Implications of microcontroller software and tooling on safety-critical automotive systems

Kai Konrad (Dipl. Ing. (FH), MBA)
Infineon Technologies AG
April 2008

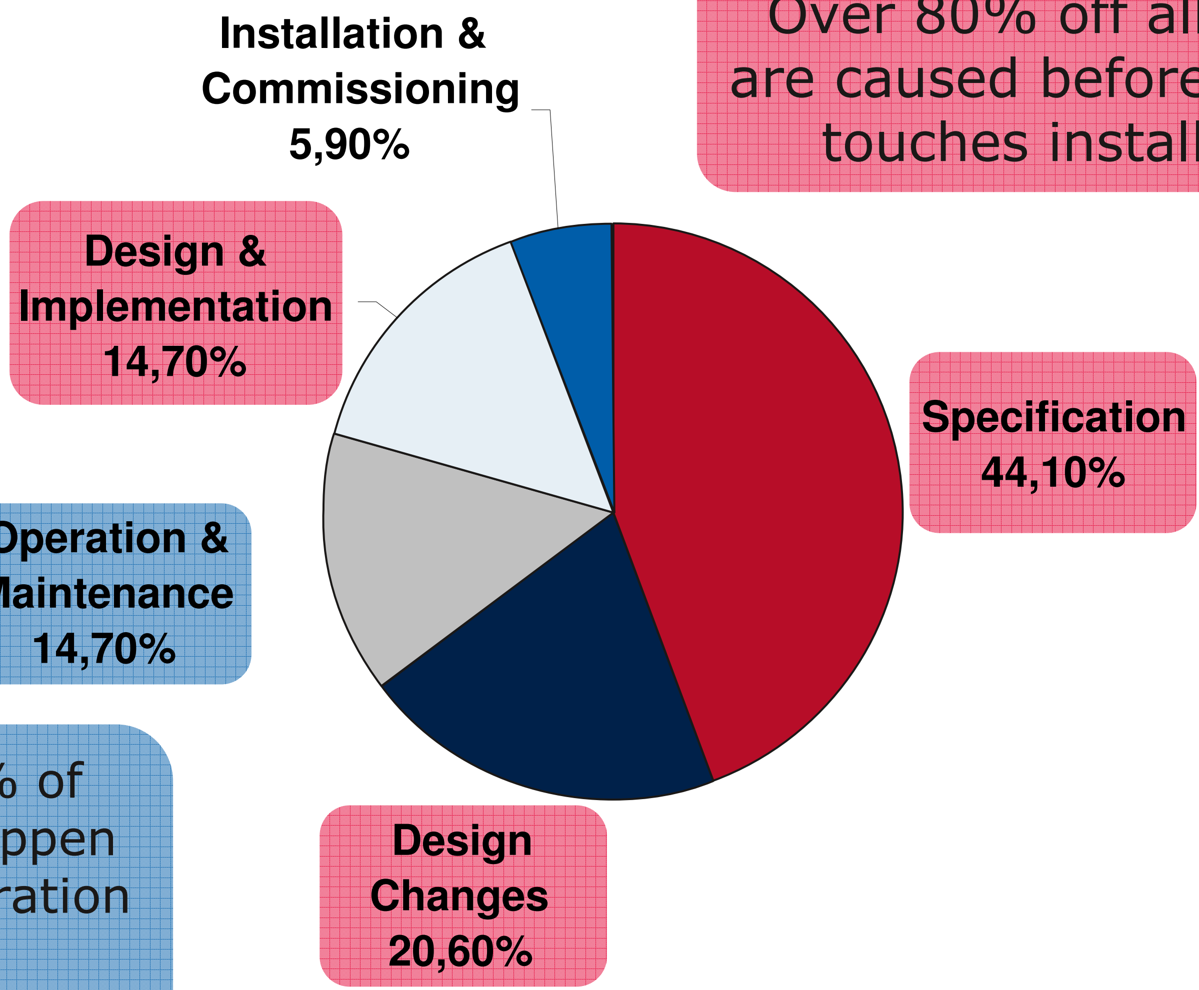
Kai.Konrad@Infineon.com

Never stop thinking

Table of contents

Primary causes of system failure - Industry as example

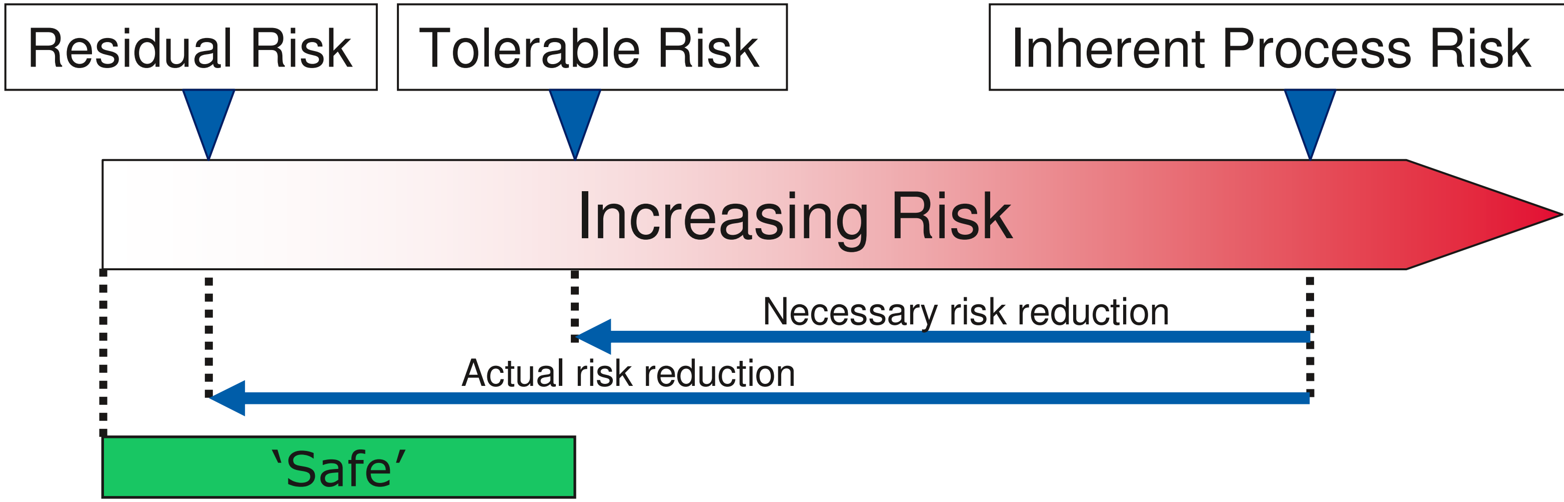
Over 80% off all failures are caused before any user touches installations



Only 15% of failures happen during operation and maintenance

Source: HSE UK report 1999, based on industrial accidents based on 34 incidents

Safety for automotive systems: Inherent risk minimization as major task

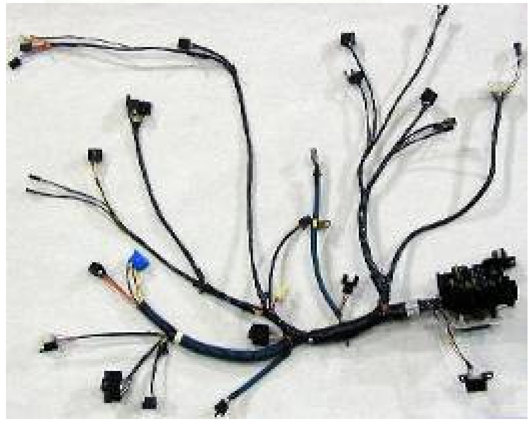


- **A system is 'Safe' if there is a 'Tolerable Risk' when it is in operation**

- **Functional Safety**
  "Part of the overall safety which depends on the correct functioning of safety-related systems for risk reduction. Functional safety is achieved, when every safety function is performed as specified"

Automotive system safety includes:

- Sensors
- Processing
  - Hardware
  - Software
- Actuators
- Interconnections
- Energy Supply



Process standardization as key to define functional safety future

- Industrial Standard for functional safety **IEC61508** emerging as a guideline for current automotive systems
  - Defining Safety Integrity Levels (SIL) 1 to 4
  - Interesting for Automotive are levels up to SIL3

- New Automotive Standard ISO26262 currently under preparation
  - Will be designed for automotive systems needs
  - Defining Automotive Safety Integrity Levels ASIL A to D

- Motivation for standardization
  - Common process over automotive industry
  - Legal protection
  - State-Of-The-Art definition
  - Higher system quality

IEC61508 - Sil3 (Safety Integrity Level 3)

- SIL3 as SYSTEM safety accreditation standard
  Means:
  - Probability of dangerous failure < 10^{-7} per hour of operation
    (100 FIT = 100 Failure In Time)
  - Possible failure modes leading to a dangerous system state of < 1% of total system operational envelope
    (99% Safe Failure Fraction = SFF)

  Requires:
  - Detailed system and component fault analysis (FMEA)
  - High component quality
  - Strict design and integration methodology, documentation, application support
- Example: A person who lives to 80 years old has 701280 hours of life before their 'dangerous failure'

  Equivalent to 0.7ppm → SIL 2

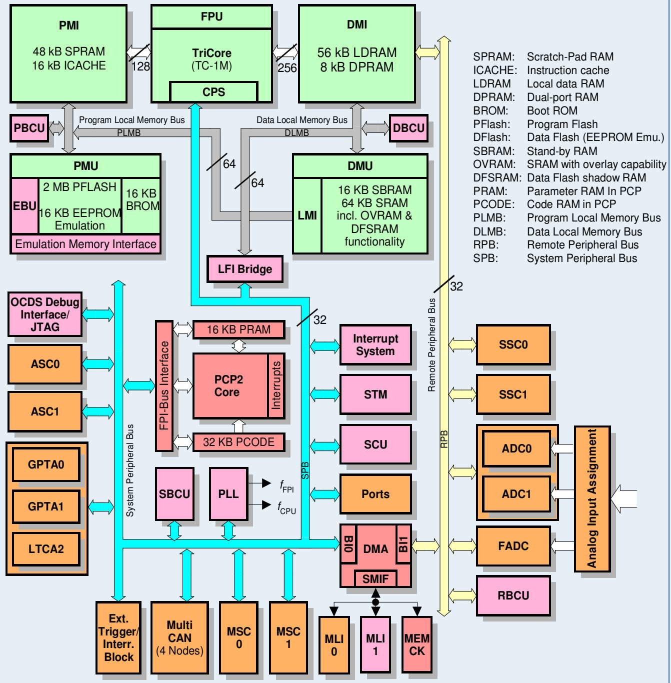
What does IEC 61508 SIL 3 mean when applied to a microcontroller?

- **Microcontroller + Safety-Driver + Application Functional Safety Software** has to meet:

Safe Failure Fraction (SFF) > 99%

(Covers > 99% of used silicon!!!)

State of the Art Microcontroller



Infineon TriCore® TC1796

Probability of failure per hour (PFH) <<10-7

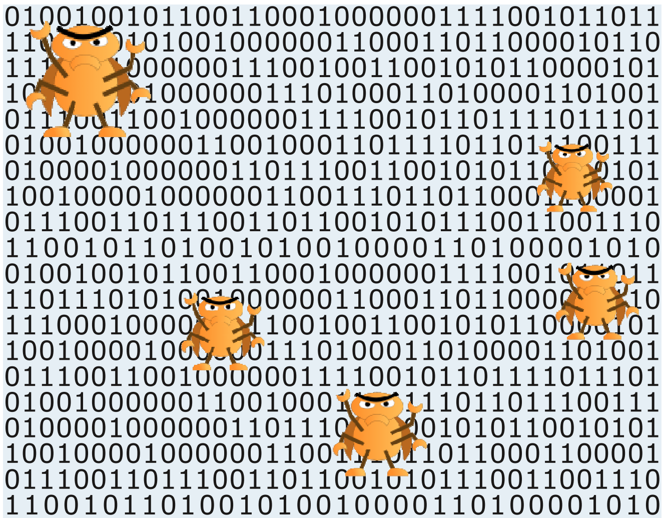
Hardware errors in semiconductor as cause of dangerous systems faults

Error Case	Statistical/Transient Error	Systematic/Static Error
General Behavior	Short Temporal Duration	Permanent Nature
Potential Causes	ESD EMI Radiation …	ESD, EMI Electrical / Mechanical Overstress Specification Errors Hardware and Software Bugs (Common Mode Errors) …
Measurement	FIT Rate Determination (e.g. Experimental)	PPM Rate Estimation (e.g. Field Experience)

Table of contents

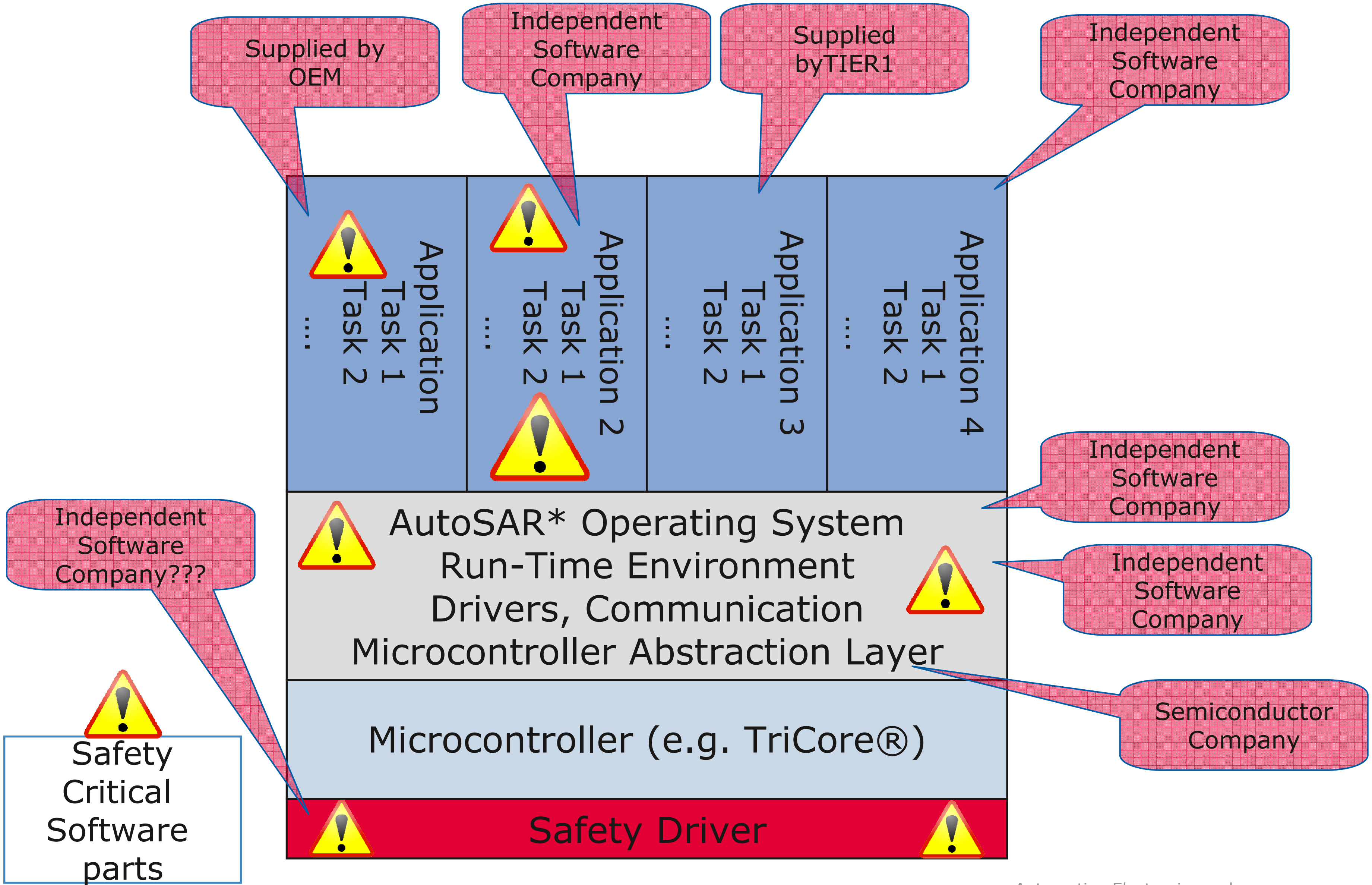
Software development as critical issue for system safety

- Is it possible to write software without bug’s???
- After initial coding you can expect one bug per 20 lines of code
- After thorough unit testing you can expect 1 bug per 1000 lines of code in the final release
  - 1 line ~5 bytes, so 1 bug per ~5KB

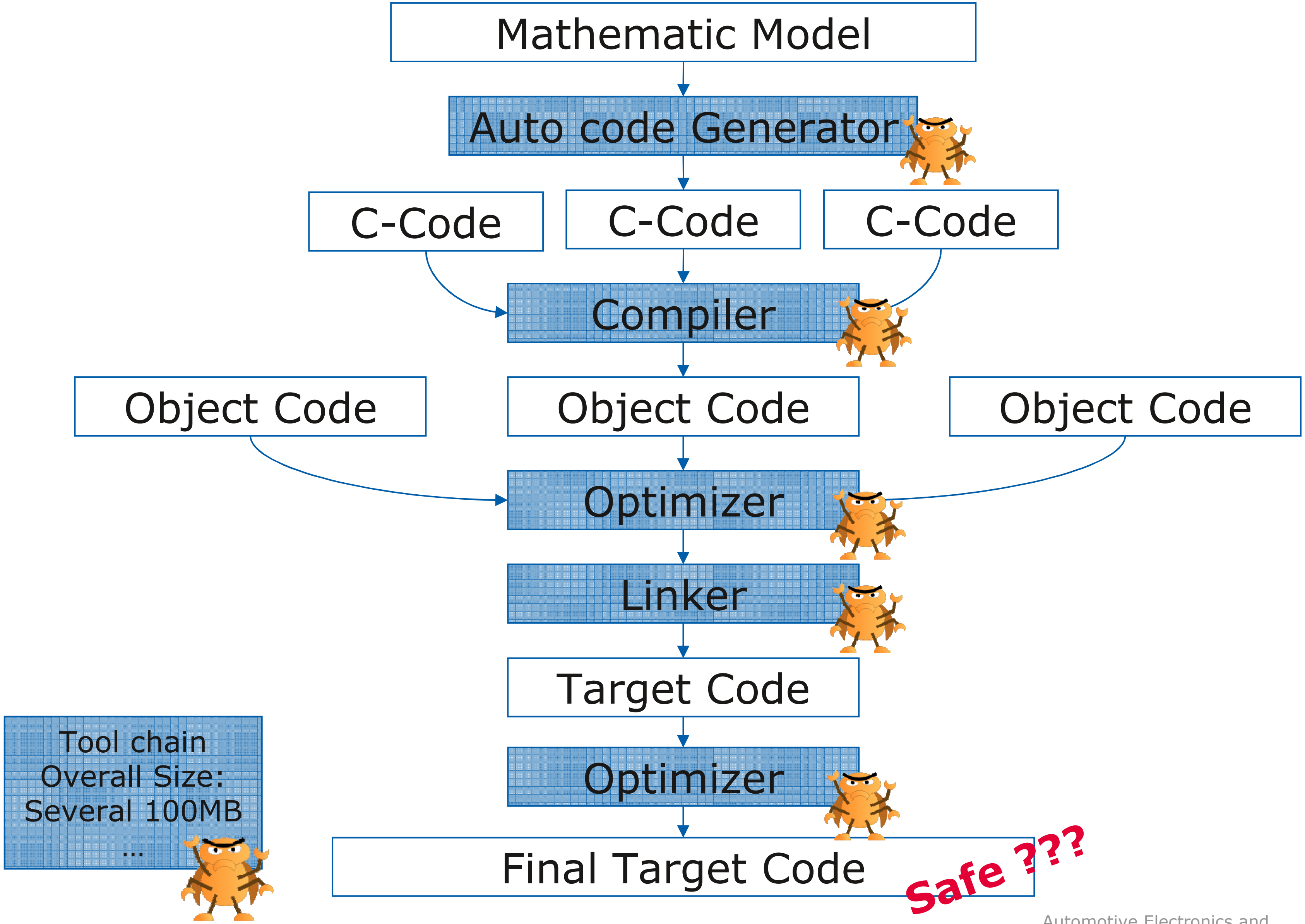


Application	Microcontroller Type	Code Size	Statistics
Steering Angle Sensor	8 Bit	32KB	7 Bugs
Low-end Sensor Cluster	16 Bit	128KB	26 Bugs
Airbag Controller	16/32 Bit	256KB	52 Bugs
EPS Controller	16/32 Bit	512KB	104 Bugs
Central Chassis Controller	32 Bit	1.5MB	308 Bugs

Today's automotive software partitioning as critical issue



Software compilation flow as critical issue



Additional Safety Driver requirements

- Coverage of transient computation faults
- Fault model for testing data and addresses of registers, caches, internal RAM, Flash, CSFRs
- Test for dynamic cross-over of memory cells or registers
- No, wrong or multiple addressing
- Testing of opcode decoding and execution including flag registers
- Test of watchdog, traps, ECC (Parity), …
- Peripheral configuration and operation
- Testing of program counter and stack pointers
- Detection of Continuous interrupts, Crossover of interrupts, Unused Interrupts
- Task execution monitor for OS and critical tasks
- External ASIC covers common cause failure
  Power supply, short circuit on chip Temperature of chip
  EMC System clock

Application independent requirements for functional safety in microcontrollers

Table of contents

Feasible functional safety approach for microcontrollers:

- Functional safety of a microcontroller as part of the system has to be split into:
  - Microcontroller functional ***dependent*** safety
    - Must be considered in the application itself
    - Key competence of (automotive) ECU supplier
    - Concept support by semiconductor supplier
  - Microcontroller functional ***independent*** safety
    - Must be supported independent from application
    - Key competence of semiconductor supplier
      - Hardware support by microcontrollers
      - Supply pre accredited software and concepts to all customers
      - Supply maintenance and know how
      - Supply scalability over IEC61508

Example: Infineon TriCore® PRO-SIL™ concept

Existing Infineon TriCore® products **can fulfill IEC61508 SIL3** requirements
without an additional safety microcontroller or dual core lockstep technology.

- E.g. TriCore TC1796 or TC1766
- Based on asynchronous / asymmetric dual core architecture

TriCore PRO-SIL™ concept for functional dependent safety

- Concept support for redundant or diverse calculation
- Software encapsulation schemes
- Task and OS execution monitoring

TriCore PRO-SIL™ concept for functional independent safety

- Supply accredited safety concepts
- Supply and maintain State-Of-The-Art safety driver
  - Written after CMMI standard
  - Application, operation system and runtime environment independent
  - Customer independent
  - Scalable common code set over many OEM and ECU suppliers for
    - **greater quality**
    - **Interoperability**
    - **legal protection**

Table of contents

Options for safe software development

Write and certify:
- All used Tools
- All application software

to SIL3 Standards

Safe and Robust Code

Use Proven-In-Use or diverse tools, build redundancies and diversity into application

Requirements for safe computation

Coverage of Transient Errors

- Caused by e.g. Radiation
- PFH for usual microcontroller core system is not reaching SIL3 requirements (<10^-7)

Safe and Robust Code

Coverage of Static Errors

- Caused by soft- and hardware bugs
- Avoid common mode errors from hardware and software bugs

Redundant Calculation
of critical software

Safe and Robust Software Computation

Diverse Calculation
of critical software

Software development and computation proposals

Every effort must be made to negate the need to qualify software and the tooling

- Qualification is expensive, limits configurations, freezes release levels, is difficult or impossible to prove
- *Currently there are full tool chains known to fulfill IEC61508 SIL 3 requirements*

Transient Error Detection	Static Error Detection	Programming Model	Code Generator	Compiler /Linker	Libraries	Data /Structures	Computing Cores (Hardware)	Method Proposal
no	no	Common	Common	Common	Common	Common	One Core	No Failure Consideration
		Common	Common	Common	Common	Redundant	One Core (Double Calculation)	Calculate Same Algorithm Twice For Transient Errors
		Common	Common	Common	Common	Redundant	Redundant (e.g. Lockstep)	Calculate Algorithm Twice For Transient Errors
		Common	Common	Diverse	Diverse	Redundant	Common (Running Diverse Code Set)	Compile Code Twice With Different Optimization Levels For diversity
		Common	Common	Diverse	Diverse	Redundant	Diverse (e.g. TriCore + PCP)	Use Asymmetric Core System With Two Different Tool Chains
		Common	Diverse	Diverse	Diverse	Redundant	Common (Running Diverse Code set)	Add Diverse Code Generation (e.g. Auto + Complex Code)
yes	yes	Diverse	Diverse	Diverse	Diverse	Redundant	Diverse (e.g. TriCore + PCP)	Fully Diverse Development

Robust software partitioning as requirement for functional safety

*AutoSAR - scalability class 4
Memory protection
Time protection

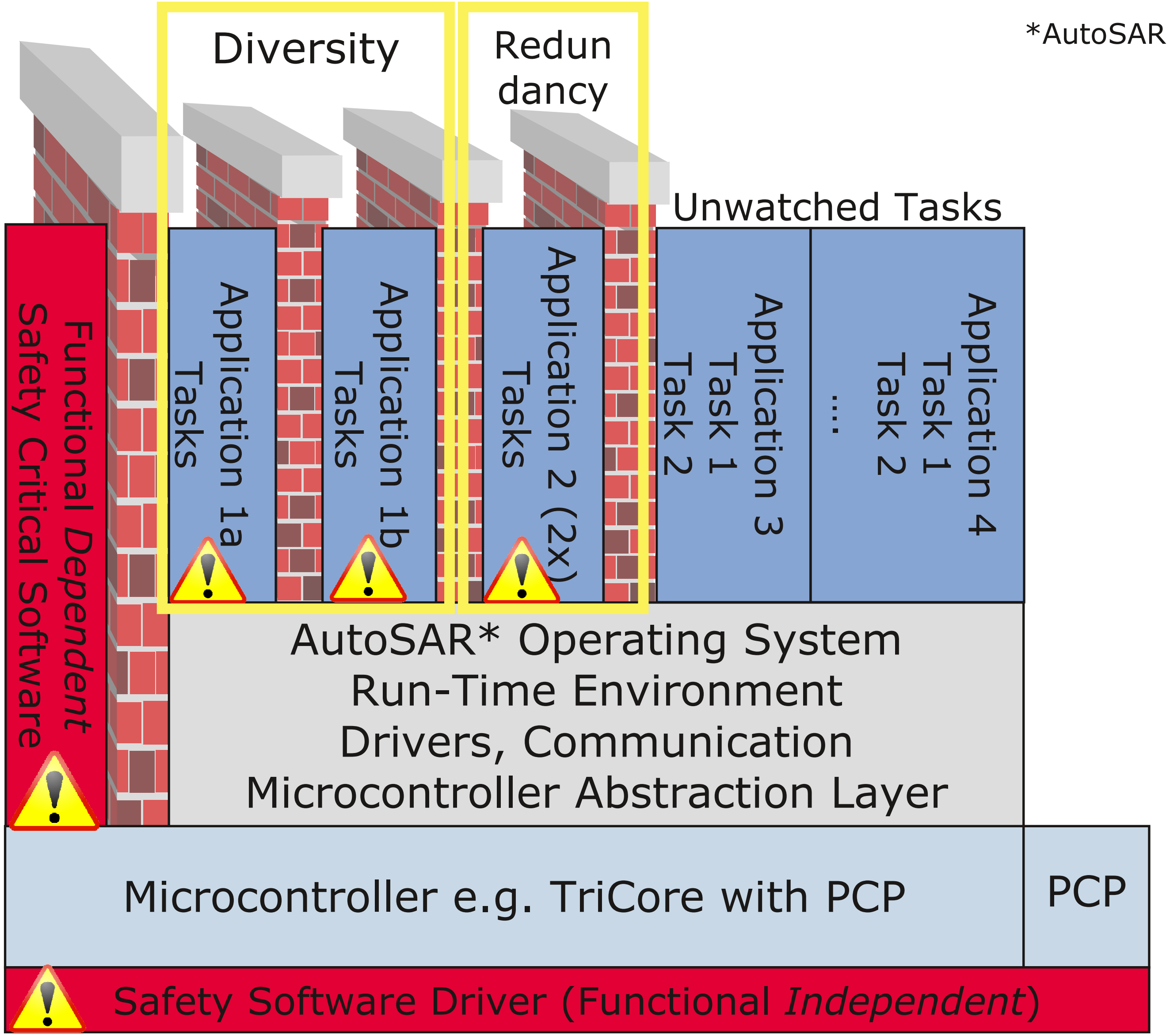
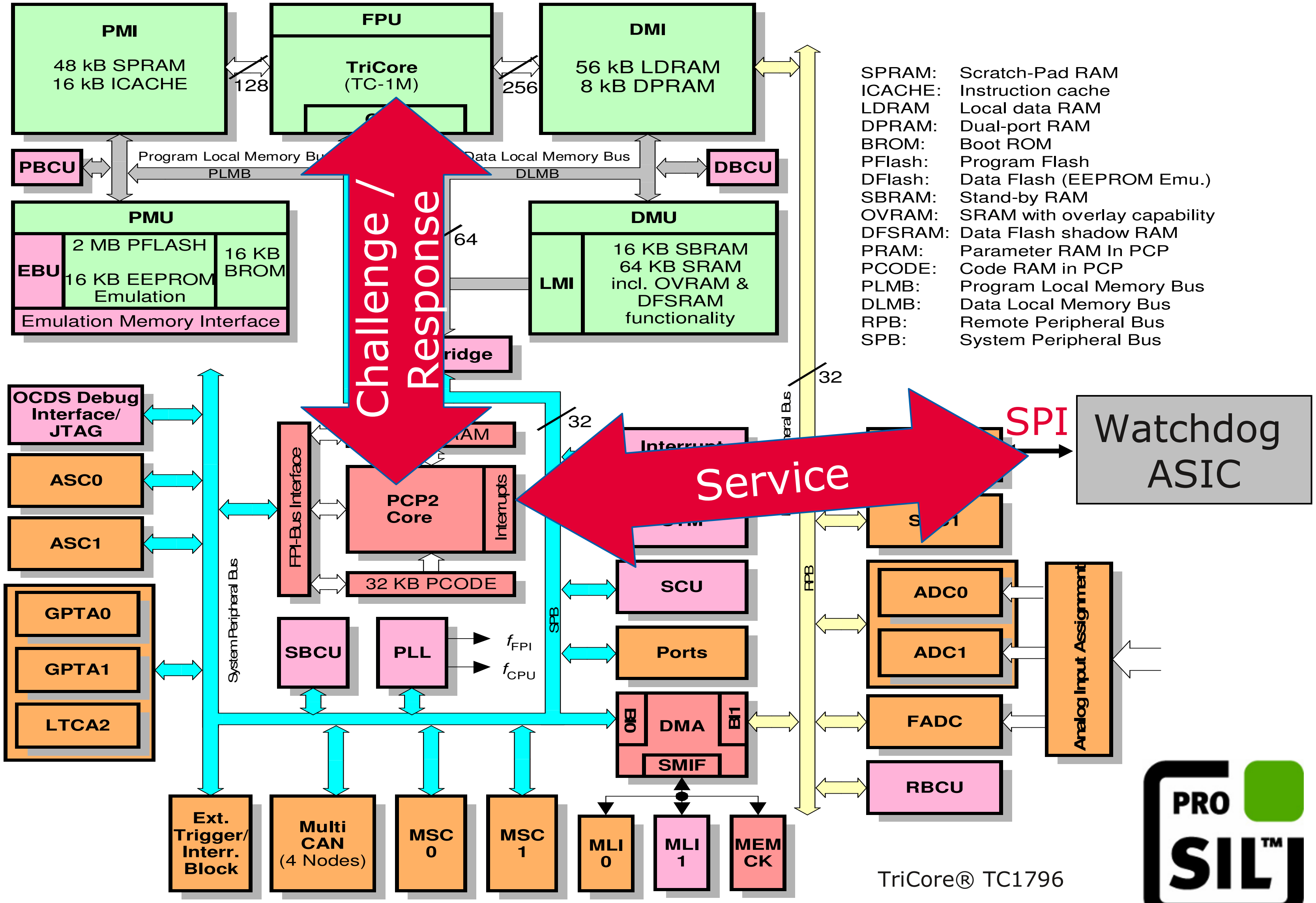


Table of contents

Example: TriCore & PCP as asymmetric dual core to support functional independent safety



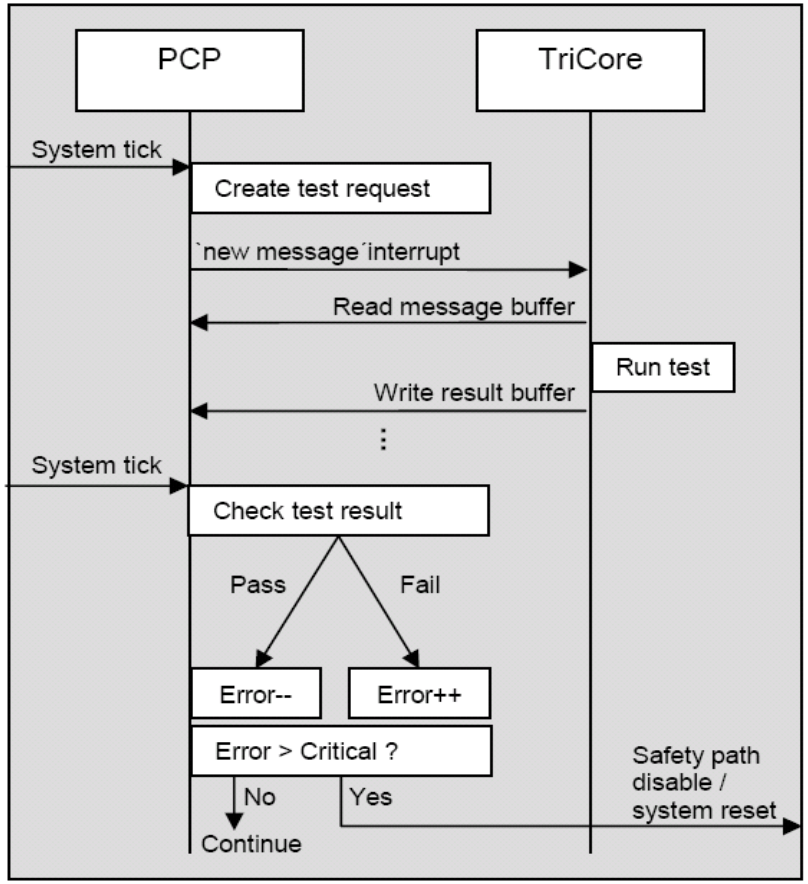
SPRAM: Scratch-Pad RAM
ICACHE: Instruction cache
LDRAM Local data RAM
DPRAM: Dual-port RAM
BROM: Boot ROM
PFlash: Program Flash
DFlash: Data Flash (EEPROM Emu.)
SBRAM: Stand-by RAM
OVRAM: SRAM with overlay capability
DFSRAM: Data Flash shadow RAM
PRAM: Parameter RAM In PCP
PCODE: Code RAM in PCP
PLMB: Program Local Memory Bus
DLMB: Data Local Memory Bus
RPB: Remote Peripheral Bus
SPB: System Peripheral Bus

Safety Driver Solution

PRO-SIL™ safety driver provides a scalable and state of the art solution – supplied by the silicon vendor

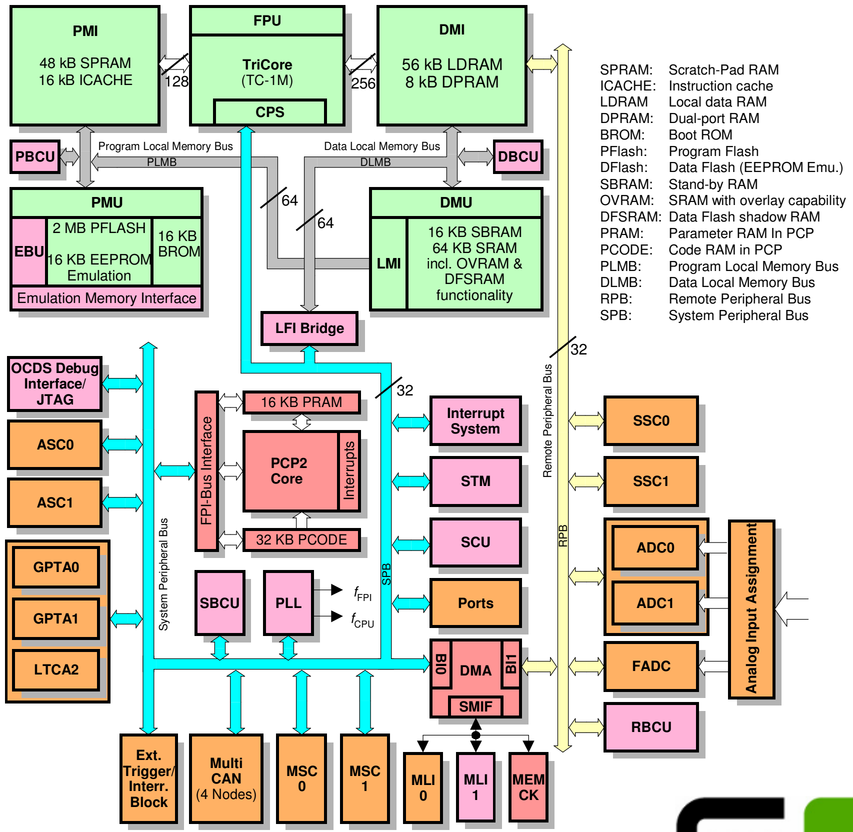
- Covering functional independent parts
- Supporting functional dependent parts

	Boot-Time or Shutdown	Runtime
Flash Checksum	X	Slices
SRAM Tests	X	Slices
Interrupt System Tests	X	
Opcode Tests	X	X
Program Flow Monitoring		X
Task Execution Timing		X
Internal Bus Tests	X	X
Inter Core Comm. Tests	X	X
Timer Tests	X	X
Internal Watchdog Tests	X	X
External Watchdog Tests	X	
Analog Converter Tests	X	
Peripheral Tests	X	X
CAN Monitoring		X
FlexRay Monitoring		X



Some PRO-SIL™ safety driver mechanisms

- Op-Code check mechanism
  - Coverage of 99% of used silicon in TriCore and PCP
  - Test running within failure reaction time
- Usage of all TriCore build in safety features
- Task Execution Monitor for functional depended software
- Test Execution Monitor
- Error injection mechanisms
  - Test The Tester
- Operation and runtime environment independent
- Application independent



TriCore® TC1796

Table of contents

Conclusions

- Designing Applications following existing (IEC61508) and new (ISO26262) standards will be THE challenge for safety critical automotive systems
  - Process implementation as major effort
  - Quality requirements to be meet with current systems
- Several safety concepts to fulfill IEC61508 are existing or in preparation
- Software is the major issue for safe systems for all involved partners
  - OEM
  - ECU supplier
  - Semiconductor vendor
- Requirements to supply safe software do not depend on fully qualified tool chains
  - Safe Software can be done with in limits of nowadays existing software development processes

References:

- IEC 61508-3 (2006), Functional Safety Of Electrical/Electronic Programmable Electronic Safety Related Systems
- Basic Single Microcontroller Monitoring Concept for Safety Critical Systems. (2007) Schneider, Eberhard, Brewerton. SAE#2007-01-XXXX (07AE-35)
- Implementation of a Basic Single-Monitoring Concept for Safety Critical Systems on a Dual- Core Microcontroller. (2007) Schneider, Eberhard, Brewerton. SAE#2007-01-XXXX (07AE-36)

Never stop thinking