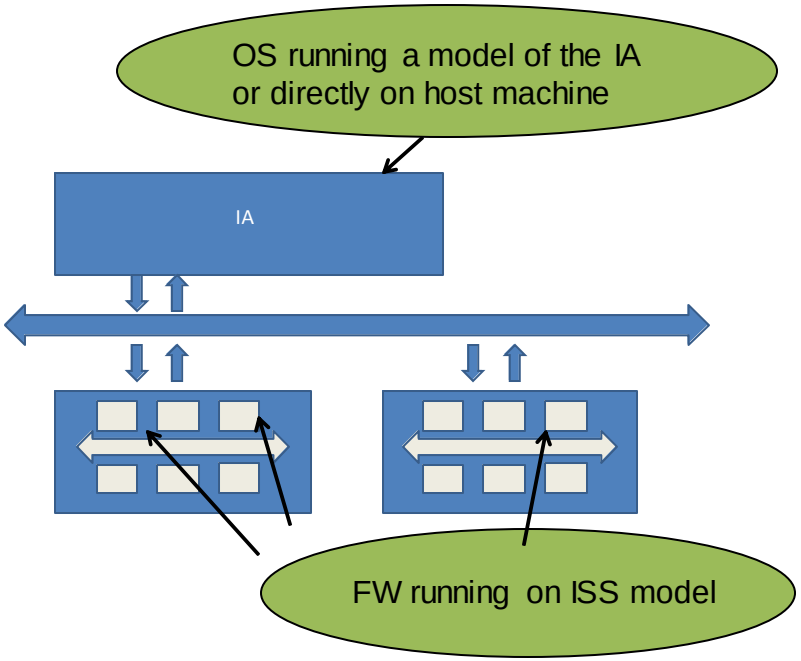


Profiling

- Runtime bottlenecks typically come from well known types of modeling constructs
  - Streaming to IO devices
  - Excessive memory allocation or inefficient memory access
  - OS interaction like thread switching
- Profiling tools help expose bottlenecks
  - Can be difficult to find in a complex model, developed across teams and incorporating 3rd party models
  - Functional profiling tools like gprof, vtunes, googleperf focus on counting function calls during execution
  - Thread profiling focused on analyzing the when and by which thread a switch was introduced

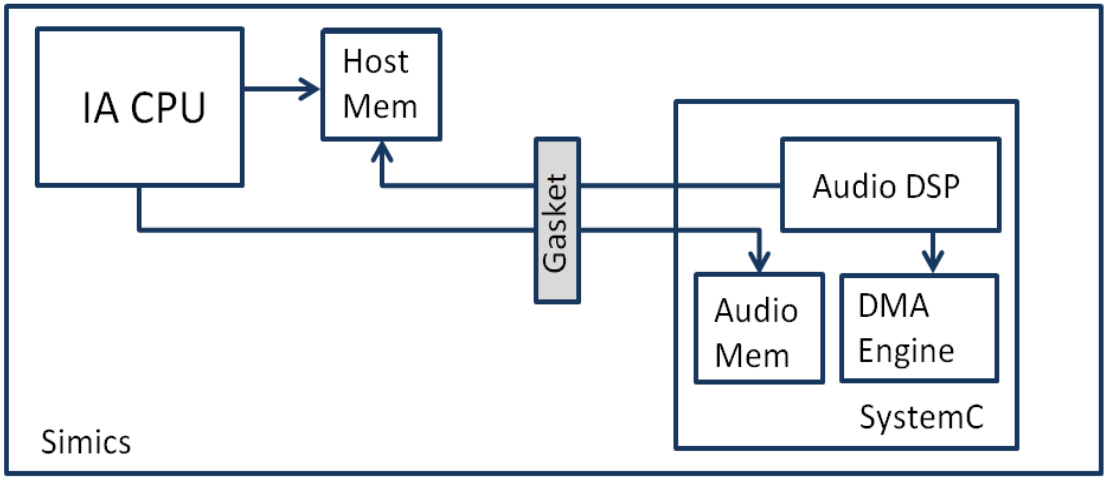
Native execution

- Running code natively on host machine can dramatically improve runtime
  - With virtualization an OS can access devices through virtual access points
  - Subsystems often don‘t support virtualization, but ISS can optimize code execution by executing code on the host system



Performance in Heterogeneous SoC Models

- A mobile platform SoC contains several subsystems executing SW or FW, which can be modelled as individual VPs.
- Potentially, an integrated VP spans multiple modelling styles.
- Case study: platform´s CPU cluster modelled in Simics‘ DML integrated with a TLM2/ SystemC model of the Audio Subsystem.

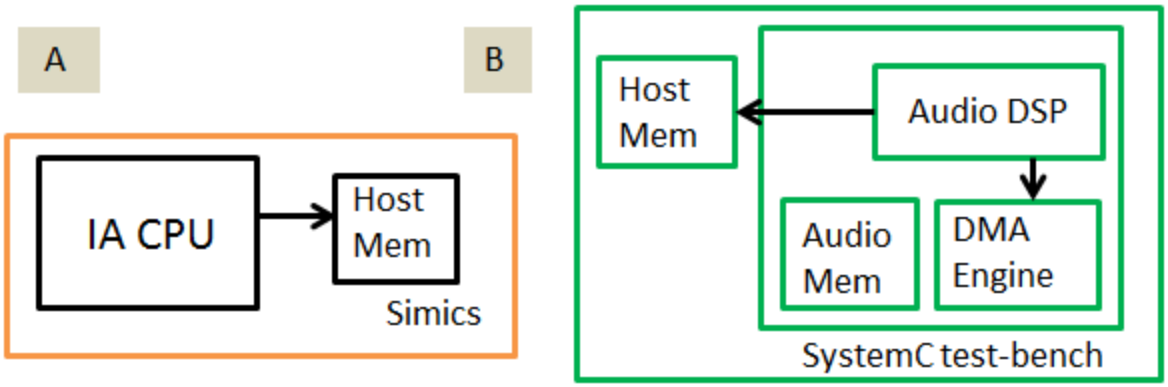


Configuration

- Transaction-initiation patterns used in scenarios
  - Used patterns span from maximum load to load in regular operation and were estimated after specifications.
  - Transaction-traffic at the Simics-Audio interface (if applicable):
    - Audio initiated: 45K, 250K, 1500K and 4800K-accesses/sec.
    - IA CPU initiated: 1K, 2K and 3K-accesses/sec.
  - Transactions initiated by Audio DSP to Audio registers (if applicable):
    - 500 K-accesses/sec.

- Measurement Setup
  - Simics running Viper VP including IA CPU model.
  - Used Quantum: 10e4 cycles for Audio DSP (= 100 us), 100 us for Simics.
  - Simulated on 1 execution threads, VMP disabled.

Standalone scenarios



- A) *Simics standalone*: IA CPU boots Linux (busybox)
- B) *SystemC-Audio standalone*: splits into B1,B2 & B3 depending on executed Audio DSP FW.
  - B.1: FW endlessly does some math but there is no significant number of accesses to either internal or external memories & registers.
  - B.2: FW endlessly accesses a series of internal Audio-control-registers. The access rate can be modulated.
  - B.3: FW endlessly accesses a memory block external to Audio-subsystem.

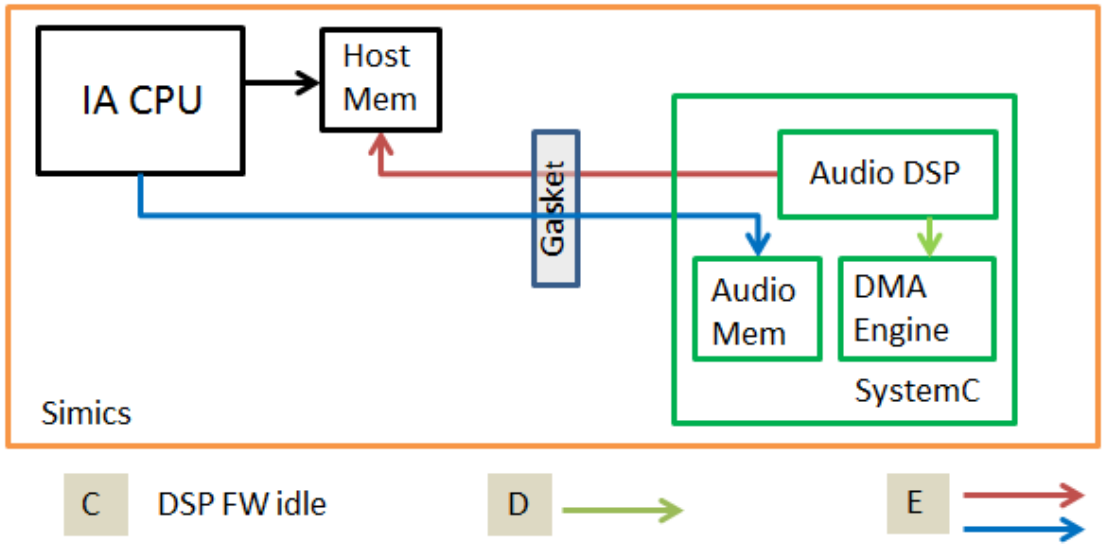
Measurements results - A,B.x

Average Real Time Factor (***RTF):***

- Defined as Wall-clock Time (WT) over Virtual Time (VT).
- Initialization time discounted.
- Measured when busybox booted (over 8,9 sec. VT), or over 1 sec. VT for Audio standalone.

	RTF (optim.)	***RTF***
A, Simics standalone, booting busybox	2,7	*n/a*
B.1, Audio standalone, number crunching	1,6	415
B.2, Audio standalone, internal-reg. traffic	17,5	613
B.3, Audio standalone, Audio→host-mem traffic	1,3 (4,8 MAcc./ sec)	667

Integration scenarios



- C) IA CPU boots Linux. No other transaction-traffic at all (~=B.1).
- D) IA CPU boots Linux. R/W transaction-traffic ***inside*** SystemC-Audio (*Green*) (~=B.2).
- E.1) IA CPU boots Linux. SystemC-Audio initiates R/W transaction- traffic ***through*** gasket to Host Memory (*Red*) (~=B.3).
- E.2) IA CPU boots Linux ***and*** accesses Audio-Memory through gasket. SystemC-Audio initiates heavy R/W transaction-traffic ***through*** gasket. (*Red + blue*)

Measurements results - C,D,E.1,E.2

Average Real Time Factor (***RTF):***

- Defined as Wall-clock Time (WT) over Virtual Time (VT).
- Initialization time discounted.
- Measured when busybox booted (over 8,9 sec. VT).

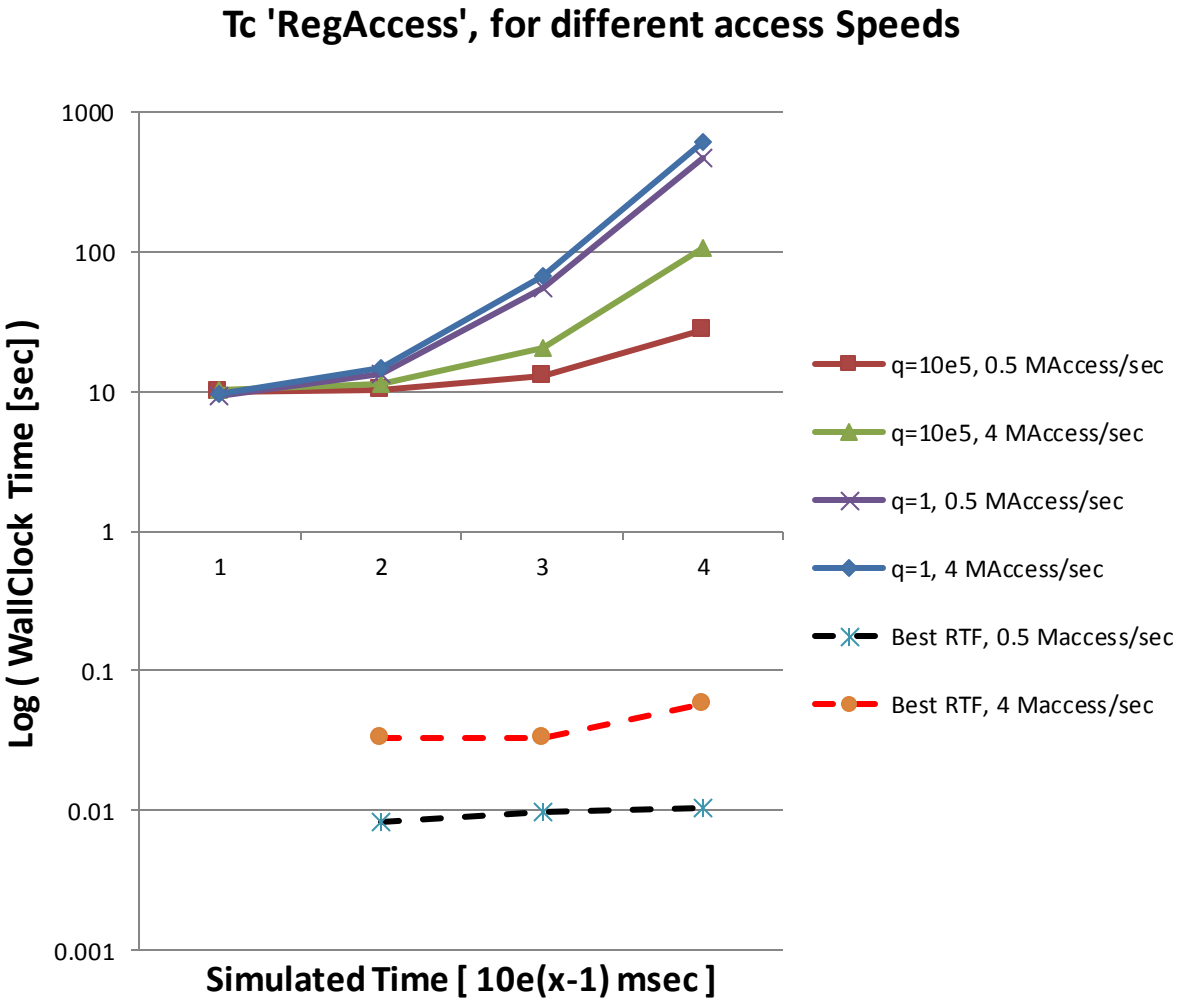
	RTF
C, integration, number crunching in Audio	3
D, integration, internal Audio traffic	12
E.1, integration, Audio → host-mem over gasket	3,1 (4,8 MAcc./sec)
E.2, integration, Audio ←→host, traffic over gasket	3,9 (4,8 MAcc./sec, 3 KAcc./sec)

Conclusions

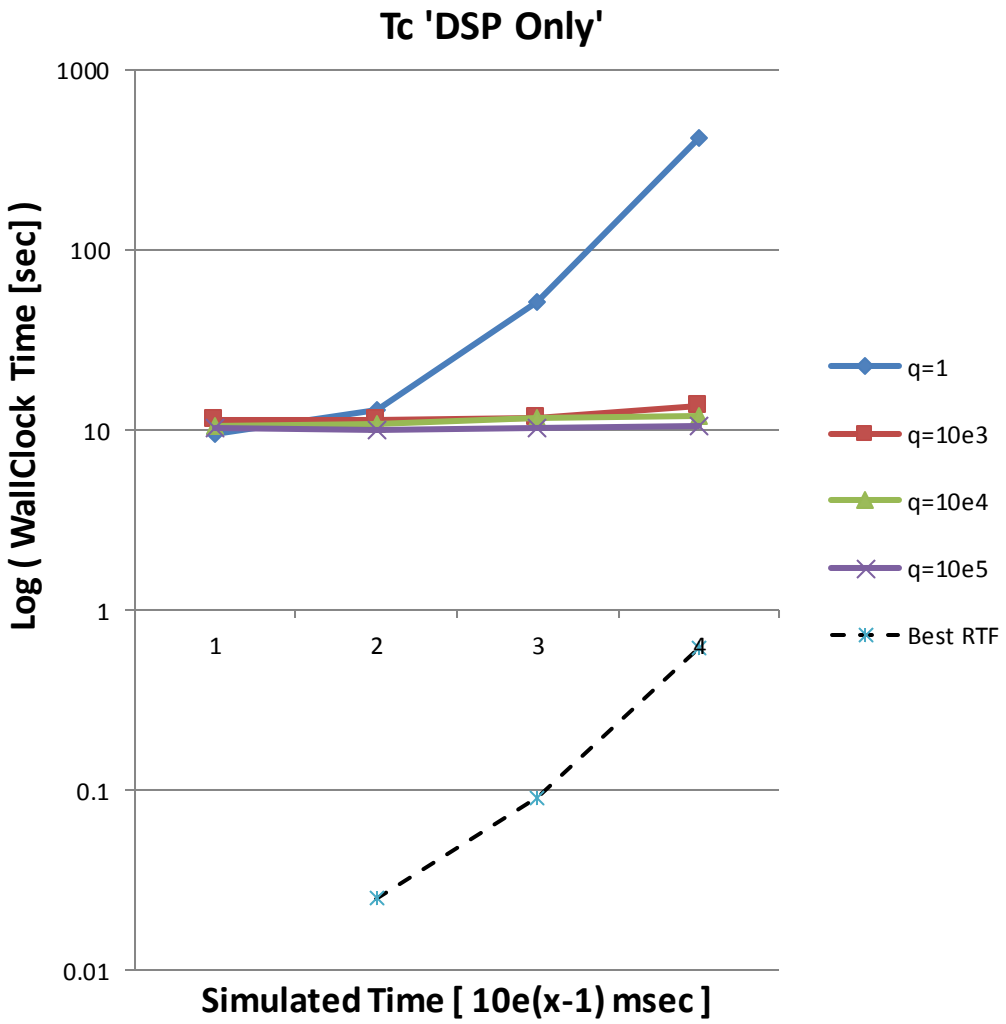
- Applied transaction level modelling and temporal decoupling techniques
- Result:
  - RTF in integration scenario lies in range: 3 - 12.
  - Linux (busybox) booting in integration scenario takes actually approx. 30-35 seconds wall-clock time for traffic-representative use-cases.
- Performance in integration scenario allows fast turn-around times for SW & driver development.

Questions

Detailed results for B.2

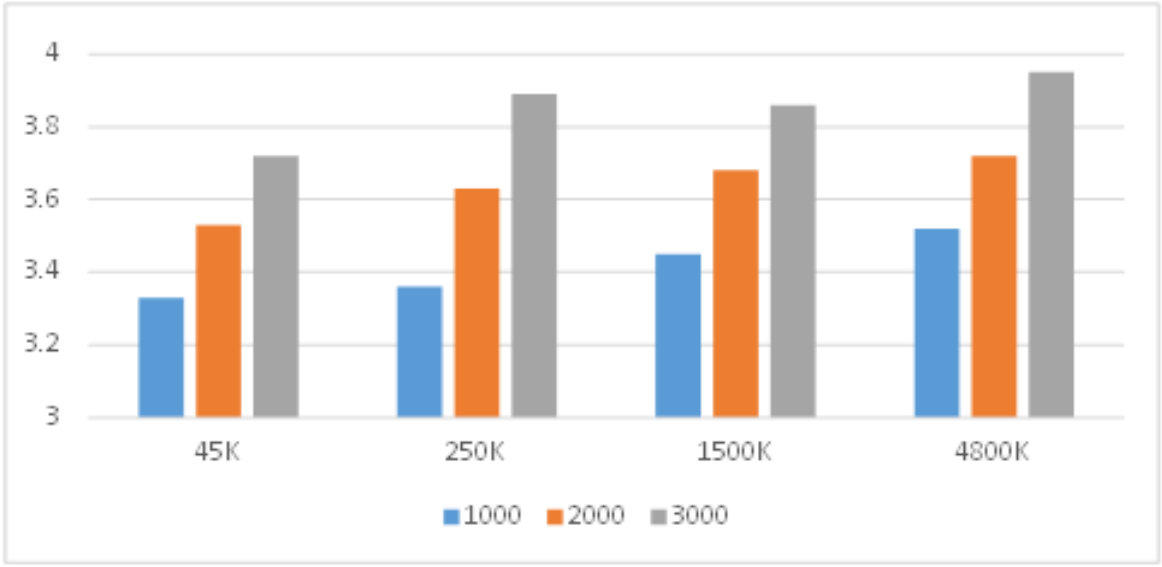


Detailed results for B.3



Measurement results - on E.2

Average RTF for combinations of IA-CPU-initiated transaction traffic (color code) versus Audio-initiated transaction traffic (x-axis)



- RTF oscillates among 3,3 and 3,9.
- Transaction traffic through gasket initiated by IA-CPU is much more expensive than those initiated by Audio (Asymmetry of the gasket)

Simics Gasket