Benchmarking Real-World In-Vehicle Applications

NVIDIA GTC 2015-03-18

mycable GmbH
Michael Carstens-Behrens
Gartenstraße 10
24534 Neumuenster, Germany
+49 4321 559 56-55
+49 4321 559 56-10
mcb@mycable.de

Rev. A

Agenda

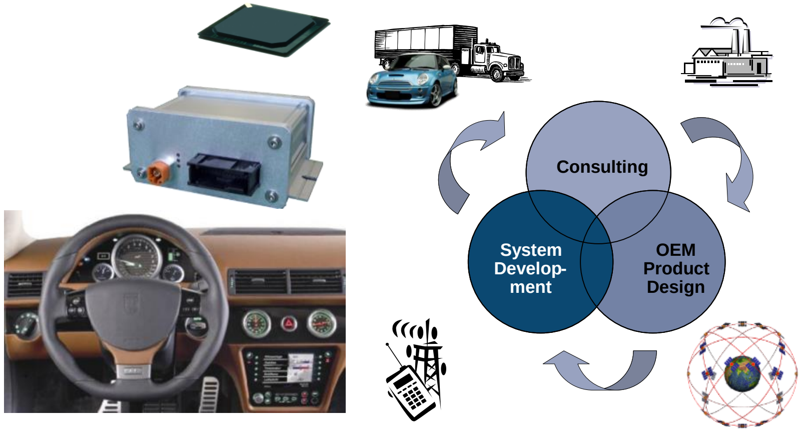
- High-End embedded Processor System Design Challenge
- Theoretical Benchmarking Approach
- Practical Approach for critical Use Case Benchmarking
- Application specific Performance Optimization with Tegra
- Future System Integration Challenges
- Q&A

High-End embedded System Design Challenge

Background

- mycable defines and develops new Technologies for automotive and industrial Applications since 2001
  - Based on high performance, highly integrated processor systems
- Application specific performance is crucial for each processor subsystem

Automotive	Industrial
Instrument Cluster	Digital Radar
Infotainment	3D Measurement
EV Connectivity	Optical Inspection
…	…



Design Challenges

- **Available Performance Figures does not help**
  - **CPU: Frequency, DMIPS, MFLOPs, etc.**
  - **GPU: GFLOPs, Mpoly/s, Mpixel/s, etc.**
  - **Even standard or application benchmarks often do not help**
    - SPECint, EEMBC, etc.

- **Application specific Performance is very specific and depends on many Resources**
  - **Of course on CPU and GPU performance, but also on**
    - Volatile and non volatile memory size, performance and latency
    - Internal/external bus and interface bandwidth/arbitration including full path to memor
    - Cache sizes and architecture
    - Further acceleration units like hardware crypto acceleration, DSP, etc.

Further Design Challenges

- A Proof of Concept is usually not possible because
  - The system-on-chip you intend to use is not yet available
  - Evaluation system does have the interfaces of your application
  - Your application software is not yet ready
- Nevertheless Time is always critical in your project!
- A Model-based Approach can help you to analyze your application specific Performance on a not-yet-existing Processor Subsystem
  - By building a system resource model
  - By building an application specific software benchmark

Theoretical Benchmarking Approach

System Example

- A High-End Infotainment System is selected as an Example for Building a System Resource Model
  - An infotainment system is a good “worst case” example due to its combination of many tasks with different system resource demands
  - System use cases combining performance critical tasks with excessive usage of system resources

- Methodology can be transferred to any other system type

Theoretical Benchmarking Approach

❶ **Define your elementary high Level Tasks**
- Top level tasks running on your system combined in use cases

❷ **Create a System Resource Model from your System Architecture**
- Simplified architecture with focus on performance relevant resources

❸ **Quantify the particular Resource Demand for elementary Tasks**
- By calculation, estimation, simulation or extrapolation

❹ **Benchmark your System by combining elementary High Level Tasks to System Use Cases**
- Detect mismatches for critical resource usage like CPU performance, memory performance, bus bandwidths, and others

❶ Define your elementary high Level Tasks

System
- scheduler
- drivers
- services
- display output
- file system, de/encryption
- …

3D Navigation
- positioning
- dynamic route calculation
- 3D map rendering
- traffic information proc.
- database access
- …

HMI
- 3D HMI rendering
- HUD rendering
- speech recognition
- text to speech
- list handling
- …

Entertainment
- H.264 video decoding
- audio decoding
- DRM handling
- audio processing
- video transcoding
- …

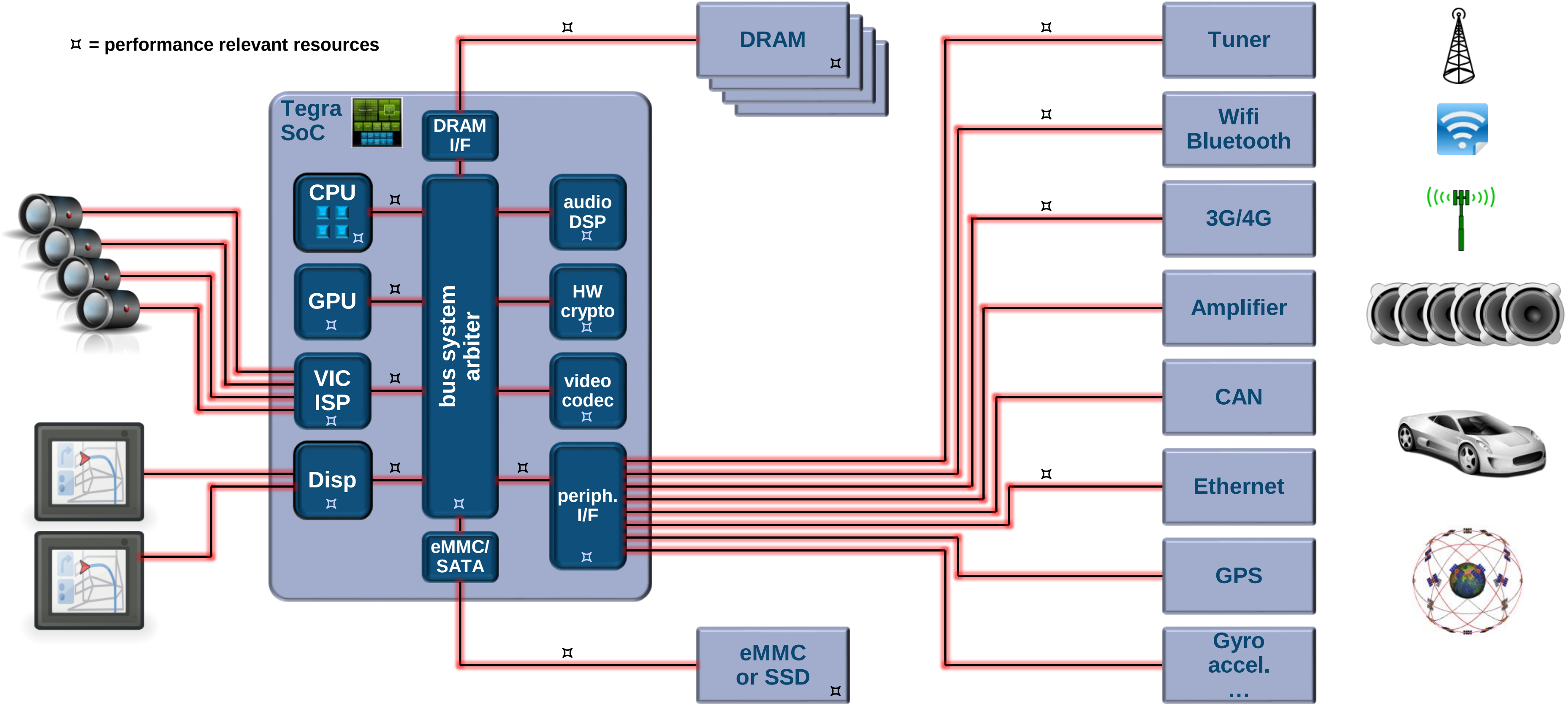
Connectivity
- hands free telephony
- 3G/4G connectivity
- WiFi AP
- gateway functionality
- data de-/encryption
- …

Driver Assistance
- camera input
- image pre-processing
- object recognition
- camera view compositing
- warning generation
- …

…

❷ Creating a simplified System Resource Model

⌑ = performance relevant resources



❸ Quantify particular Resource Demand for Tasks

cache hit rate to be considered →

		CPU MIPS	GPU GFLOPS	DRAM MB	DRAM bus MB/s	DSP MIPS	eMMC MB	SDIO MB/s	HW crypto MB/s	video dec. Mb/s	video enc. Mb/s	Ethernet Mb/s	USB #1 Mb/s	USB #2 Mb/s	…
3D Navigation	positioning	500	0	20	100	0	0	0	0	0	0	0	0	0	…
	dynamic route calc.	2800	0	30	100	0	2G	6	0	0	[illegible]	0	0	0	…
	3D map rendering	1500	100	650	3200	0	26G	12	0	0	0	0	0	0	…
	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
HMI	3D HMI rendering	1200	70	400	800	0	2G	2	0	0	0	0	0	0	…
	speech recognition	300	0	50	100	5000	8G	4	0	0	0	0	0	0	…
	display output	0	0	20	300	0	0	0	10	0	0	0	0	0	…
	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

- examples -

don't forget system load

to be extended ↓

to be extended →

❹ Benchmark your System by System Use Cases

Use Case #1
- driver seat: 3D navigation
- passenger seat: browser
- internet connectivity
- hands-free telephony
- audio processing
- …

Use Case #2
- driver seat: 3D HMI
- passenger seat: Blu-Ray playback
- speech recognition
- eye tracking
- …

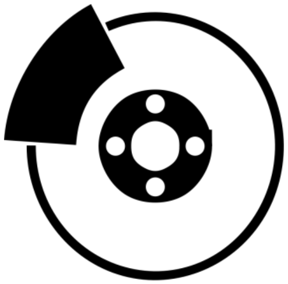
Use Case #3
- driver seat: 3D navigation
- passenger seat: 3D HMI rendering
- traffic sign recognition
- text to speech
- …

	CPU MIPS	GPU GFLOPS	DRAM MB	DRAM bus MB/s	DSP MIPS	eMMC MB	SDIO MB/s	HW crypto MB/s	video dec. Mb/s	video enc. Mb/s	Ethernet Mb/s	USB #1 Mb/s	USB #2 Mb/s	…
Use Case #1	16000	110	2.8G	28000	8500	56000	35	64	12	0	24	0	0	…
Use Case #2	11000	130	1.9G	11000	5200	35000	28	33	28	0	11	0	0	…
Use Case #3	9500	210	2.1G	14000	3400	32000	16	12	0	0	17	0	0	…

- examples -

Pitfalls in estimating integer Performance

- **Maximum raw integer of a quad Core A57, L1 Cache hit @ 2 GHz**
  - **4 x 2000 MHz x ~4.5 MIPS/MHz = 36.000 MIPS**
- **But single Thread Cache miss performance is different**
  - **Actual performance depends on external memory speed**
  - **May simply drop down to a few k MIPS**
    - E.g. 1 x 2000 MHz / 4 (ext. memory) x 4.5 MIPS/MHz = 2.250 MIPS
- **To be considered**
  - **Cache hit rates**
  - **Bus latencies, penalties, resource blocking**
  - **SMP mapping, …**



Practical Approach for critical Use Case Benchmarking

Practical Approach for critical Use Case Benchmarking

- Execution of individual benchmarks results in performance values which may not be reached if different applications share a resource in real life applications
  - Results may lead to too optimistic system architecture decisions

- Five Step Approach
  1. Isolate functions
  2. Find applicable benchmarks
  3. Estimate workload per task and scale benchmarks
  4. Execute benchmarks at the same time
  5. Find bottlenecks, Check system load, Test priority changes

Practical Approach: ❶ Isolate Functions

- **Isolate Functions from Critical Use Case Matrix which stress dedicated Resources from the System Resource Model**
  - ✓ **Already done in theoretical approach**

Ressource	HMI	Media-Database	Route-Calculation
GPU	+++		
CPU	+	+	+++
Memory	++	++	++
File-Access		+++	++

Practical Approach: ❷ Find Appropriate Benchmarks

- **Each Use Case can be represented by one or more benchmarks**

Resource	HMI	…	…
GPU	+++		
CPU	+		
Memory	++		
File-Access			

HMI-Simulation or -MockUp using elements and technologies as required: e.g. animations, transitions, OpenGL…

Standard-Benchmark like pmbw (parallel bandwidth memory benchmark)

Practical Approach: ❷ Find Appropriate Benchmarks

- **Isolate Functions from Critical Use Case Matrix which stress dedicated Resources from the System Resource Model**

Resource	…	Media-Database	…
GPU			
CPU		+	
Memory		++	
File-Access		+++	

e.g. filebench-benchmark with database-access profile

Practical Approach: ❷ Find Appropriate Benchmarks

- **Isolate Functions from Critical Use Case Matrix which stress dedicated Resources from the System Resource Model**

Resource	…	…	Route Calculation
GPU			
CPU			+++
Memory			++
File-Access			++

Route Calculation Benchmark performing typical algorithms on map-data

e.g. filebench with a defined multiple read access profile on a file profile fitting to typical map data

Practical Approach: ❸ Estimate Workload per Task

- **Create Configurations for:**

Resource	…	…	Route Calculation
GPU			
CPU			+++
Memory			++
File-Access			++

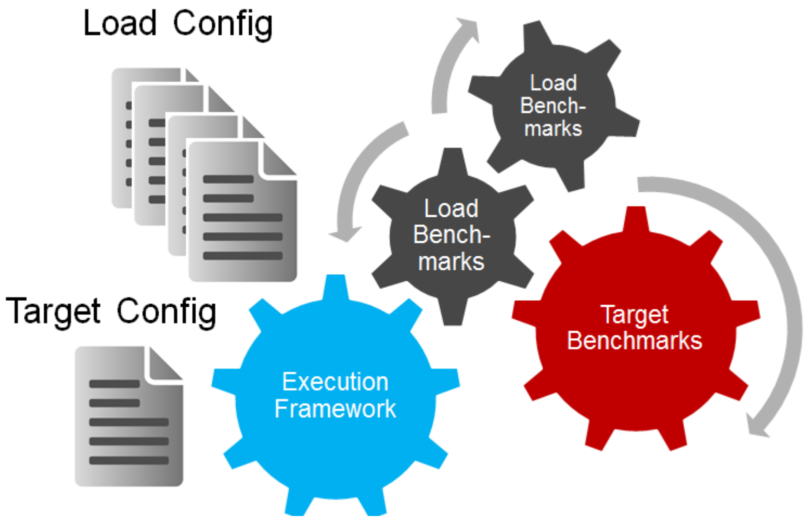
Route Calculation Benchmark performing typical algorithms on map-data

e.g. filebench with a defined multiple read access profile on a file profile fitting to typical map data

Average Load	Peak Load
Route Calc. Average Config	Route Calc. Peak Config
Route Calc. Filebench Average Config	Route Calc. Filebench Peak Config.

Next to the behavior (e.g. load, files, number of accesses) you can define scheduler priorities within the configuration

Practical Approach: ❹ Execute Benchmarks



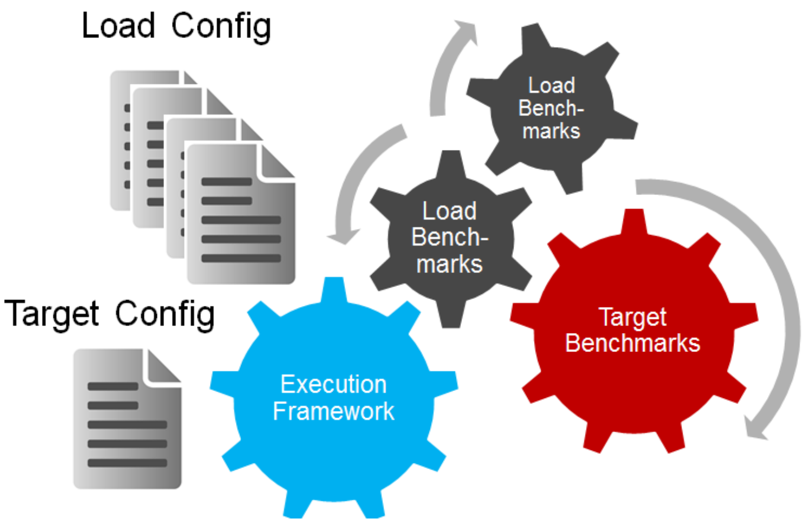
For fast reproduction and modification of results it is recommended to use an execution framework.

The configurations for all load benchmarks and for the target benchmark (the benchmark under test) is loaded by the framework and executed in always the same way.

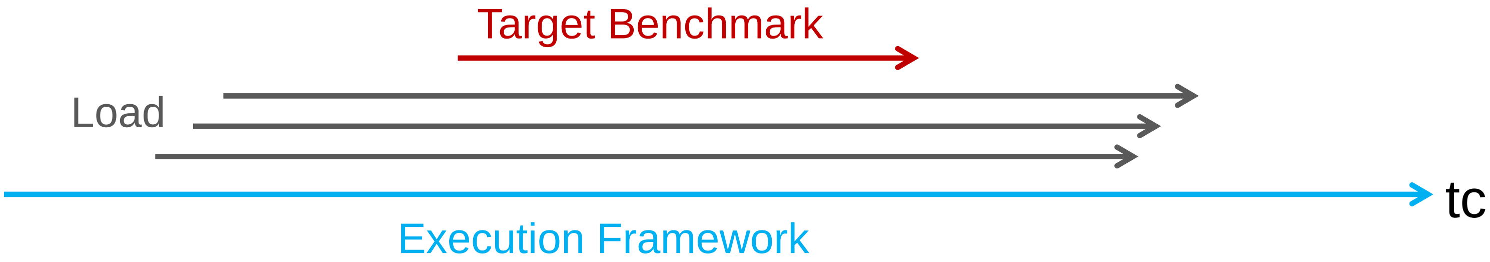
The framework:

- Checks the correct execution and detects hanging/dying processes
- Collects settings and results in a defined output format
- Informs with onscreen and webview about live values (e.g. fps)
- Is written in an easy portable script language (e.g. python)

Practical Approach: ❹ Execute Benchmarks



- The framework ensures reproducible results.
- The target benchmark runs on a specified system load.
- Each system resource can be stressed with a combination of real life usages
- Negative side effects can be disclosed



Practical Approach: ❺ The Results

- **Results and Opportunities**
  - **The results can be compared for different systems**
    - e.g. compare SoC generations
  - **Bottlenecks can be identified**
  - **Changes in current systems can be simulated in advance without generating/porting a whole system image/set of applications**
  - **Simulations of load profiles can be repeated on different architectures/SoCs with less porting effort (only framework and benchmarks)**
  - **Other benchmarks like EEMBC or SpecInt/SpecFP can be used as target benchmarks for specific performance analysis under specific load conditions**

Application specific Performance Optimization with Tegra

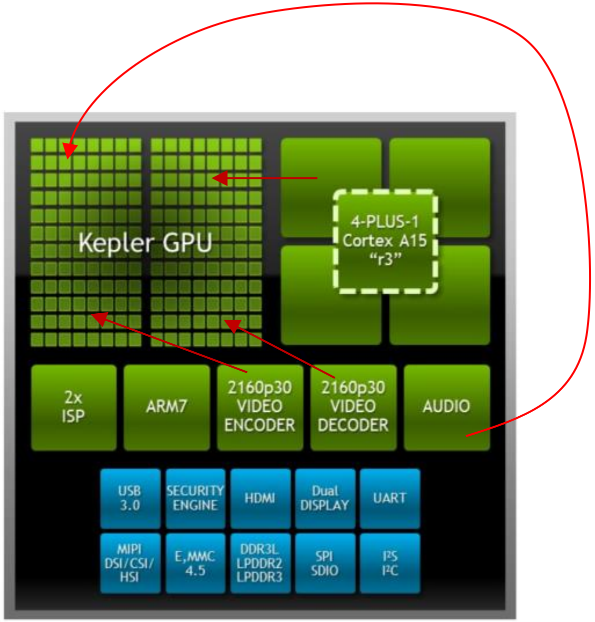
Application specific Performance Optimization with Tegra

- **Certain Tasks consume complete available System Resources**
  - **Route Calculation**
    - Uses all available integer CPU performance
  - **Speech Recognition**
    - Uses all available DSP performance
  - **Camera inputs, Display Outputs**
    - High permanent usage of memory bandwidth

- **But available GPU performance on Tegra Processors often exceeds application specific demand**
  - **Even high resolution 3D navigation and 3D HMI rendering does often not require full GPU performance**

Application specific Performance Optimization with Tegra

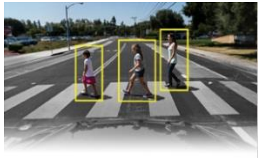
- Moving Tasks to GPU enables higher system specific performance
  - Data types must fit
  - CUDA eases algorithm porting
  - K1: 192 CUDA cores
- Examples for moving Tasks to GPU
  - Speech recognition
  - Driver assistance
  - Image and video processing
  - Basic algorithms
    - FFT
    - Linear algebra, etc.



Future System Integration Challenges

Future System Integration Challenges

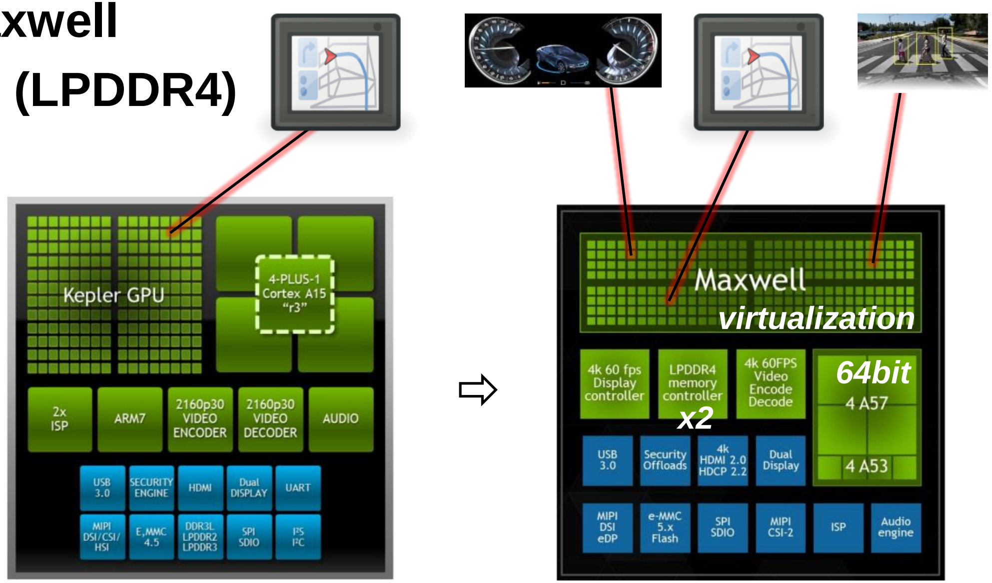
- **SoC Performance drives Automotive Component Integration**
  - **Partial or full integration of instrument cluster, infotainment and driver assistance**



mycable

Future System Integration Challenges

- **K1 ⇨ X1 Performance Scalability paves the way to further System Integration**
  - CPU 32bit ⇨ 64bit
  - GPU Kepler ⇨ Maxwell
  - Memory Bandwidth ⇨ x 2 (LPDDR4)
  - Video ⇨ 4k
- **Further Properties**
  - CPU virtualization
  - GPU virtualization



Thank you for your Attention