



Multicore Power Architecture for time-critical automotive applications

Rainer Makowitz

May 27, 2008



Overview

- » Multicore Trend
- » Real Time applications
- » Automotive Software for Multicore

Automotive Market Segments & MCU Architecture

Powertrain & Hybrid

- Engine Management
- Transmission
- Calibration
- Signal acquisition Processing sensor input (knock, cylinder pressure)
- Symmetric Multiprocessing
- Intelligent timer concepts

Body Electronics

- Central Body Controller
- Gateway
- Dashboard
- Autosar software
- Low Power
- Networking
- Asymmetric Multiprocessing

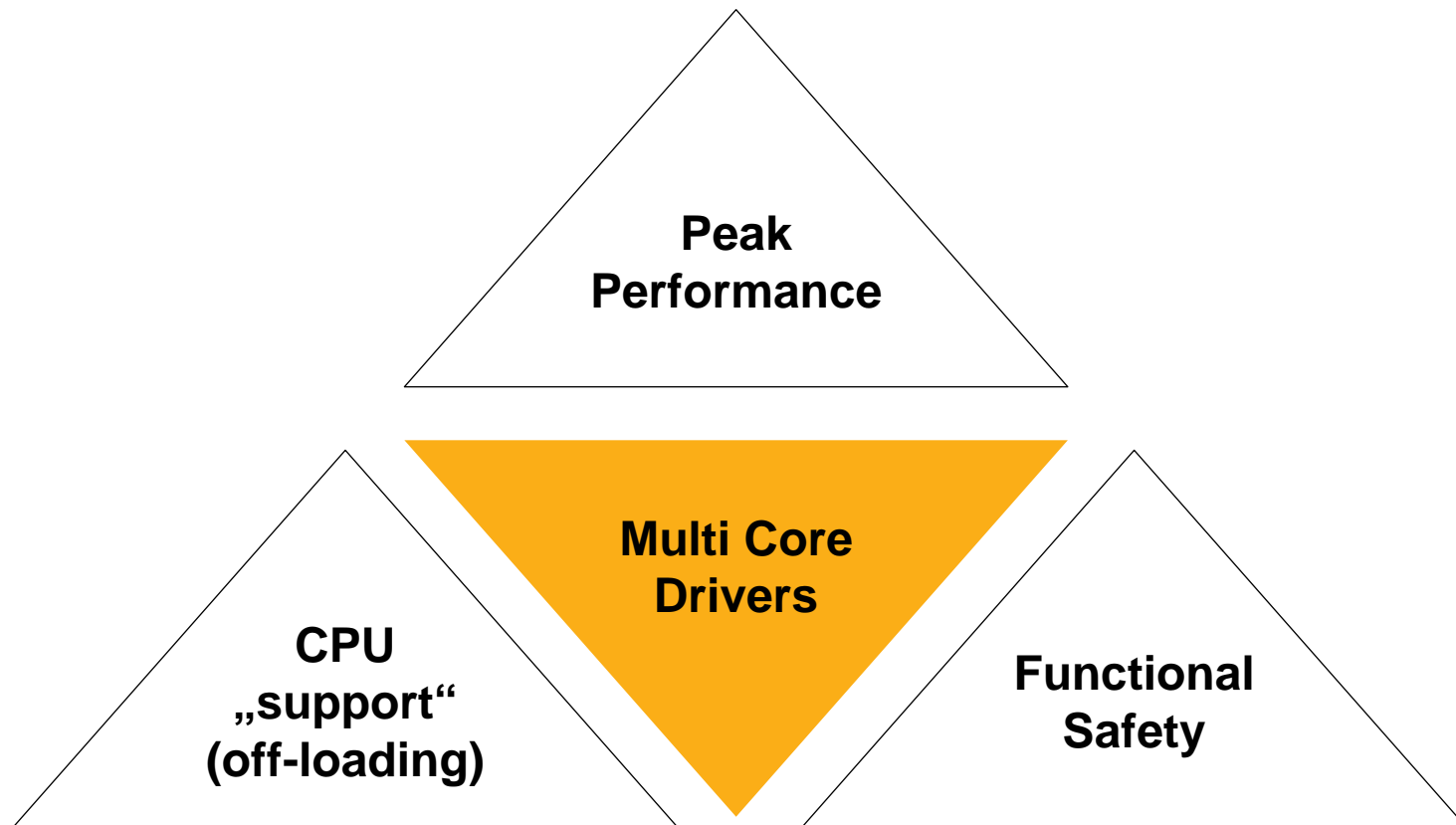
Chassis & Safety

- Airbag
- Steering
- Braking/VSC
- Suspension
- Integrated chassis management
- Functional Safety
- Redundant Cores

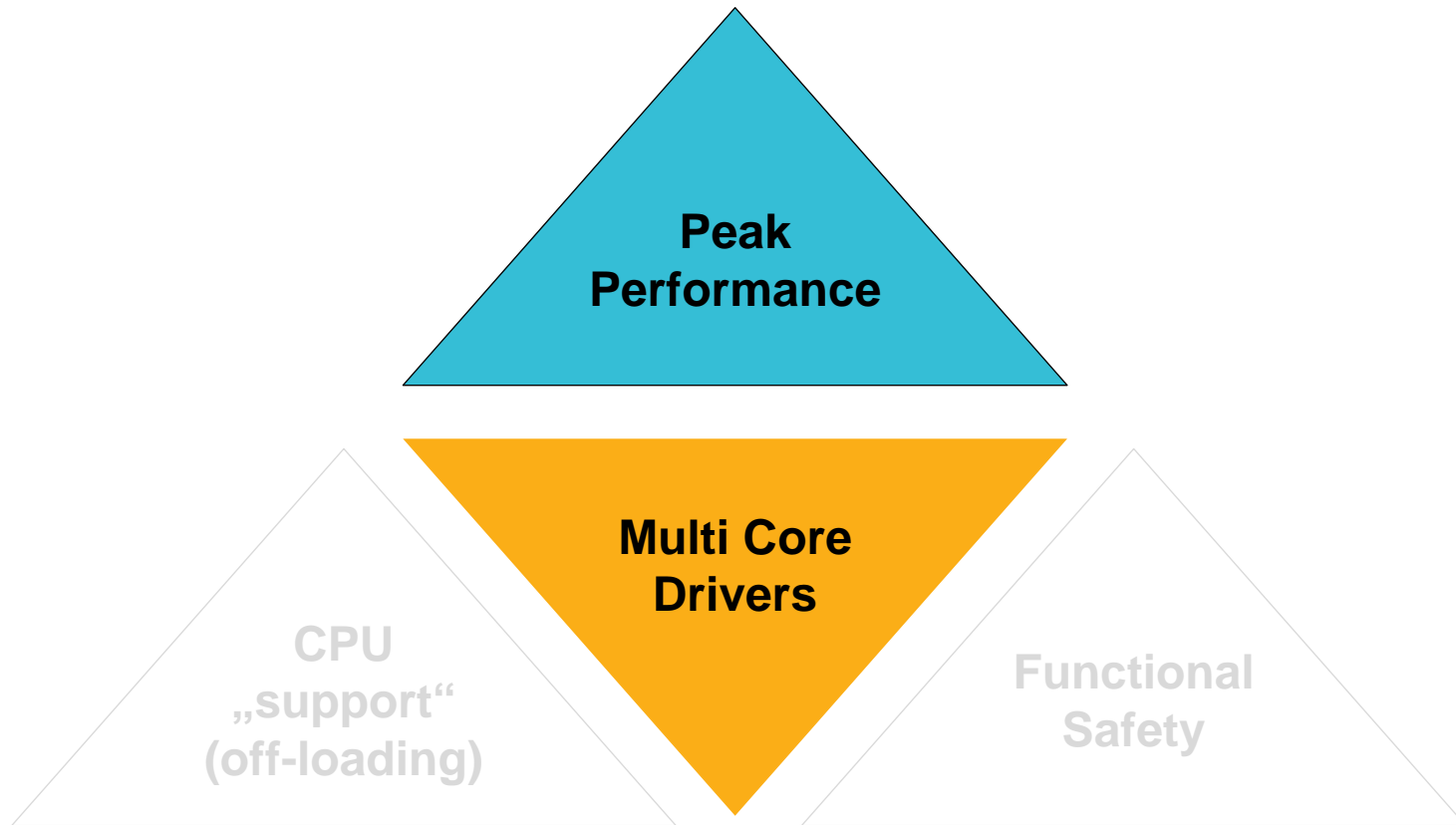
Driver Assistance & Infotainment

- Vision
- Radar
- Infotainment
- 2-3D Graphics
- Symmetric Multiprocessing
- Accelerator Engines

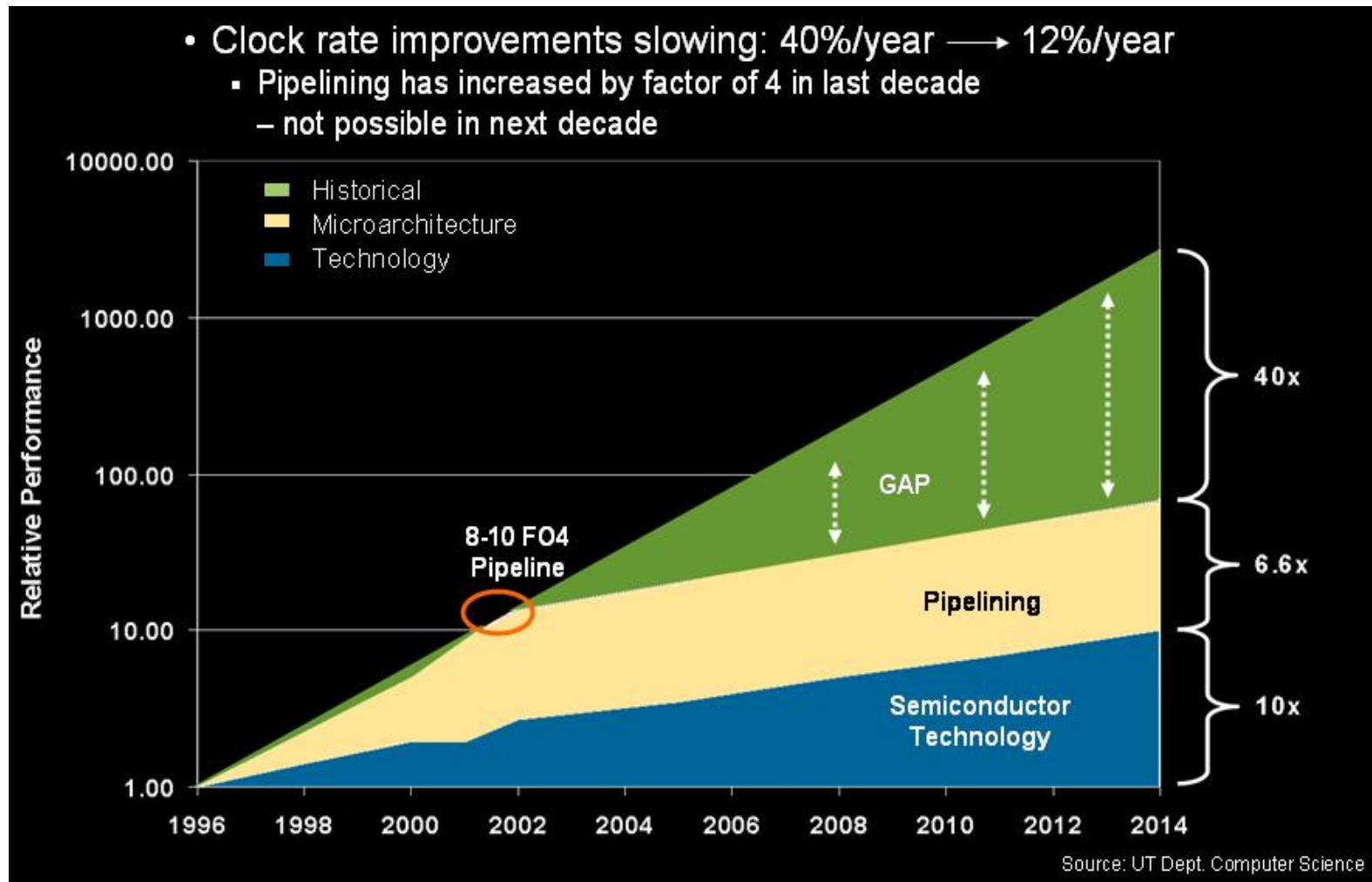
Three Drivers of Multi Core Architectures



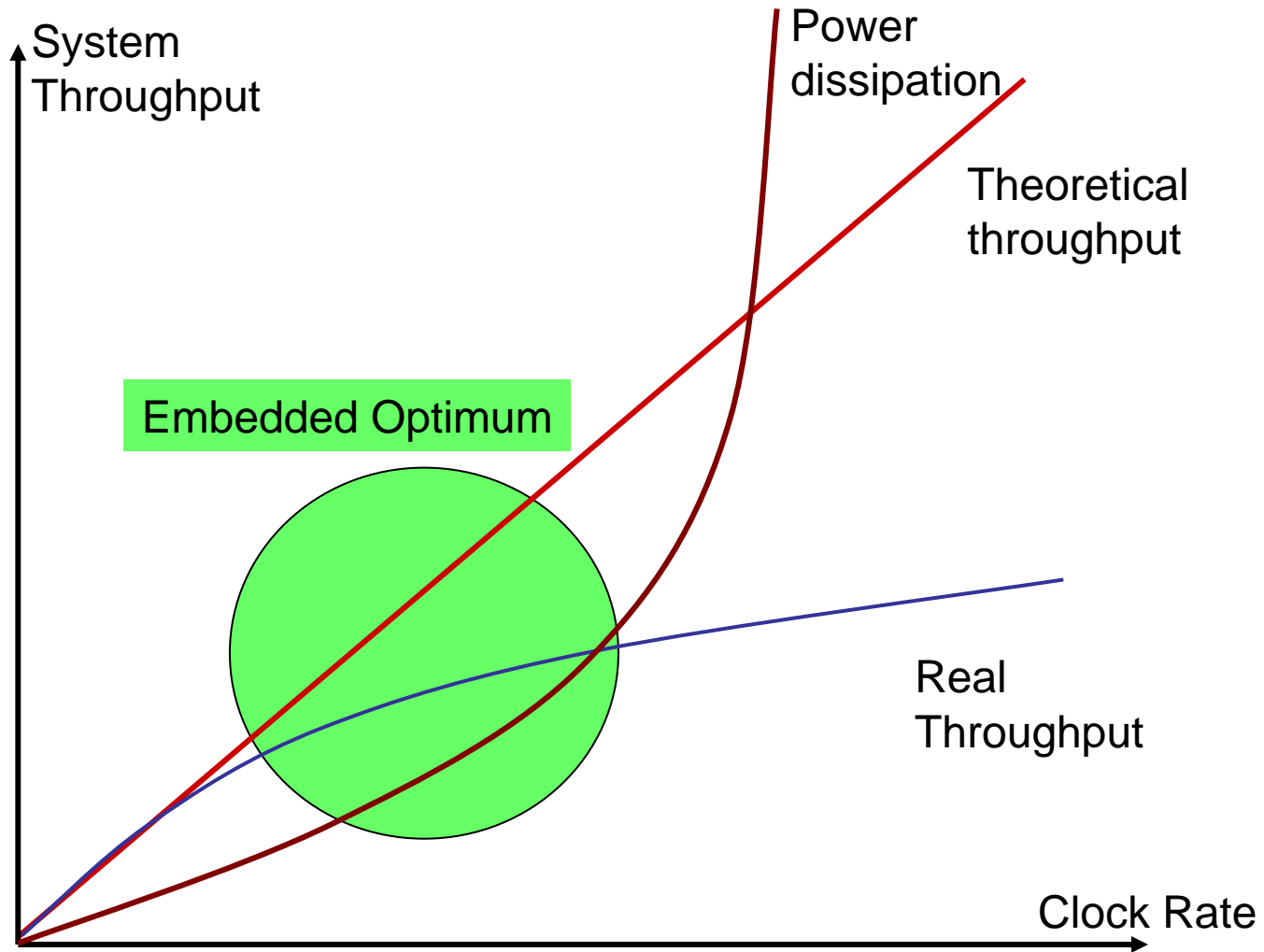
Peak Performance



Performance Scaling & Technology Challenges



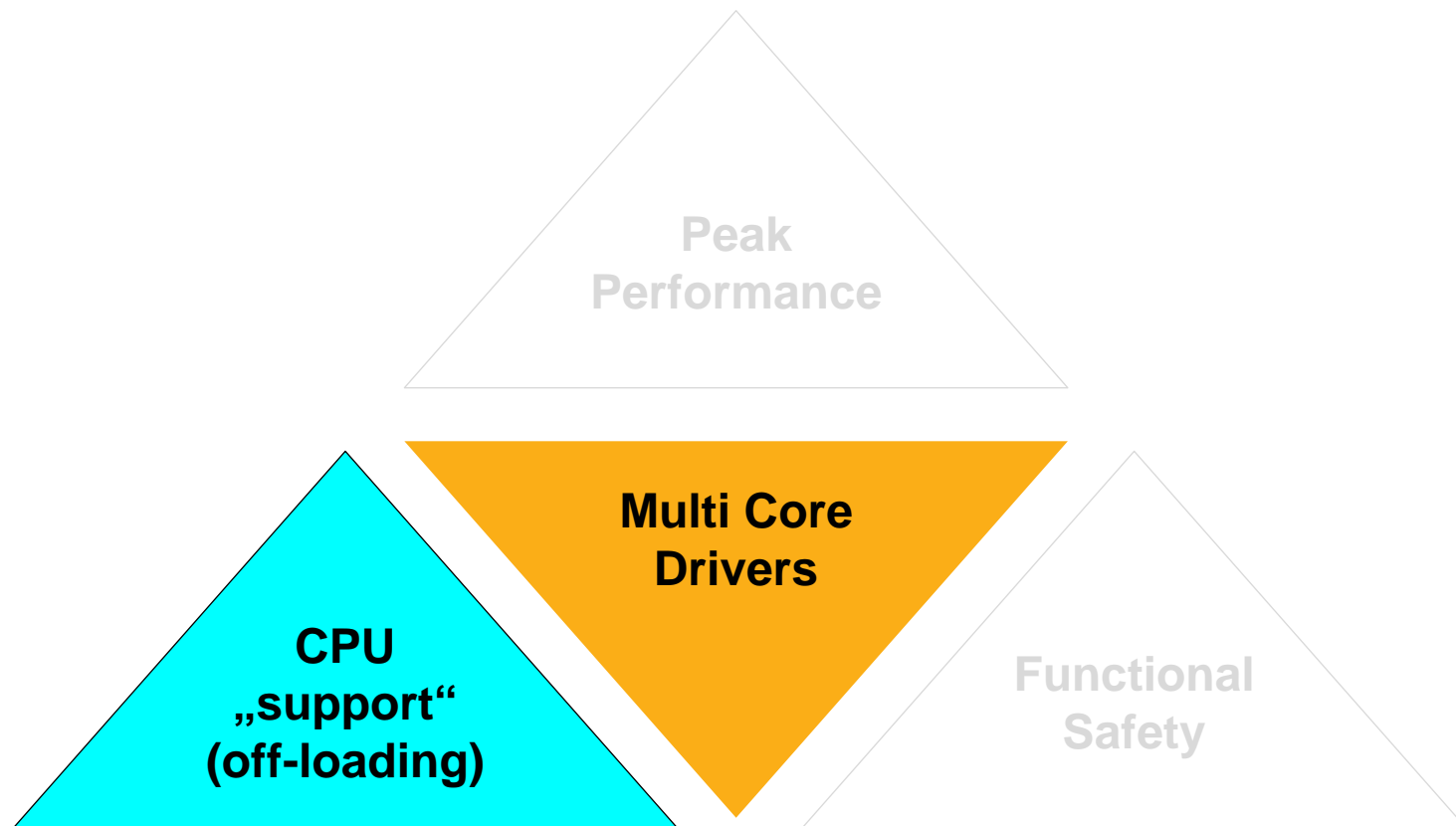
Compute Power: What we want & What we get



Rationale for Multi Core Processors

- » Key MCU performance enablers
 - Clock speed
 - Technology improvement
 - Super pipeline architecture
 - Architectural choice
 - Superscalar architecture (multi issue of instructions, out order execution)
 - Multi threading
 - Multiprocessor
- » Priorities for Automotive
 1. Multiprocessor
 2. Superscalar architecture: expensive core
 3. Multi threading: limited advantage, more difficult to program
 4. Super pipeline architecture: pipeline loss high w/ automotive code

Diversity of compute tasks



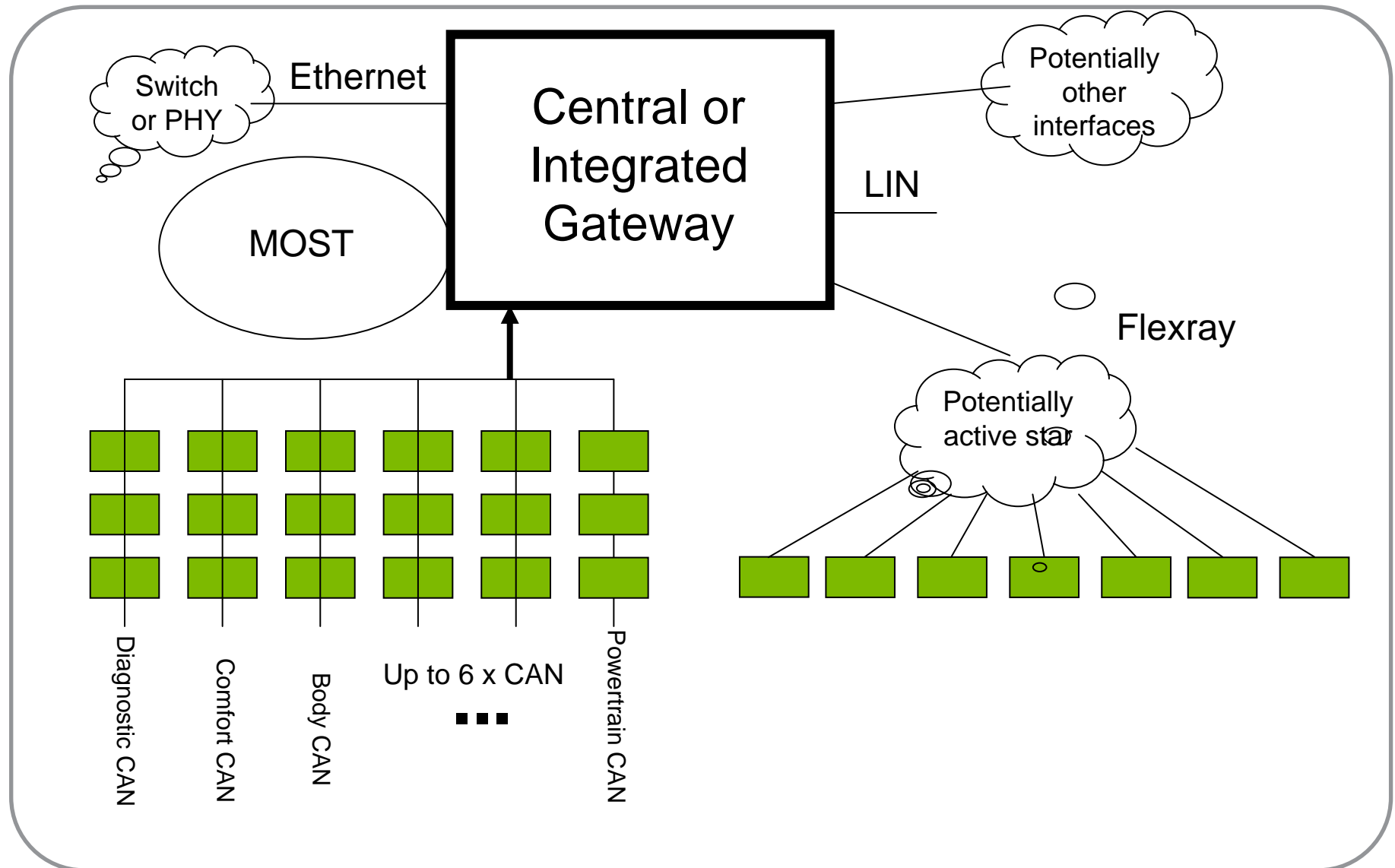
Real Time applications





Example: Automotive Gateways

Multiplex Architecture for modern cars



Gateway MCU Requirements

- » Separation and encapsulation of different networks; mixture of
 - event triggered,
 - time based,
 - synchronous and
 - asynchronous interfaces
- » Handle multiple different protocol types (LIN, CAN, FlexRay, Ethernet, MOST...)
- » Requirement of high performance bandwidth throughput and low latency routing delay
- » Gateways are deeply embedded
 - Invisible value for a car buyer
 - Cost must be minimized

Gateway Implementation Architectural Alternatives

» Yesterday: Datagram Forwarding Gateway

- Simple approach, datagrams are forwarded accross different communication buses – typical legacy CAN-CAN Gateway approach
- For connection of different communication buses part of the complexity is pushed into the connected ECUs (e.g. MOST devices might need to understand CAN Transport Protocol)

» Today: Protocol Gateway

- Terminates and generates the protocols for each communication bus
- ECUs connected to one bus type do not need any knowledge about the other buses (e.g. a MOST node does not need to understand, that the speed signal came originally from the CAN bus and does not need any knowledge about CAN)
- Allows Unit conversion and Signal reassembly
- Mediation between high-speed and low speed networking (natural flow control generation)
- All complexity of conversion is only located in the Gateway

Gateway Routing Core Implementation Alternatives

Static Approach, Gateway Functionality hard coded...

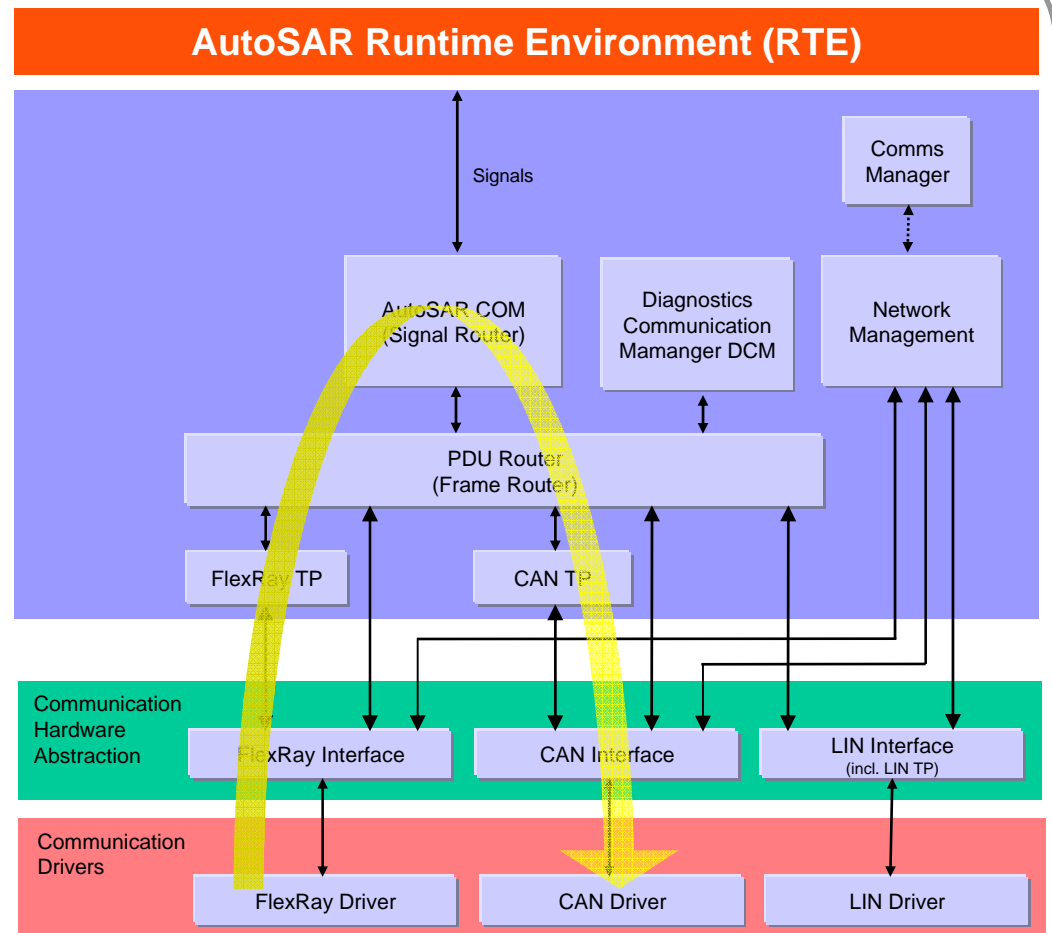
- ✓ Hand crafted resource optimizations
- ✗ Risk of side effects in the system through manually coding and changing of the Gateway translation rules
- ✗ High effort for porting the solution to different CPU types
- ✗ Difficult to estimate resource usage and run time budget without actual implementation
- ✗ Inflexible approach, not suitable for modern OEM scenarios where all system field bus topology, signal and message structures is kept in a central database where the rule set for the Gateway is exported in a machine readable file

Interpreter based Solution, table driven control of GW

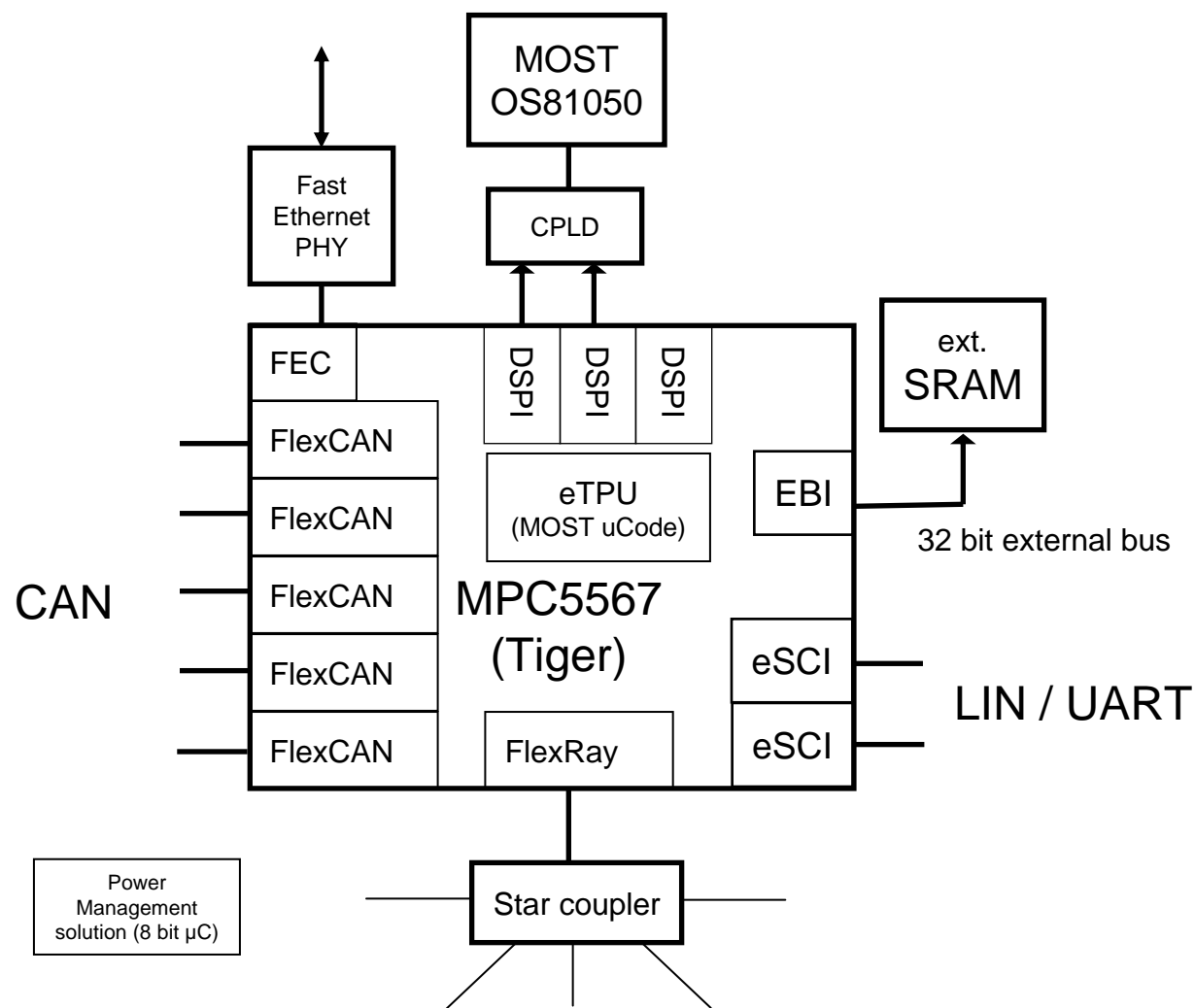
- ✓ No side effects through changes of GW rule set (GW table)
- ✓ Post build process approach without need of re-compilation of the firmware
- ✓ Pre-calculation of resource consumption (RAM/ROM) and estimation of execution time budget based on given GW tables
- ✓ Flexible Approach, process oriented, fully supported by a Tool chain: System Topology, Message and Signal layout inside one database, fully automated GW rule set derivation, Conversion of the rule set and execution in the ECU/car

AutoSAR Gateway Software Architecture

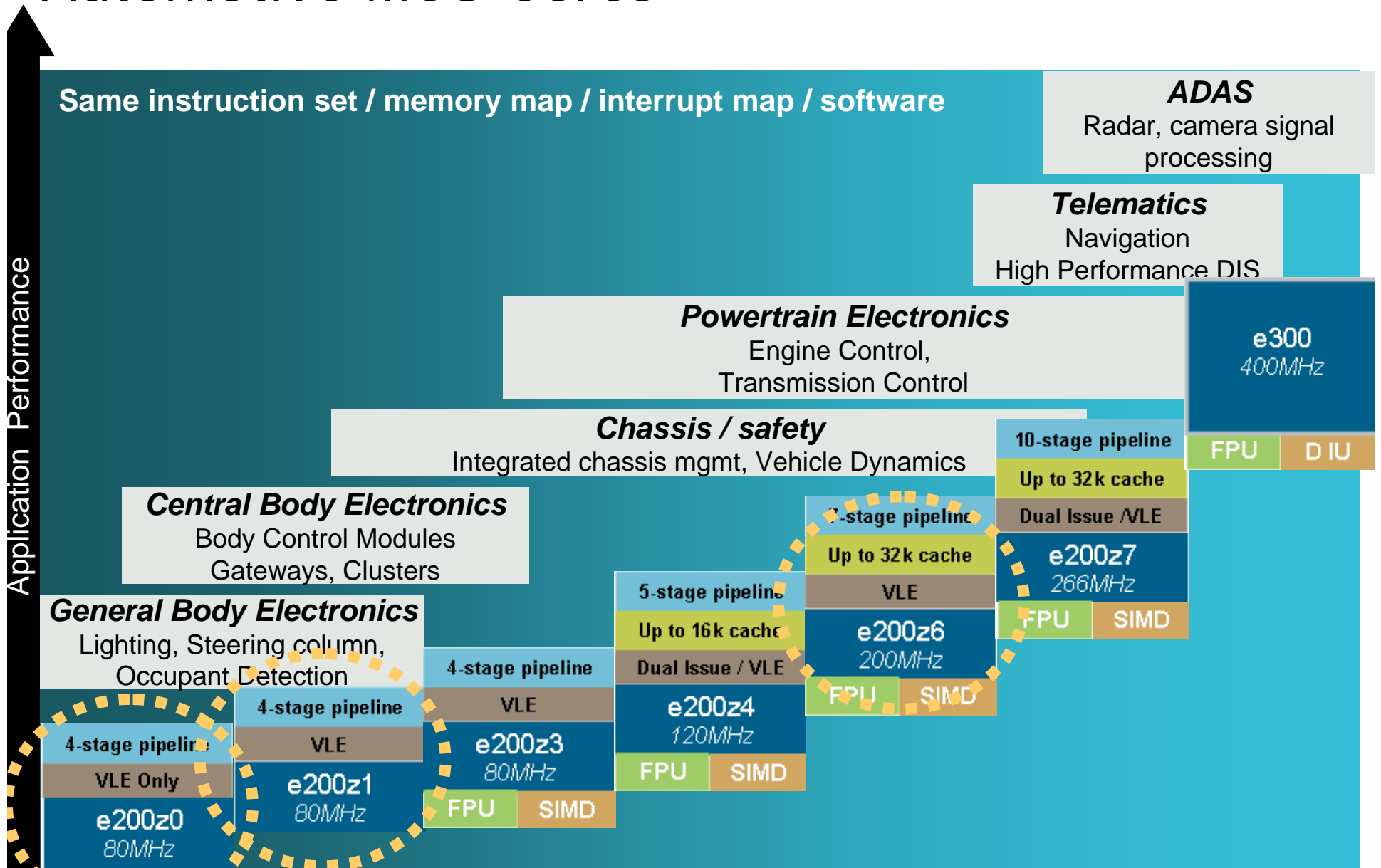
- » Automotive gateway contains several software modules
- » Communication stacks can have relatively large code size
- » Static table based configuration: Size heavily depends on configuration
- » CPU loading can be high, particularly with signal based messaging
- » Maximum latency through gateway needs to be guaranteed: Typically 1 ms



Ethernet-based Gateway Production Solution

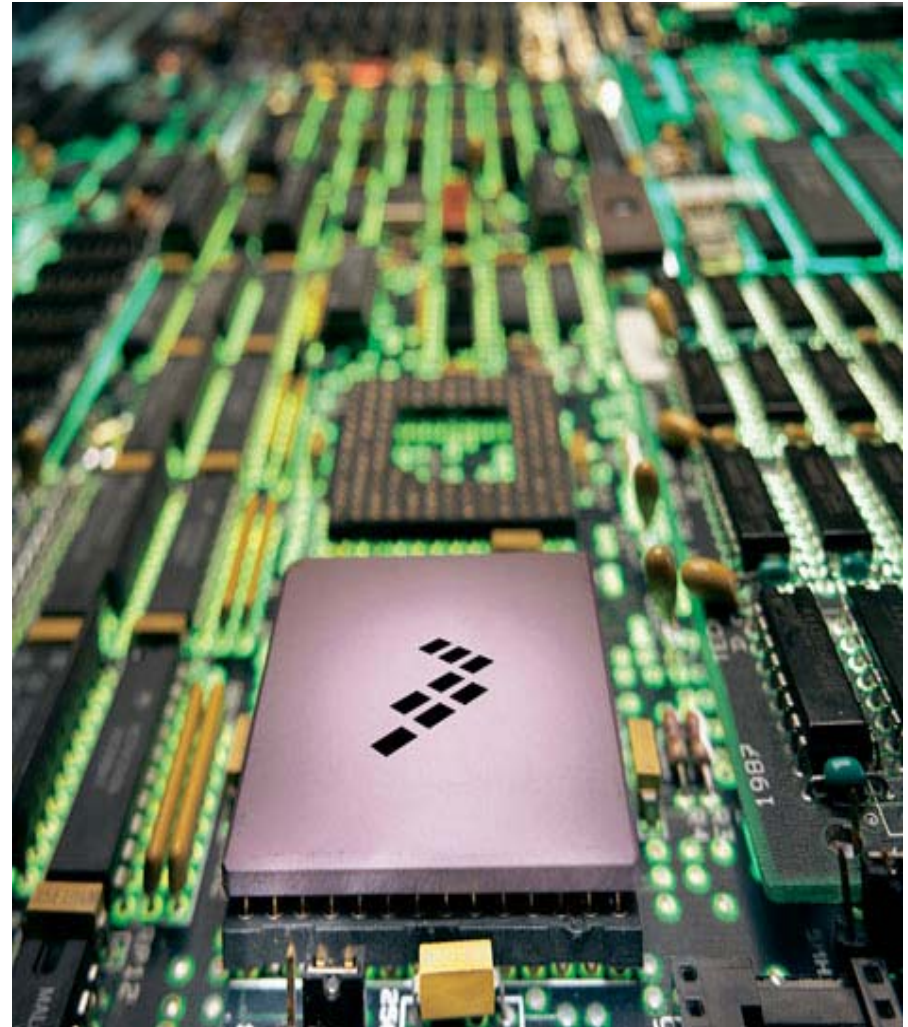


Automotive MCU Cores

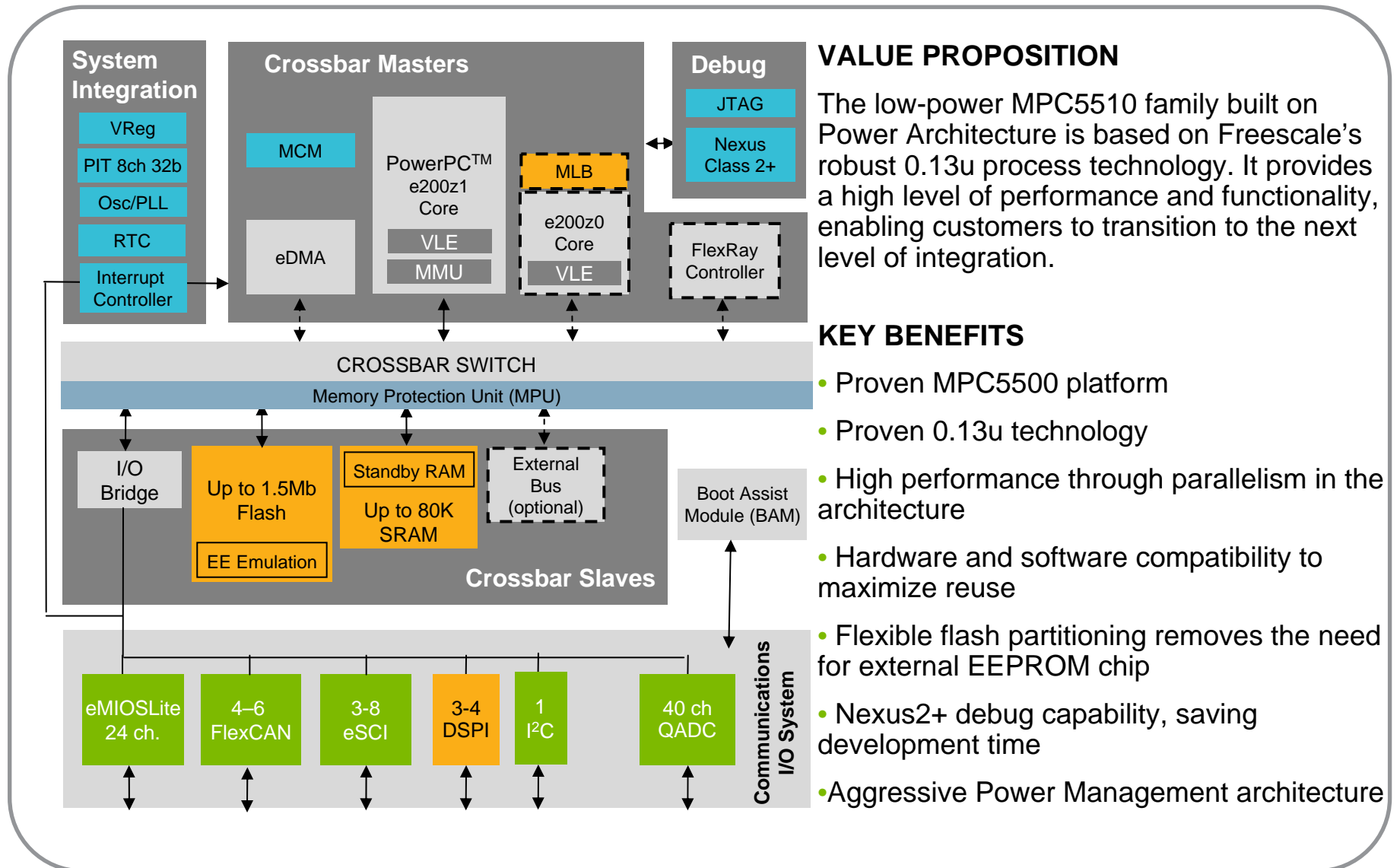


e200z0 I/O Processor

- » Accelerate networking
 - CAN gateways
 - LIN protocol implementation
 - Emulation of proprietary protocols
- » Build your own peripheral
 - Display controller
 - Intelligent watchdog...
- » Real-time software design made simple
- » Customers' Innovation and Differentiation
- » Common toolflow w/ the main CPU !



MPC551x Family Architecture



VALUE PROPOSITION

The low-power MPC5510 family built on Power Architecture is based on Freescale's robust 0.13u process technology. It provides a high level of performance and functionality, enabling customers to transition to the next level of integration.

KEY BENEFITS

- Proven MPC5500 platform
- Proven 0.13u technology
- High performance through parallelism in the architecture
- Hardware and software compatibility to maximize reuse
- Flexible flash partitioning removes the need for external EEPROM chip
- Nexus2+ debug capability, saving development time
- Aggressive Power Management architecture

Low Power Challenge



- » Historically, MCUs for Body applications such as Gateways use the „STOP“ mode, where clock was stopped to conserve power
- » With shrinking semiconductor technology transistor leakage becomes a growing element and just stopping the clock will not achieve the desired low power results.
 - 180 nm seems to be the last technology, where this works without additional steps
 - MPC5510 is designed in 130 nm and therefore we introduce power gating with this generation
 - Only blocks required are actually powered – the other blocks can be shut off
- » For blocks powered Intelligent clocking is applied
 - Only clock what is needed
 - Only clock at the minimum speed
 - On-chip clocking generation
- » Additionally Active Well biasing is used to further reduce power consumption
- » Intelligent Autonomous operation of peripherals (with the help of eDMA or coprocessor) further reduce the need of CPU activity

Performance Enhancements for Gateway (1)

- » High performance PowerPC e200z core with pipeline architecture optimized for memory subsystem (i.e. e200z6 core with 7 stage pipeline and cache and e200z1 core with 4 stage pipeline and no cache)
- » FEC (Fast Ethernet Controller)
 - Integrated FEC (100 Mbit Ethernet)
 - DMA transfer of Ethernet data into memory
 - No bottleneck of accessing Ethernet through external narrow bus
 - No need for Ethernet driver to copy data (zero copy stack)
 - Industry standard (over 70 M units shipped) PowerQUICC Ethernet Controller programming model
- » FlexRay
 - Support of 128 message buffers
 - DMA transfer of message content into memory
 - No performance bottleneck of copying message content into memory by CPU
- » FlexCAN
 - Flexcan V3 enhancements: Additional TX/RX echo filter, individual filters per message buffer and support of RX FIFOs
 - Flexcan V4 enhancements: Hardware RX FIFO mode, TX transmit queue with local priority field and transmit abort mechanism

Performance Enhancements for Gateway (2)

- » TCP/IP Protocol Stack CPU performance requirements
 - FEC programming model allows minimal possible CPU loading compared to external Ethernet MACs (where CPU needs to copy data)
 - Industry rule of thumb ~ 1 MHz Pentium III class CPU per Megabit TCP/IP throughput (depending on socket size, MTU, ...)
 - Production ECU measurements with MPC5567 are roughly in this ballpark
- » MediaLB traffic can create significant CPU performance requirements
 - E.g. MLB SRAM Macro does not support DMA operation and multi (double) buffering without CPU intervention, CPU needs to copy buffers into / from application space
- » „Soft“ MediaLB solution
 - Solution requires some specific peripherals from the MPC55xx / MPC56xx family (2 x DSPI and eDMA) plus a coprocessor (eTPU or e200z0 core).
 - The coprocessor must be dedicated to the MLB emulation (in other words can not be used for other things additionally)
 - Clever system approach significantly reduces CPU loading
 - Extreme example of sync traffic routing reduces that to 0 compared to significant loading in other cases
- » MOST vs Ethernet
 - Experience with current development for automotive Gateways shows a factor of ~5 for CPU performance requirements for MOST (MHP) versus TCP/IP
 - Example implementation delivers 8-10 Mbps transport from Ethernet to MOST, where TCP/IP consumes ~10-11 % CPU load and MOST NetServices consume ~ 50 % CPU load
 - Potential solution for higher performance networks (e.g. MOST150) is TCP/IP transport over MOST framing

Freescale Gateway Microcontrollers

MPC5567 (Tiger)

available now (130 nm)

5 x CAN, 2 ch FlexRay
e200z6 132 MHz with 8 kB cache
MLB 256/512 fs (with ext. CPLD)
2 x LIN/UART
3 x DSPI (2 are used for MLB)
2 MB Flash / 80 kB SRAM
Fast Ethernet

MPC5516G rev. 0

available now (130 nm)

6 x CAN, 2 ch. FlexRay
e200z1 + e200z0 (66-80 MHz)
MLB 256 fs (with ext. CPLD)
8 x LIN/UART
3 x DSPI (2 are used for MLB)
1 MB Flash / 64 kB SRAM

MPC5517G rev. A

(130 nm)

6 x CAN, 2 ch FlexRay
e200z1 + e200z0 (66-80 MHz)
MLB 256 fs (integrated)
8 x LIN/UART
4 x DSPI (2 are used for MLB)
1.5 MB Flash / 80 kB SRAM

MPC5514G

512 kB Flash

MPC5604C Bolero

(90 nm)

512 kB Flash
6 x CAN
e200z0 (64 MHz)
4 x LIN/UART
3 x DSPI

MPC5606C Bolero

MPC561x

In design (90 nm)

6 x CAN, 2 ch. FlexRay
e200z6 + e200z0
MLB integrated
8 x LIN/UART
4 x DSPI
Fast Ethernet

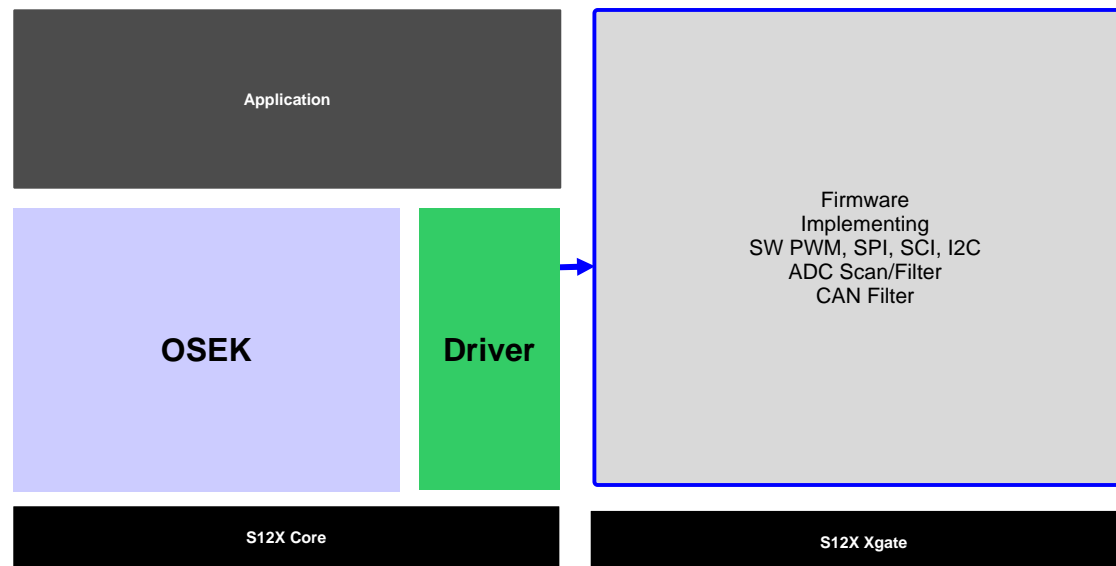
Automotive Software for Multicore



AUT@SAR

Before Autosar - Early Success Stories: S12X

- » Low overhead by using the coprocessor like a peripheral
 - Xgate firmware loaded into RAM at startup
 - Interrupts routed to Xgate for optimal latency
 - Efficient, simple peripherals made smart



Co-Processors and Autosar Today

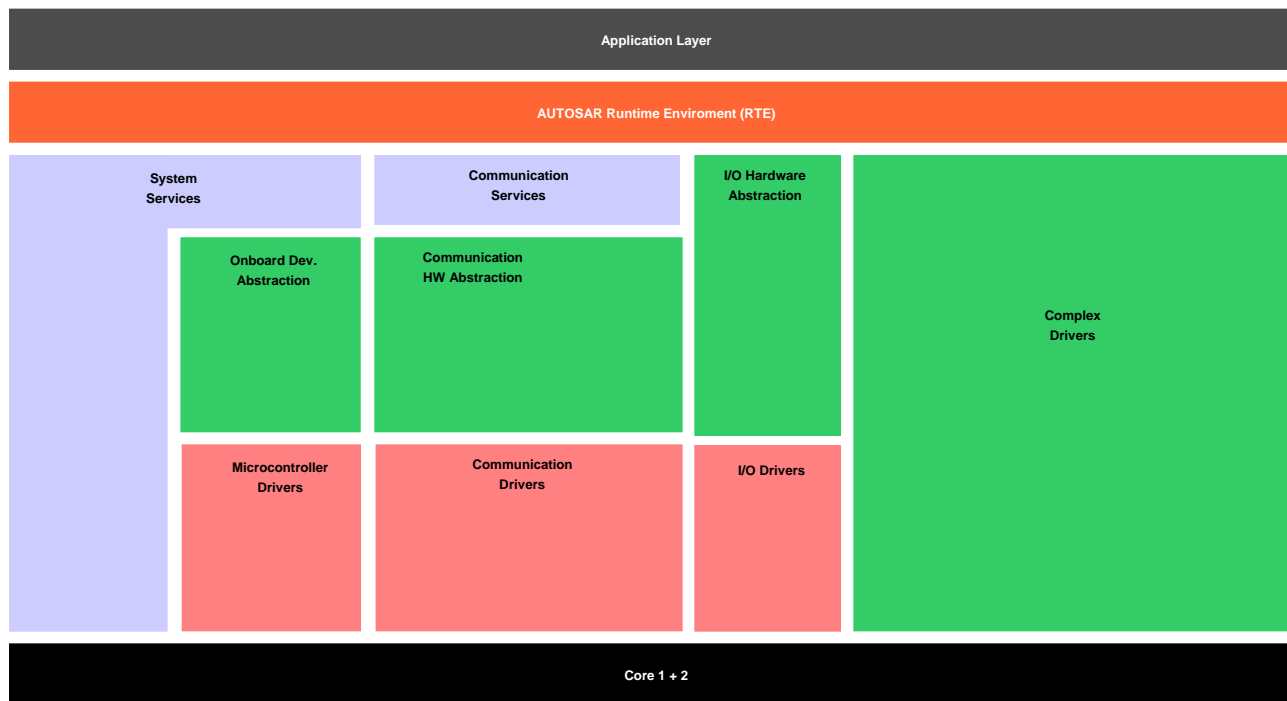
- » Autosar specification releases 1, 2 & 3 do not define multiprocessing
- » Co-processor usage
 - IOP still has 'private' firmware loaded
 - Communication to the IOP firmware through the standard interface of an AUTOSAR complex driver



Multi-Core Support in Autosar - Future

» Autosar 4.0

- Multi-core support in definition
- Virtual Function Bus to connect RTEs on separate cores
- Single image vs. multiple images



Summary

- » Multi core architectures are a significant trend in Automotive
 - For getting more MIPS/Watt and more MIPS/mm2
 - For offloading time-critical tasks from main CPU
 - Gateway is the application that benefits enormously from multiple cores
 - Multimedia network integration is most demanding for system throughput
- » Structured software development faces a few challenges when using dual core processors as a target
 - Tasks of low complexity are running happily today on co-processors
 - Autosar will provide the mainstream route for the majority of automotive dual core applications

Power.ORG ™

Munich