Developing intelligent automotive systems with functional safety

Optimised, efficient SoC technology powering innovation in automotive

18th July 2018
Cadence Automotive Seminar

Developing intelligent automotive systems with functional safety

- Automotive markets trends
- Technical challenges
- Functional safety

James Scobie

Senior Product Manager
Arm, Cambridge
Embedded & Automotive LoB

Ann Keffer

Product Management Director
Cadence Design Systems, San Jose
System and Verification Group

The most complex piece of electronics you will own

Increasing complexity in functional safety markets

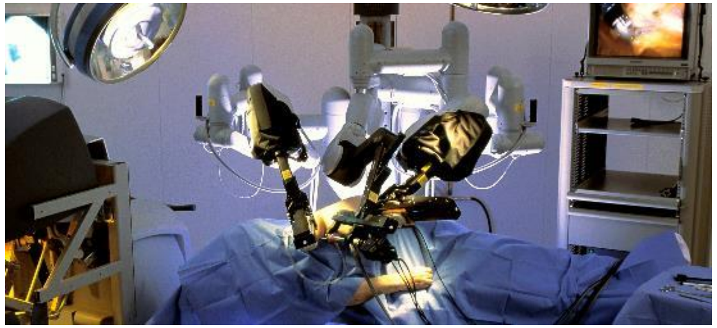
Automotive
Autonomous driving

Transportation
Train control systems

Industrial
Factory automation

Avionics
Flight systems

Healthcare
Robotic surgery

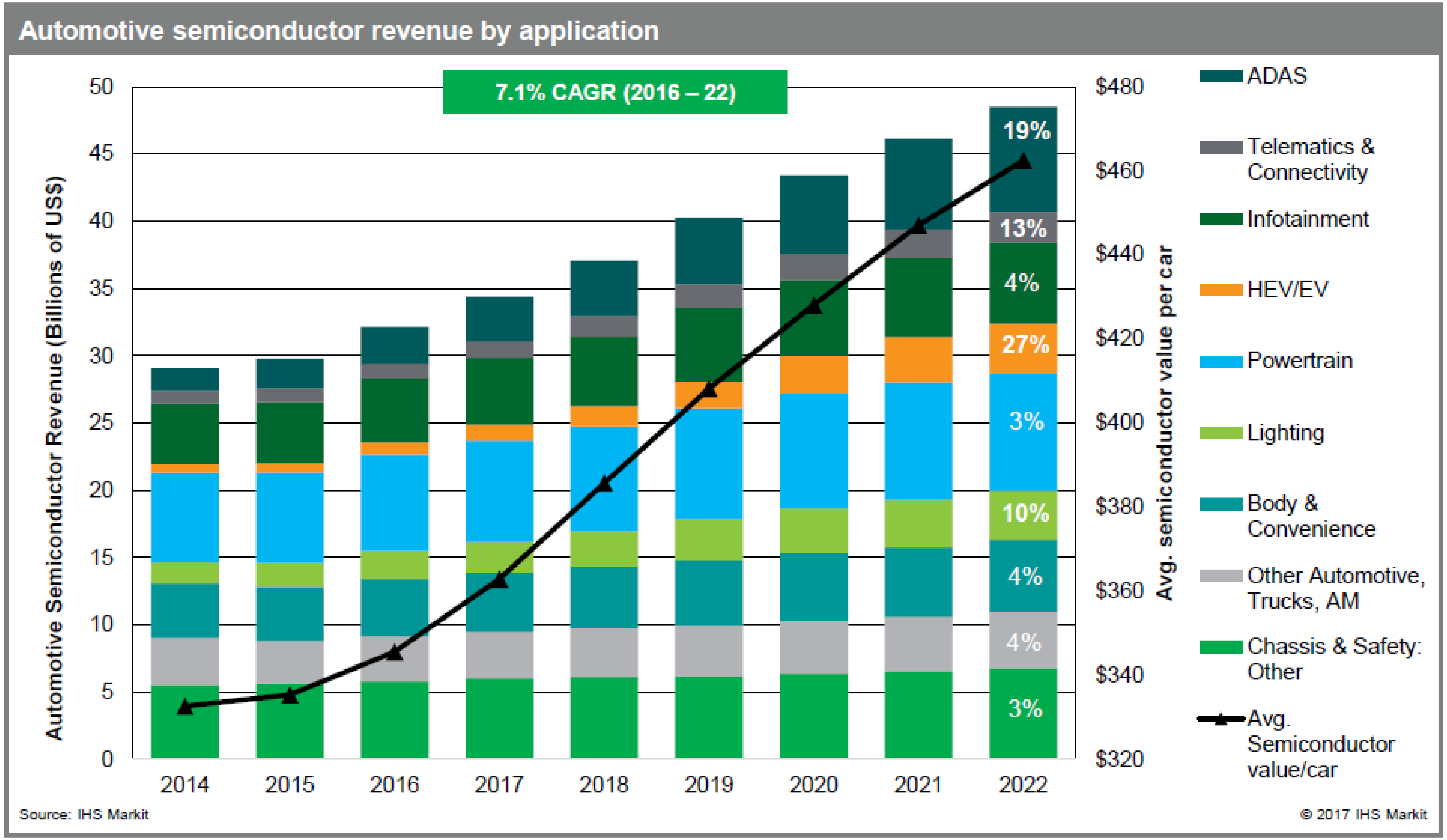


Consumer
Domestic robots

arm

cādence®

Automotive semiconductor growth

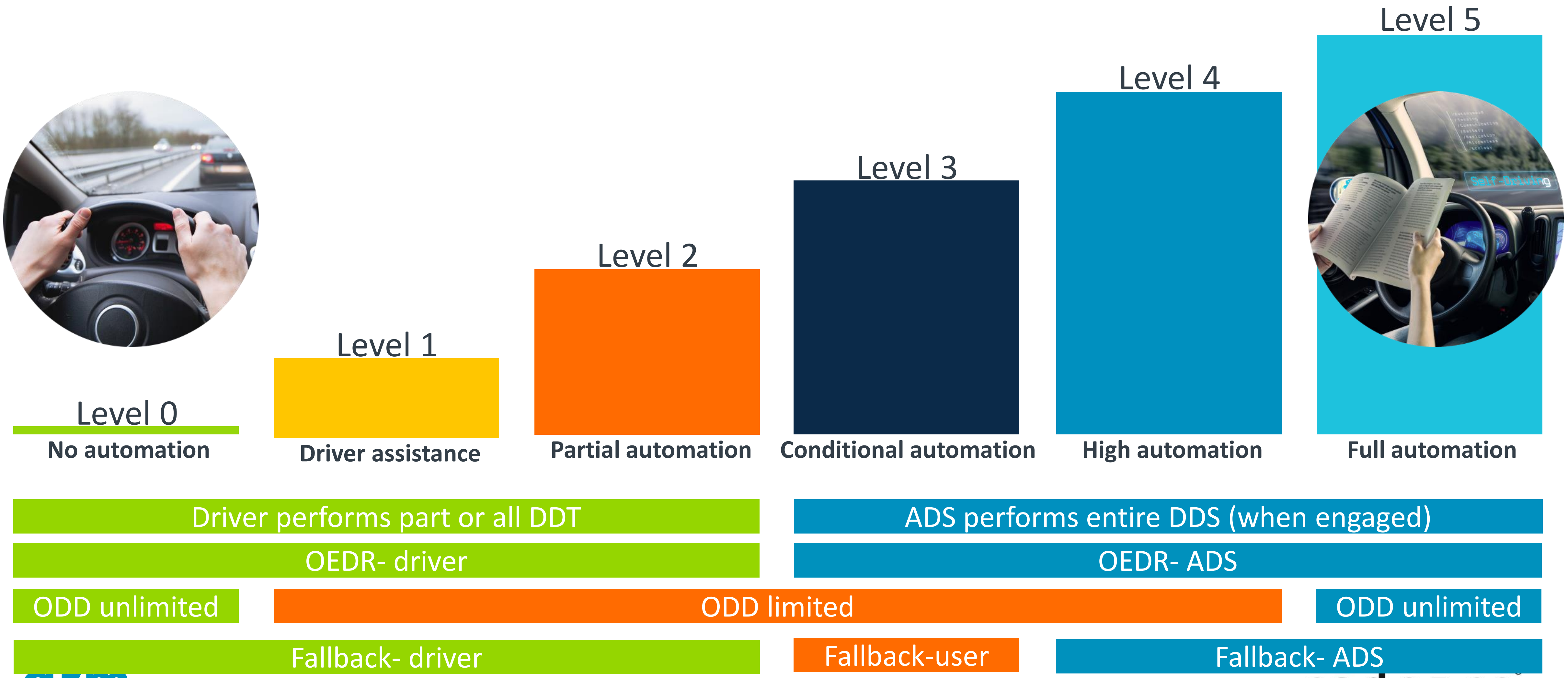


Source: IHS Markit

© 2017 IHS Markit

Source: IHS Markit, 2017

Autonomous vehicles



Level 0 — No automation

Level 1 — Driver assistance

Level 2 — Partial automation

Level 3 — Conditional automation

Level 4 — High automation

Level 5 — Full automation

Driver performs part or all DDT | ADS performs entire DDS (when engaged)

OEDR- driver | OEDR- ADS

ODD unlimited | ODD limited | ODD unlimited

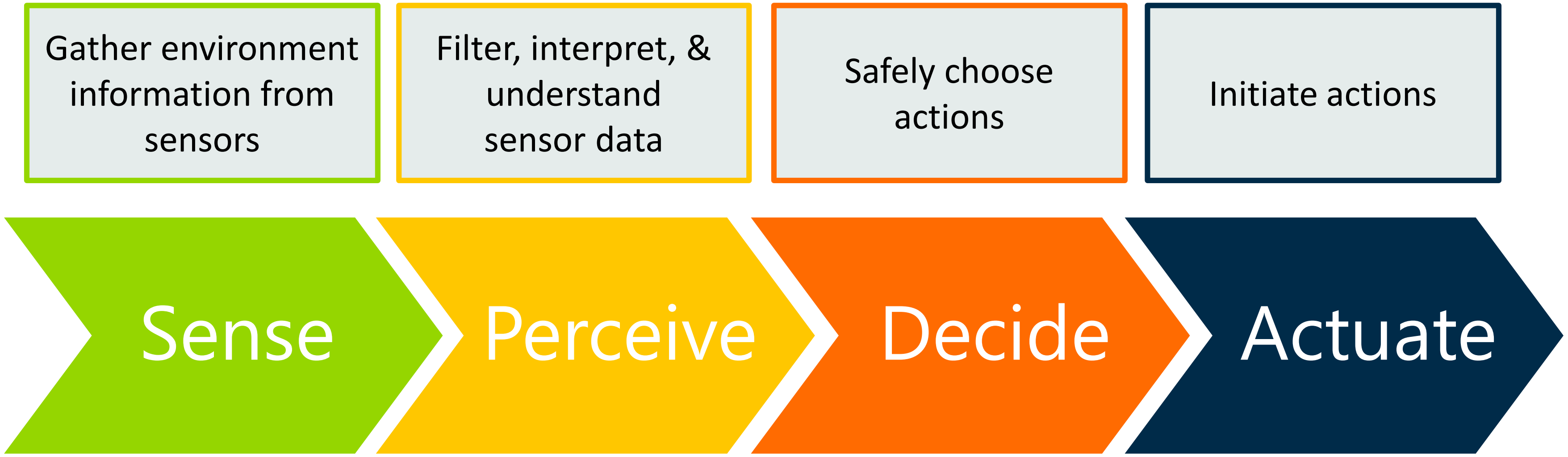
Fallback- driver | Fallback-user | Fallback- ADS

"Almost 80% of automotive innovation comes from electronics (semiconductors) and software"

– Audi at CES Asia

The foundation for autonomous systems

Autonomous system



Gather environment information from sensors

Filter, interpret, & understand sensor data

Safely choose actions

Initiate actions

Sense

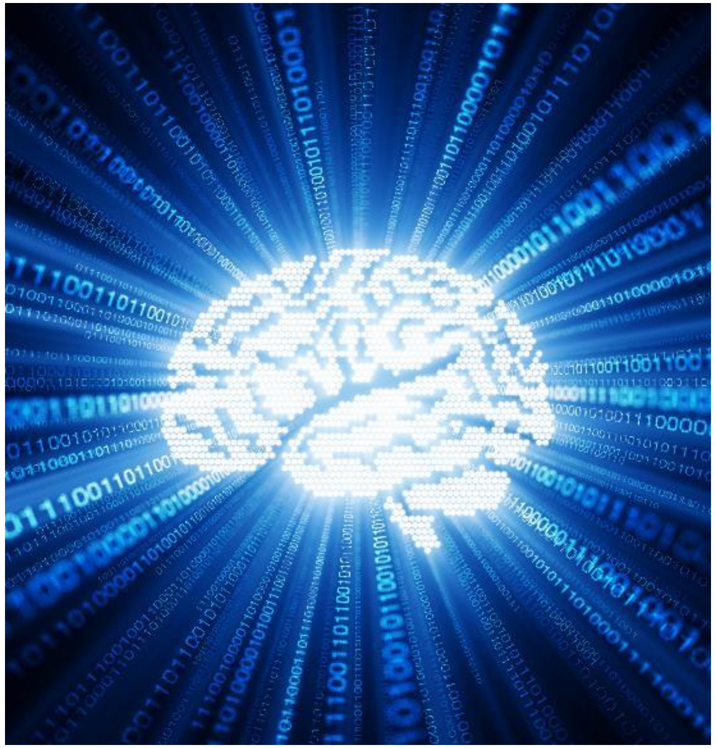
Perceive

Decide

Actuate

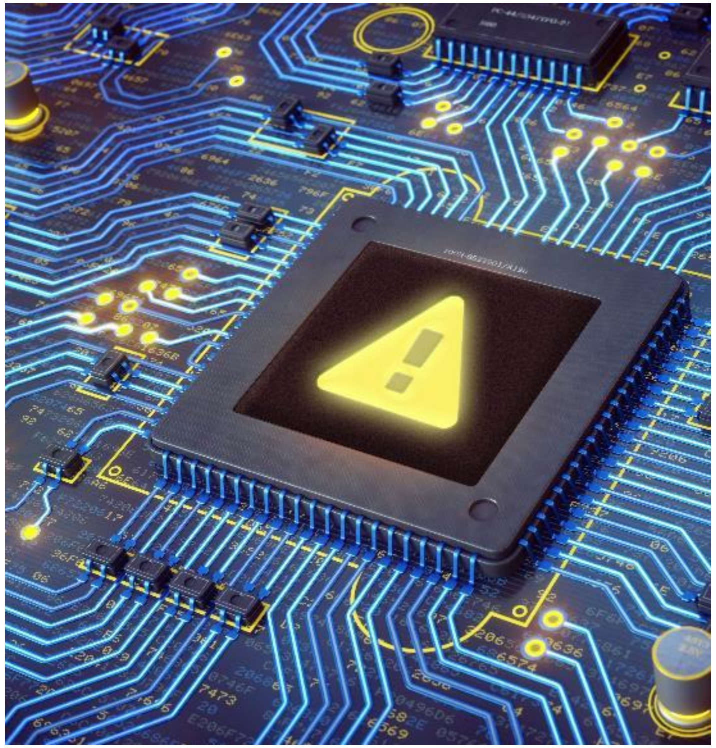
What are the challenges?

Complex and demanding compute requirements



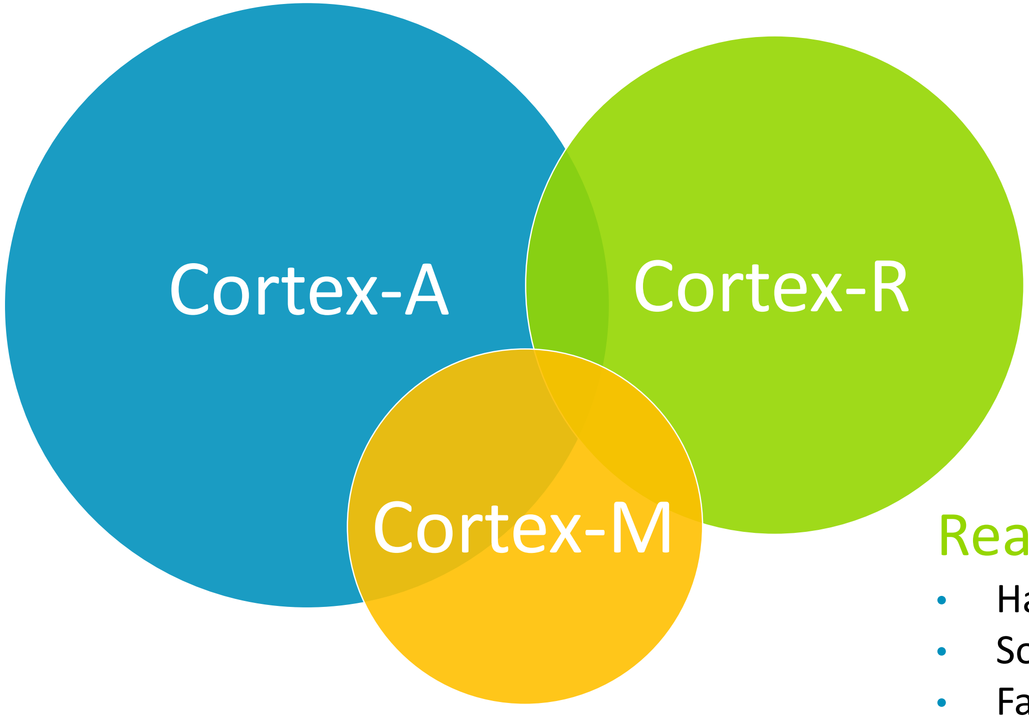
Increasing need for security

Rising functional safety requirement



Arm® Cortex® processors offer a range of choices

Complex and demanding compute requirements



Highest performance

- Sophisticated virtual memory support for rich OS
- Advanced programmer's model
- Software-managed interrupts
- Multi-core and multi-cluster
- Arm TrustZone® technology support

Real-time processing performance

- Hard real-time deterministic
- Software-managed interrupts
- Fast interrupts
- Multi-core
- Hardware virtualization (in Armv8-R)

Smallest area and lowest power profile

- Standardized memory map, optimized for RTOS
- Simple programmer's model
- Hardware-managed interrupts and lowest latency
- TrustZone technology in Armv8-M

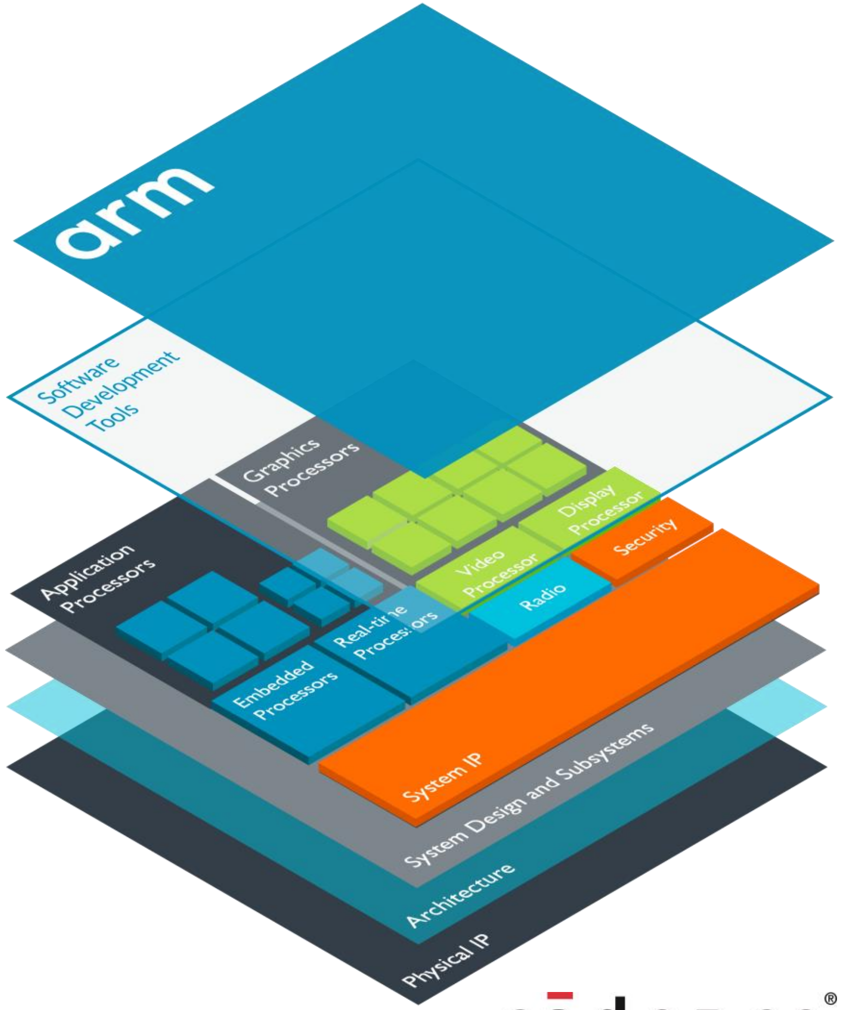
*Size of bubble indicates increasing system and software complexity

Flexible solutions need a range of capabilities

Heterogeneous compute requirements

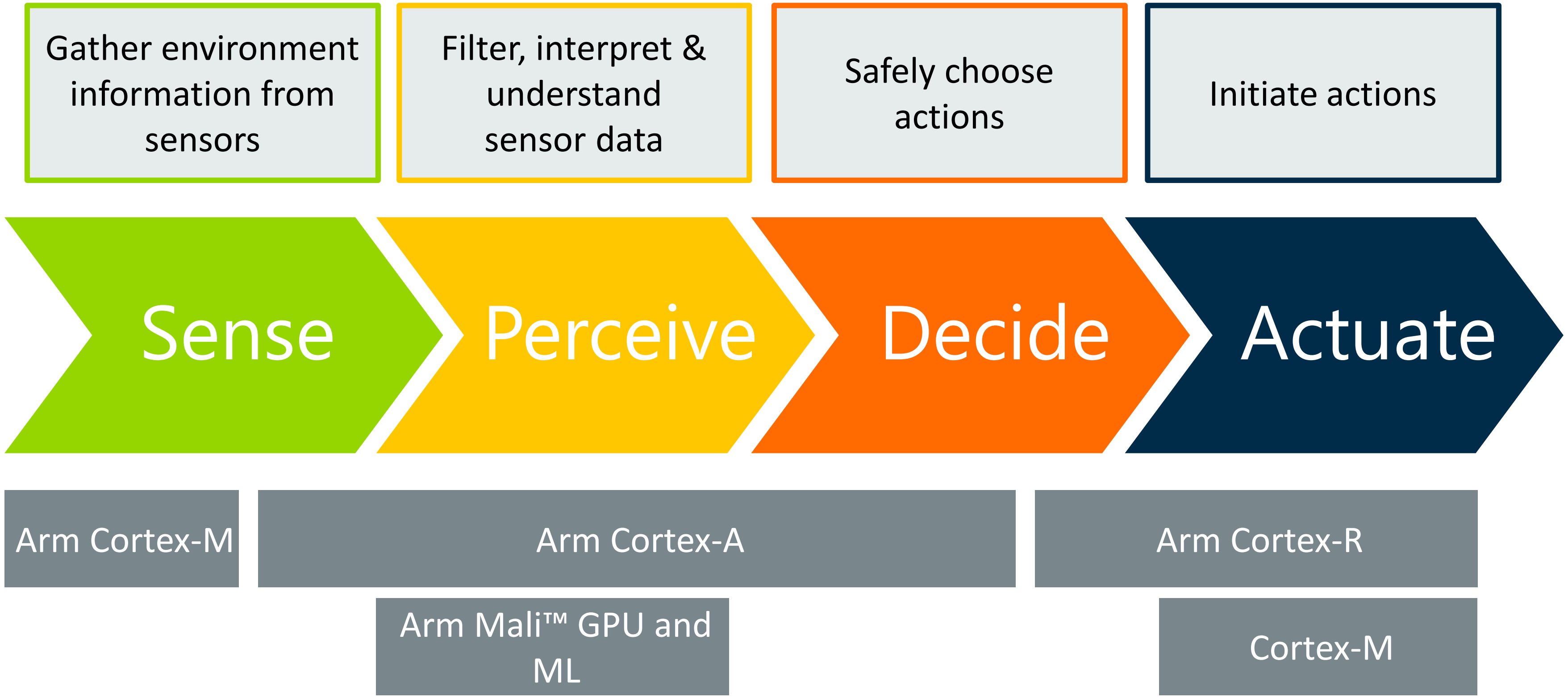
Mix of IP and solution

- Compute capability to meet the requirements
  - Within the constrained power window
- Accelerators
  - Diverse components designed for specific tasks
- System IP
  - Interconnect system IP delivering coherency and the quality of service required for lowest memory bandwidth
- Software
  - Increasing system efficiency with optimized software
- Subsystems
  - Efficient integration



Arm: the foundation for autonomous systems

Autonomous system



Gather environment information from sensors

Filter, interpret & understand sensor data

Safely choose actions

Initiate actions

Sense

Perceive

Decide

Actuate

Arm Cortex-M

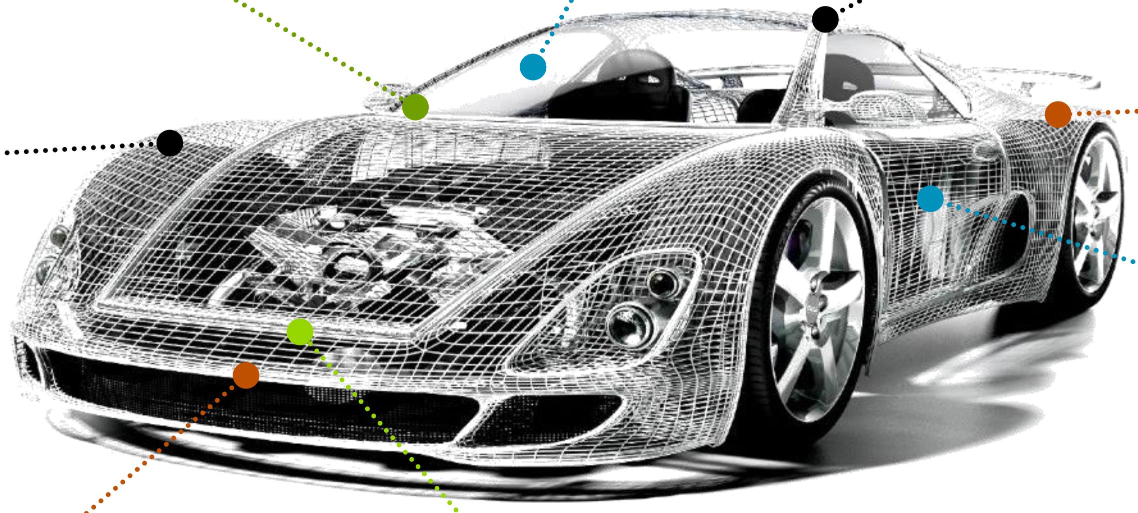
Arm Cortex-A

Arm Cortex-R

Arm Mali™ GPU and ML

Cortex-M

Autonomous vehicle security challenges



Firmware

- Firmware Rollback
- Malicious Firmware Update

Mobile applications

- Malicious Mobile Applications Synchronization

Connected Vehicle Services

- Code Bugs or Non Secure Code Attack
- Download Attacks

Connected Vehicle Services

- Mobile Device Malicious Application Synchronization

Wireless Communications

- MITM – Man in the Middle Attacks

Integrated Vehicle Security

- Integrated Attack on Keystore or KMS
- Weak Random Number Generation

ADAS / Autonomous Vehicle Controls Systems

- Spoofed Hardware Identity
- Compromised ECU via SW Injection

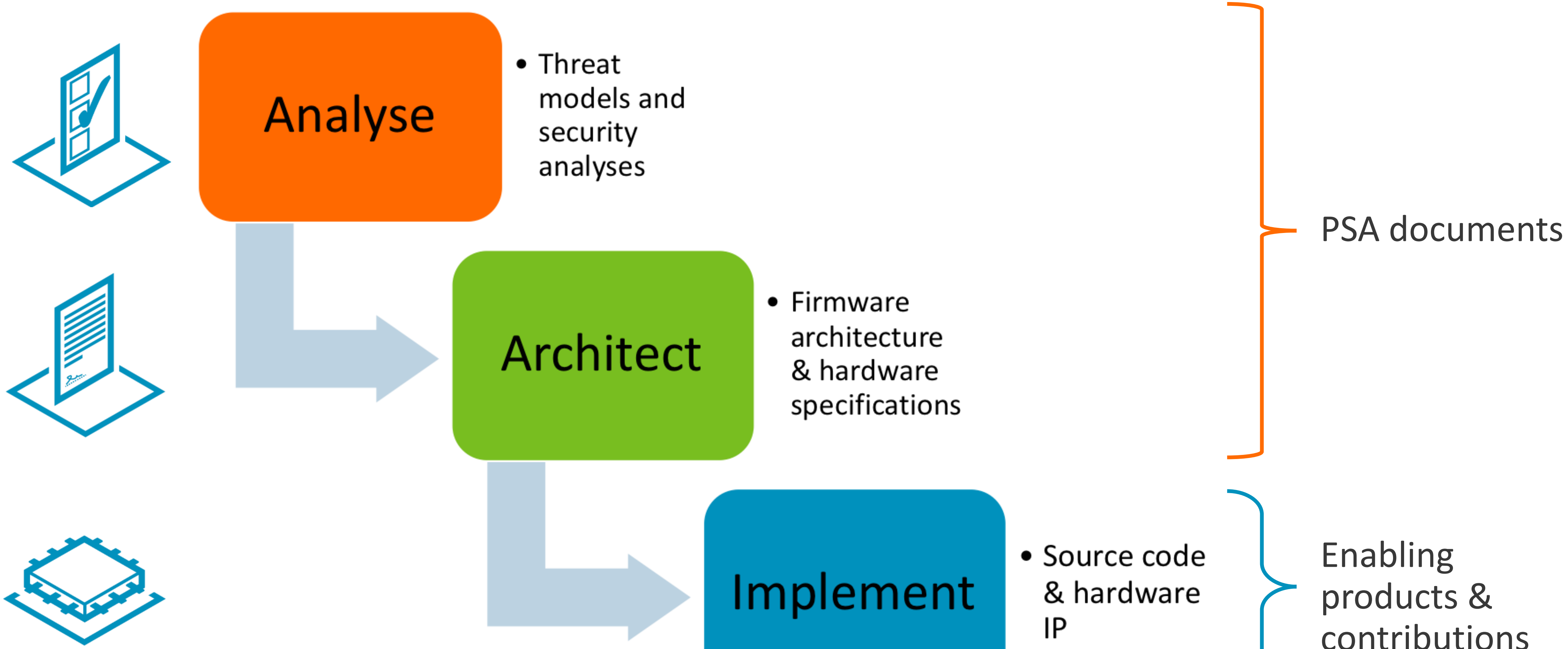
Vehicle Communication Busses

- Injection Attack on Vehicle Communication Busses
- Data Capture / Sniffing Communication Busses

Note : These characterizations are loose, subsystems may exist in multiple categories

Framework to secure 1 trillion devices…

Platform Security Architecture



Analyse

- Threat models and security analyses

Architect

- Firmware architecture & hardware specifications

Implement

- Source code & hardware IP

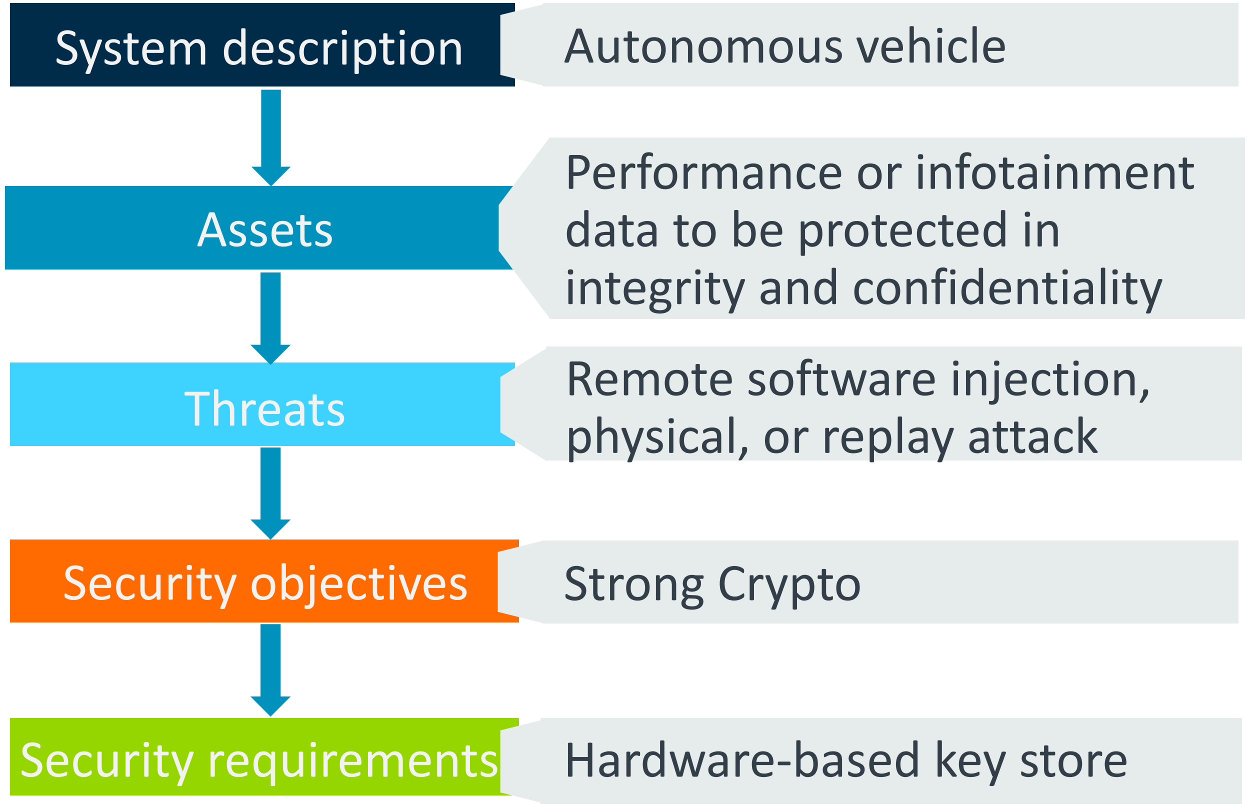
PSA documents

Enabling products & contributions

cadence

arm

Threat models and security analyses example

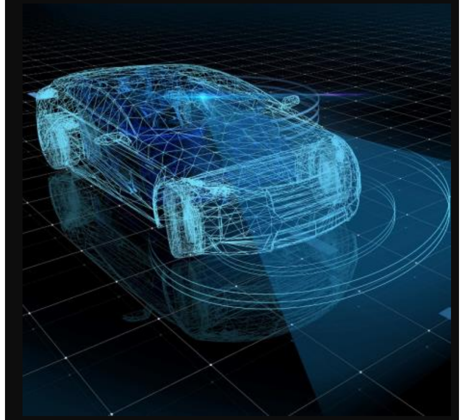


System description	Autonomous vehicle
Assets	Performance or infotainment data to be protected in integrity and confidentiality
Threats	Remote software injection, physical, or replay attack
Security objectives	Strong Crypto
Security requirements	Hardware-based key store

Functional safety controls risks of hazards

Rising functional safety requirement

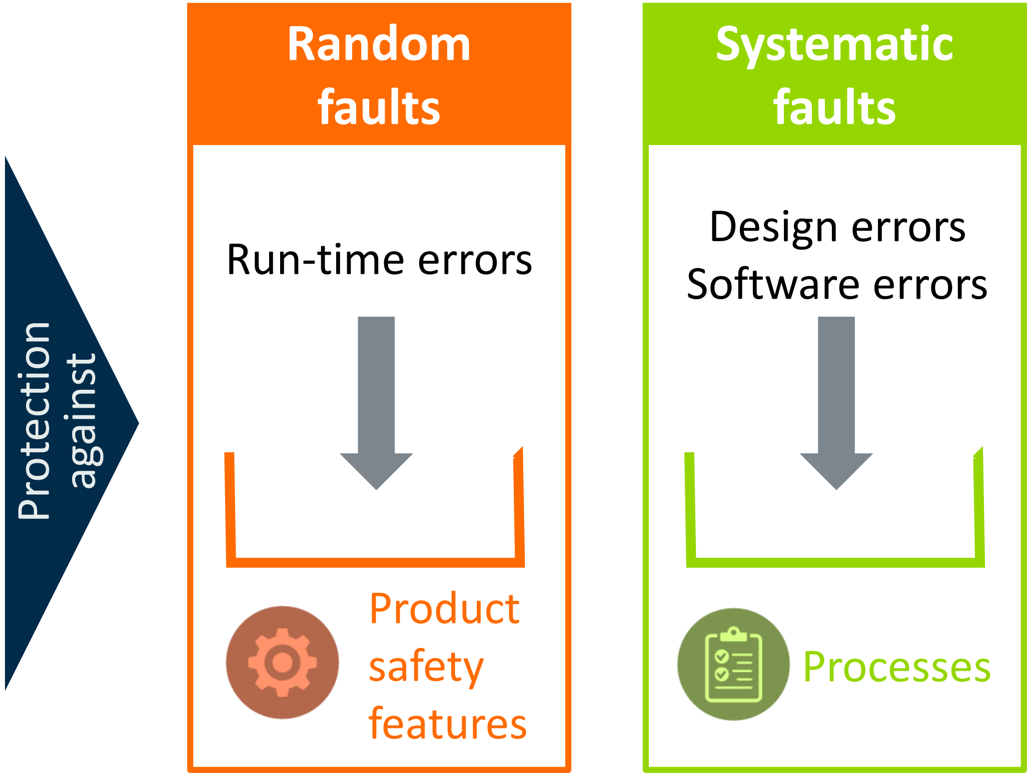
Safety application



Advanced Driver Assistance Systems

Safety application

Braking system



Protection against

Random faults

Run-time errors

Product safety features

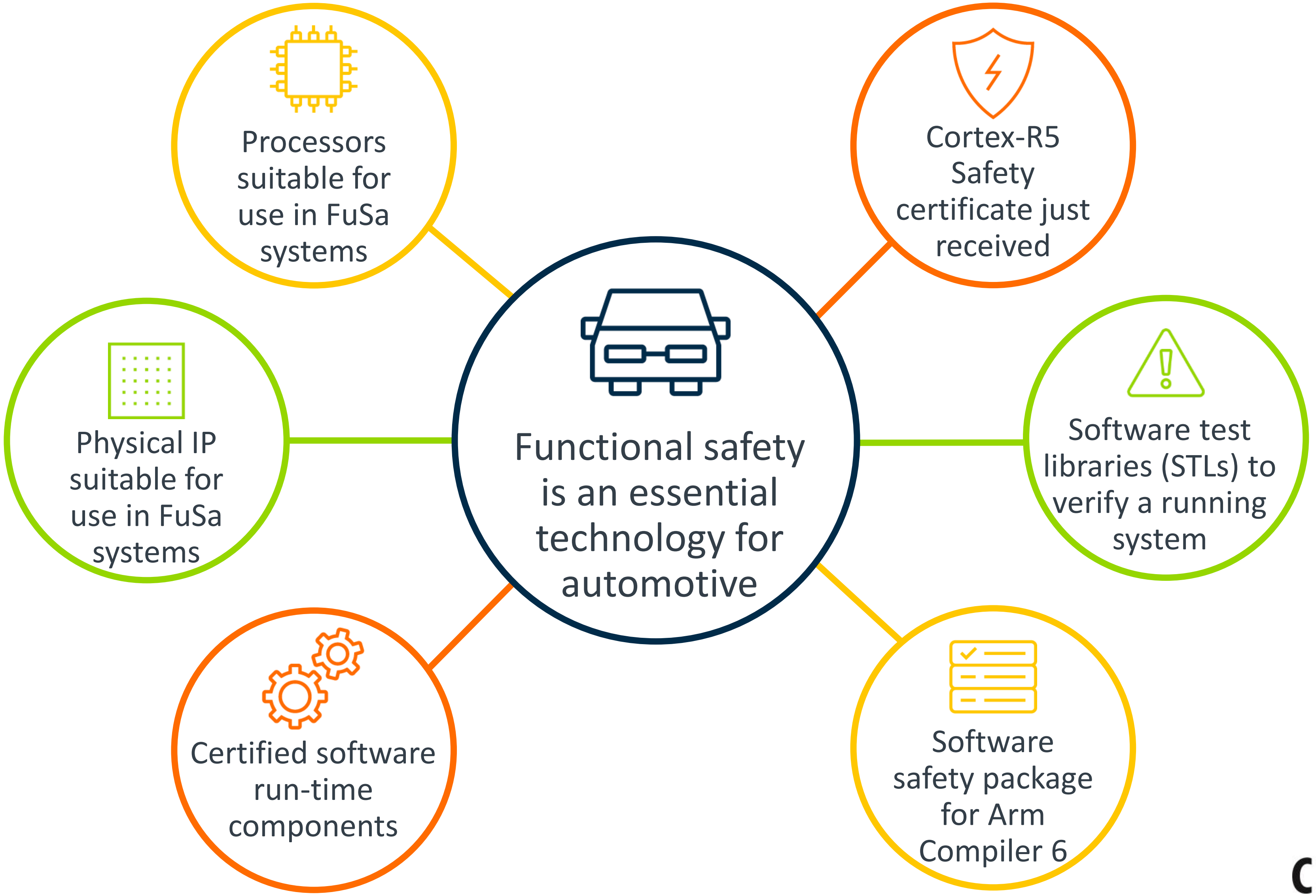
Systematic faults

Design errors
Software errors

Processes

"Absence of unreasonable risk due to hazards caused by malfunctions"

Functional safety (FuSa) essential for automotive applications



Functional safety is an essential technology for automotive

Processors suitable for use in FuSa systems

Cortex-R5 Safety certificate just received

Physical IP suitable for use in FuSa systems

Software test libraries (STLs) to verify a running system

Certified software run-time components

Software safety package for Arm Compiler 6

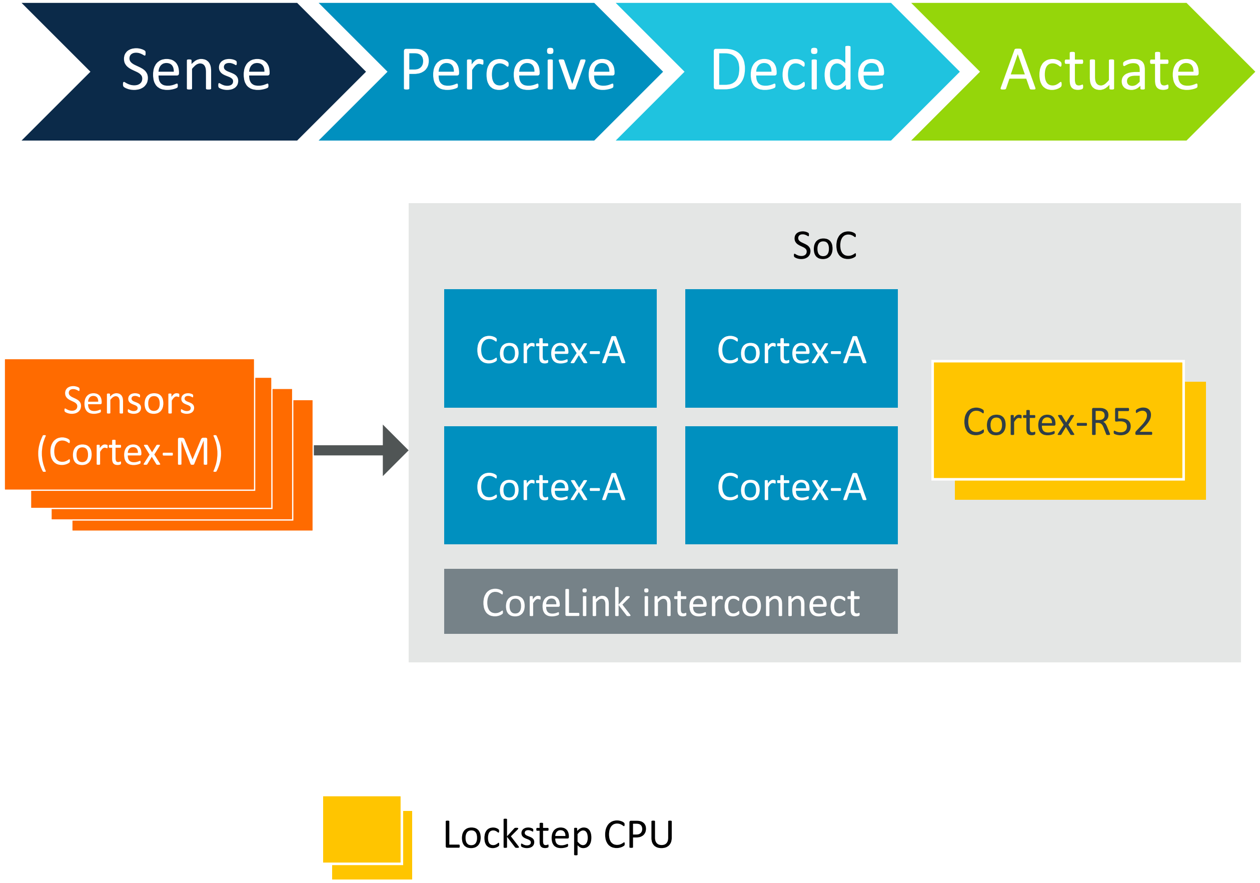
arm

cādence®

Safety island concept

Combine “safety island” with application processors

- Optimised real-time capability for actuation
- Integrate checker functions into SoC
- Sits on independent power and clock rails to reduce common cause failures
- Manages overall safety for SoC
- Enables both high compute with high safety integrity
- Reduces BOM cost and footprint



Arm functional safety package

Safety manual

- Design and verification process
- Fault detection and control
- Verification summary

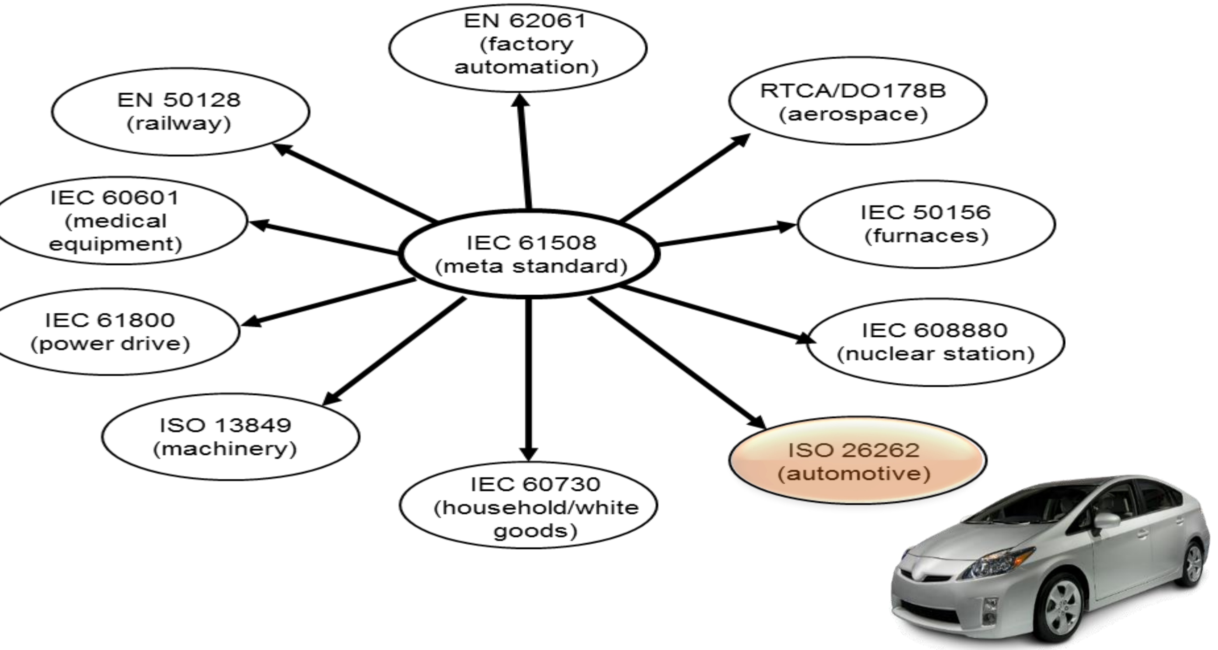
FMEA report

- Evidence of safety analysis on the Arm IP
- Aids partners with their own SoC level FMEA

Development Interface Report

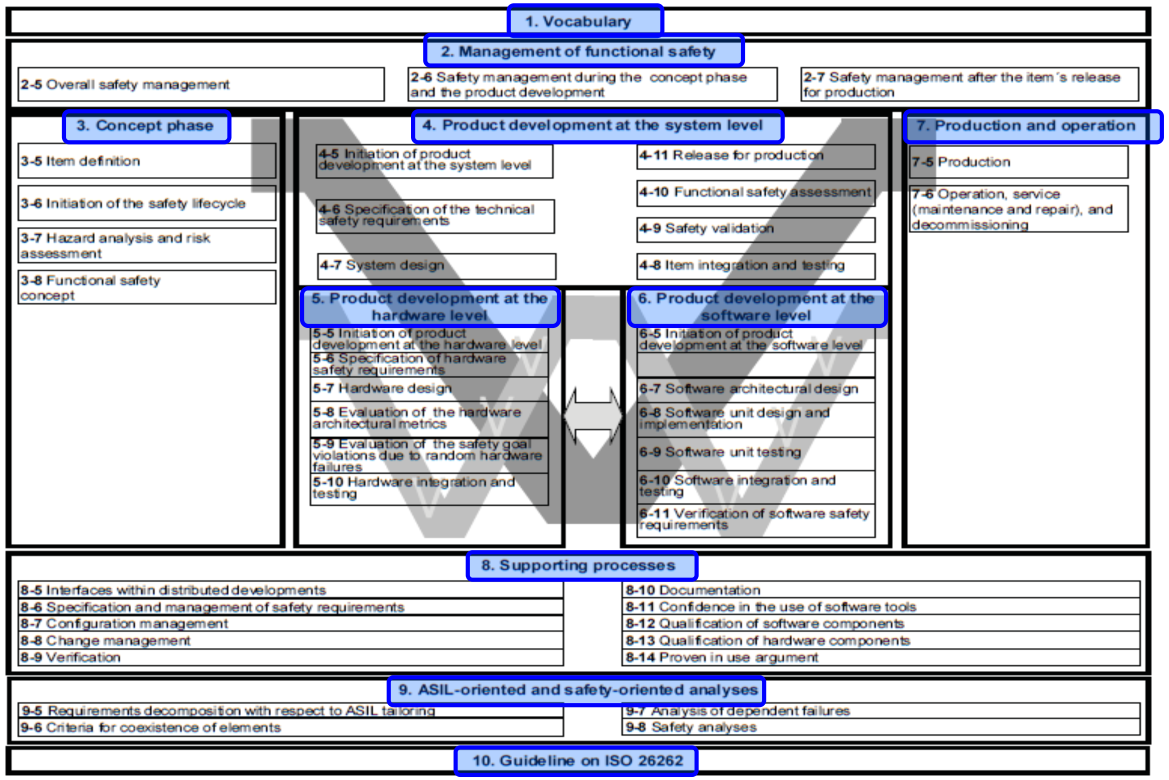
- Interworking relationship
- Replaces conventional DIA
- Ambiguity avoidance

Functional safety standards



ISO 26262 defines

- Processes to follow
- Hardware/software performance to achieve
- Safety documentation to produce
- Software tools compliance process



1. Vocabulary

2. Management of functional safety

2-5 Overall safety management

2-6 Safety management during the concept phase and the product development

2-7 Safety management after the item´s release for production

3. Concept phase

3-5 Item definition

3-6 Initiation of the safety lifecycle

3-7 Hazard analysis and risk assessment

3-8 Functional safety concept

4. Product development at the system level

4-5 Initiation of product development at the system level

4-6 Specification of the technical safety requirements

4-7 System design

4-11 Release for production

4-10 Functional safety assessment

4-9 Safety validation

4-8 Item integration and testing

5. Product development at the hardware level

5-5 Initiation of product development at the hardware level

5-6 Specification of hardware safety requirements

5-7 Hardware design

5-8 Evaluation of the hardware architectural metrics

5-9 Evaluation of the safety goal violations due to random hardware failures

5-10 Hardware integration and testing

6. Product development at the software level

6-5 Initiation of product development at the software level

6-7 Software architectural design

6-8 Software unit design and implementation

6-9 Software unit testing

6-10 Software integration and testing

6-11 Verification of software safety requirements

7. Production and operation

7-5 Production

7-6 Operation, service (maintenance and repair), and decommissioning

8. Supporting processes

8-5 Interfaces within distributed developments

8-6 Specification and management of safety requirements

8-7 Configuration management

8-8 Change management

8-9 Verification

8-10 Documentation

8-11 Confidence in the use of software tools

8-12 Qualification of software components

8-13 Qualification of hardware components

8-14 Proven in use argument

9. ASIL-oriented and safety-oriented analyses

9-5 Requirements decomposition with respect to ASIL tailoring

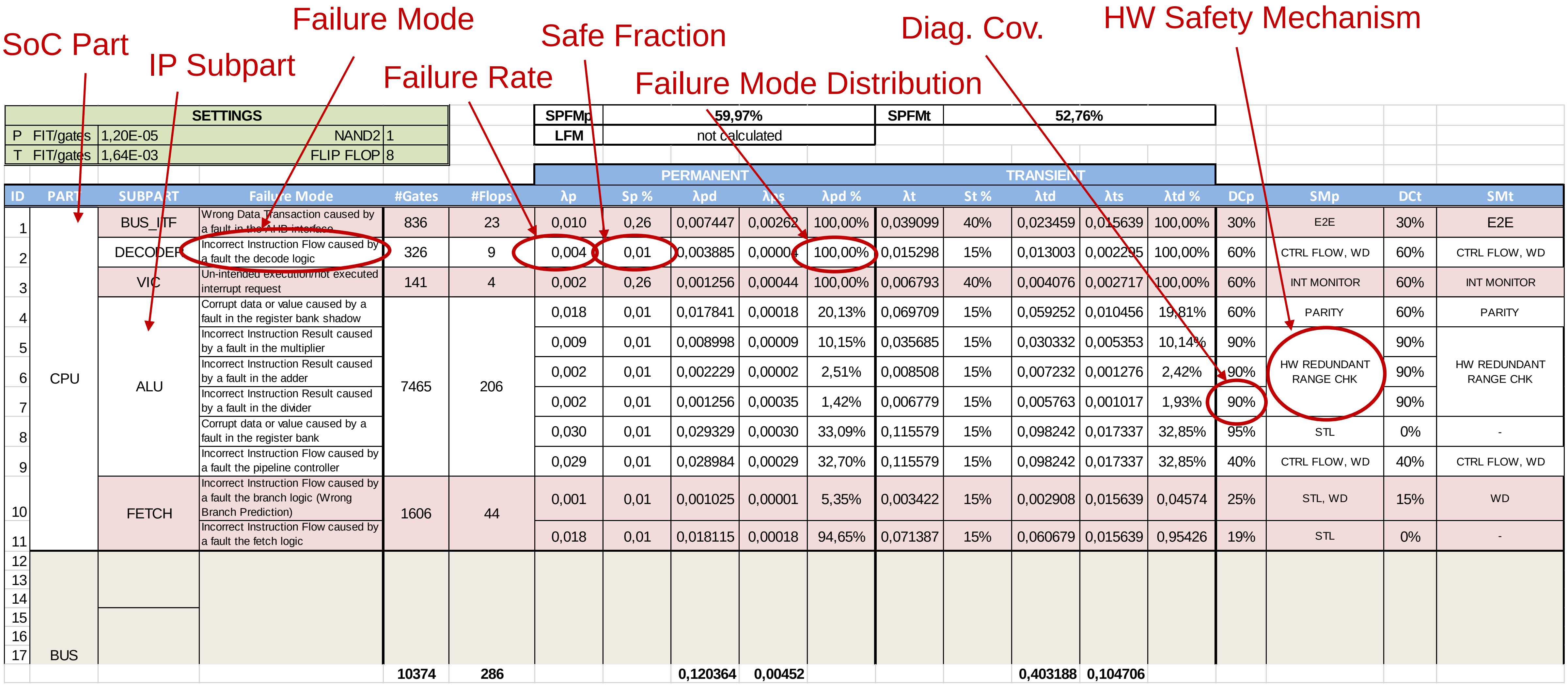
9-6 Criteria for coexistence of elements

9-7 Analysis of dependent failures

9-8 Safety analyses

10. Guideline on ISO 26262

FMEDA – capture and analyze safety goals



SETTINGS				
P	FIT/gates	1,20E-05	NAND2	1
T	FIT/gates	1,64E-03	FLIP FLOP	8

SPFMp	59,97%	SPFMt	52,76%
LFM	not calculated		

						PERMANENT					TRANSIENT								
ID	PART	SUBPART	Failure Mode	#Gates	#Flops	λp	Sp %	λpd	λps	λpd %	λt	St %	λtd	λts	λtd %	DCp	SMp	DCt	SMt
1	CPU	BUS_ITF	Wrong Data Transaction caused by a fault in the AHB interface	836	23	0,010	0,26	0,007447	0,00262	100,00%	0,039099	40%	0,023459	0,015639	100,00%	30%	E2E	30%	E2E
2		DECODER	Incorrect Instruction Flow caused by a fault the decode logic	326	9	0,004	0,01	0,003885	0,00004	100,00%	0,015298	15%	0,013003	0,002295	100,00%	60%	CTRL FLOW, WD	60%	CTRL FLOW, WD
3		VIC	Un-intended execution/not executed interrupt request	141	4	0,002	0,26	0,001256	0,00044	100,00%	0,006793	40%	0,004076	0,002717	100,00%	60%	INT MONITOR	60%	INT MONITOR
4		ALU	Corrupt data or value caused by a fault in the register bank shadow	7465	206	0,018	0,01	0,017841	0,00018	20,13%	0,069709	15%	0,059252	0,010456	19,81%	60%	PARITY	60%	PARITY
5			Incorrect Instruction Result caused by a fault in the multiplier			0,009	0,01	0,008998	0,00009	10,15%	0,035685	15%	0,030332	0,005353	10,14%	90%	HW REDUNDANT RANGE CHK	90%	HW REDUNDANT RANGE CHK
6			Incorrect Instruction Result caused by a fault in the adder			0,002	0,01	0,002229	0,00002	2,51%	0,008508	15%	0,007232	0,001276	2,42%	90%		90%	
7			Incorrect Instruction Result caused by a fault in the divider			0,002	0,01	0,001256	0,00035	1,42%	0,006779	15%	0,005763	0,001017	1,93%	90%		90%	
8			Corrupt data or value caused by a fault in the register bank			0,030	0,01	0,029329	0,00030	33,09%	0,115579	15%	0,098242	0,017337	32,85%	95%	STL	0%	-
9			Incorrect Instruction Flow caused by a fault the pipeline controller			0,029	0,01	0,028984	0,00029	32,70%	0,115579	15%	0,098242	0,017337	32,85%	40%	CTRL FLOW, WD	40%	CTRL FLOW, WD
10		FETCH	Incorrect Instruction Flow caused by a fault the branch logic (Wrong Branch Prediction)	1606	44	0,001	0,01	0,001025	0,00001	5,35%	0,003422	15%	0,002908	0,015639	0,04574	25%	STL, WD	15%	WD
11			Incorrect Instruction Flow caused by a fault the fetch logic			0,018	0,01	0,018115	0,00018	94,65%	0,071387	15%	0,060679	0,015639	0,95426	19%	STL	0%	-
12	BUS																		
13																			
14																			
15																			
16																			
17																			
				10374	286			0,120364	0,00452				0,403188	0,104706					

An SM can cover more than one FM

One FM can be covered by multiple SMs

Automotive SoC verification challenges

Systematic Failure Verification

Concurrent SW Development

Requirements Traceability

Use Case Verification

Performance Verification

Security Verification

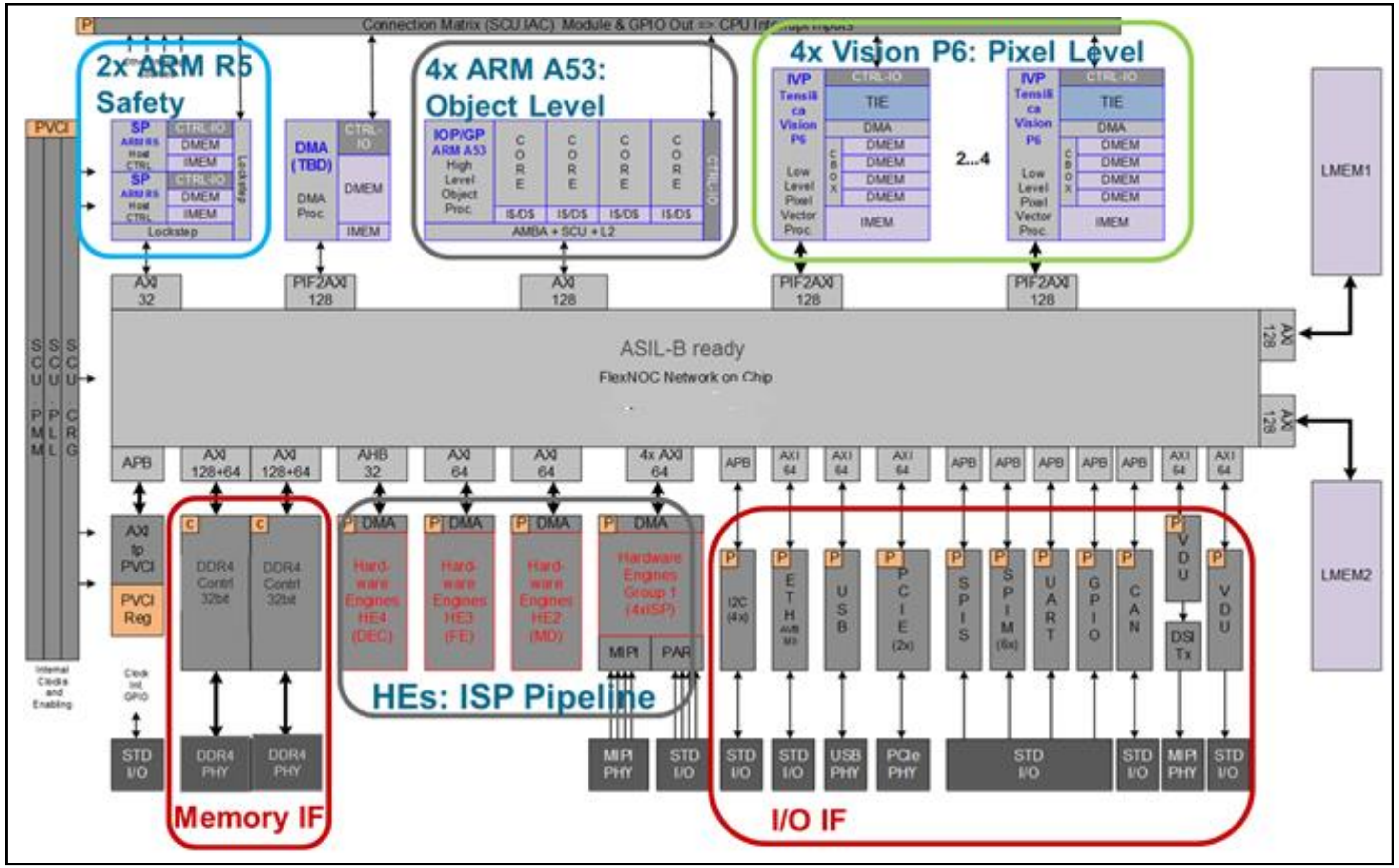
Automotive Protocol Verification

Mixed Signal Verification

Functional Safety Verification

Random Failure Verification

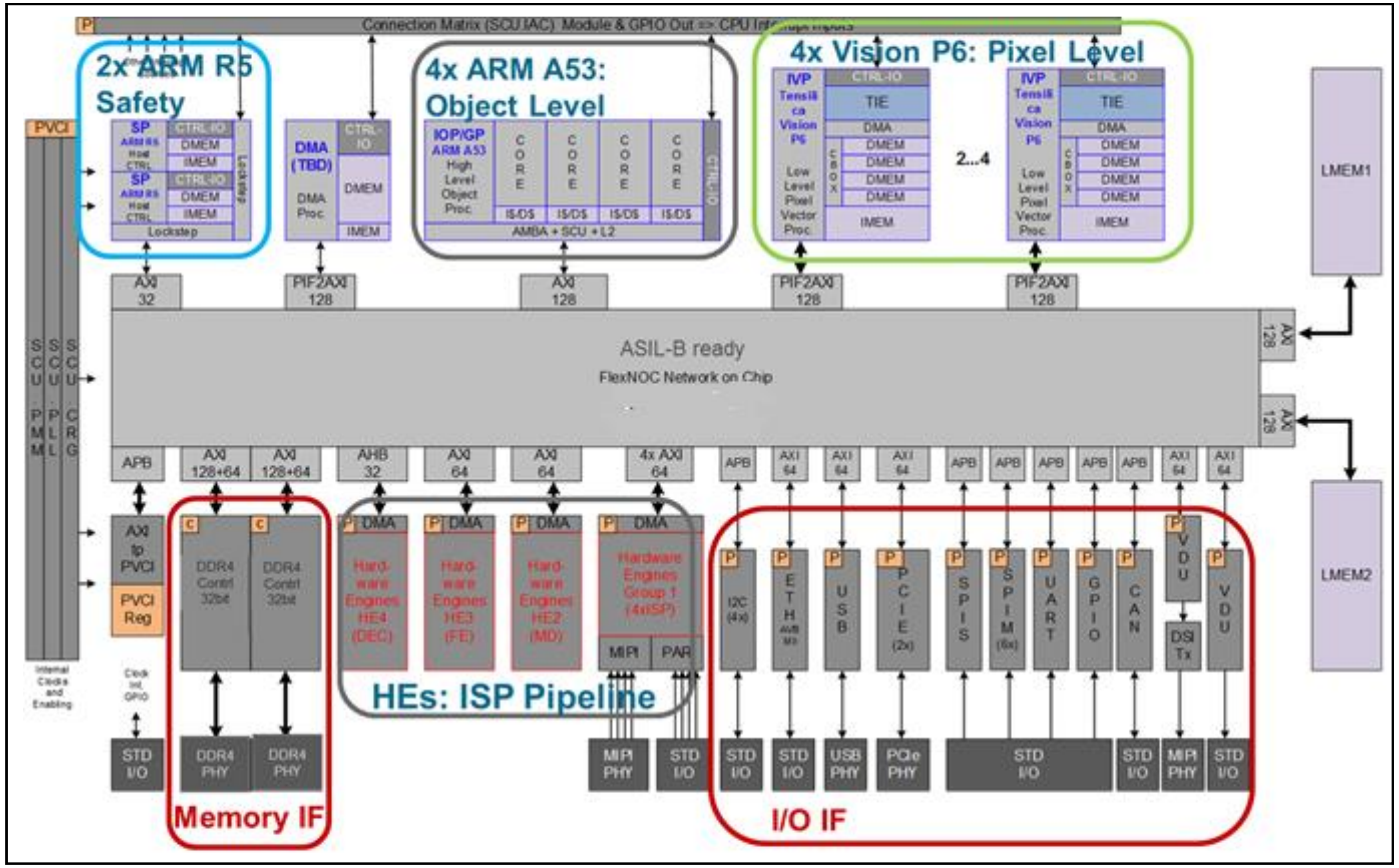
ADAS SoC Example



Automotive Functional Safety challenges

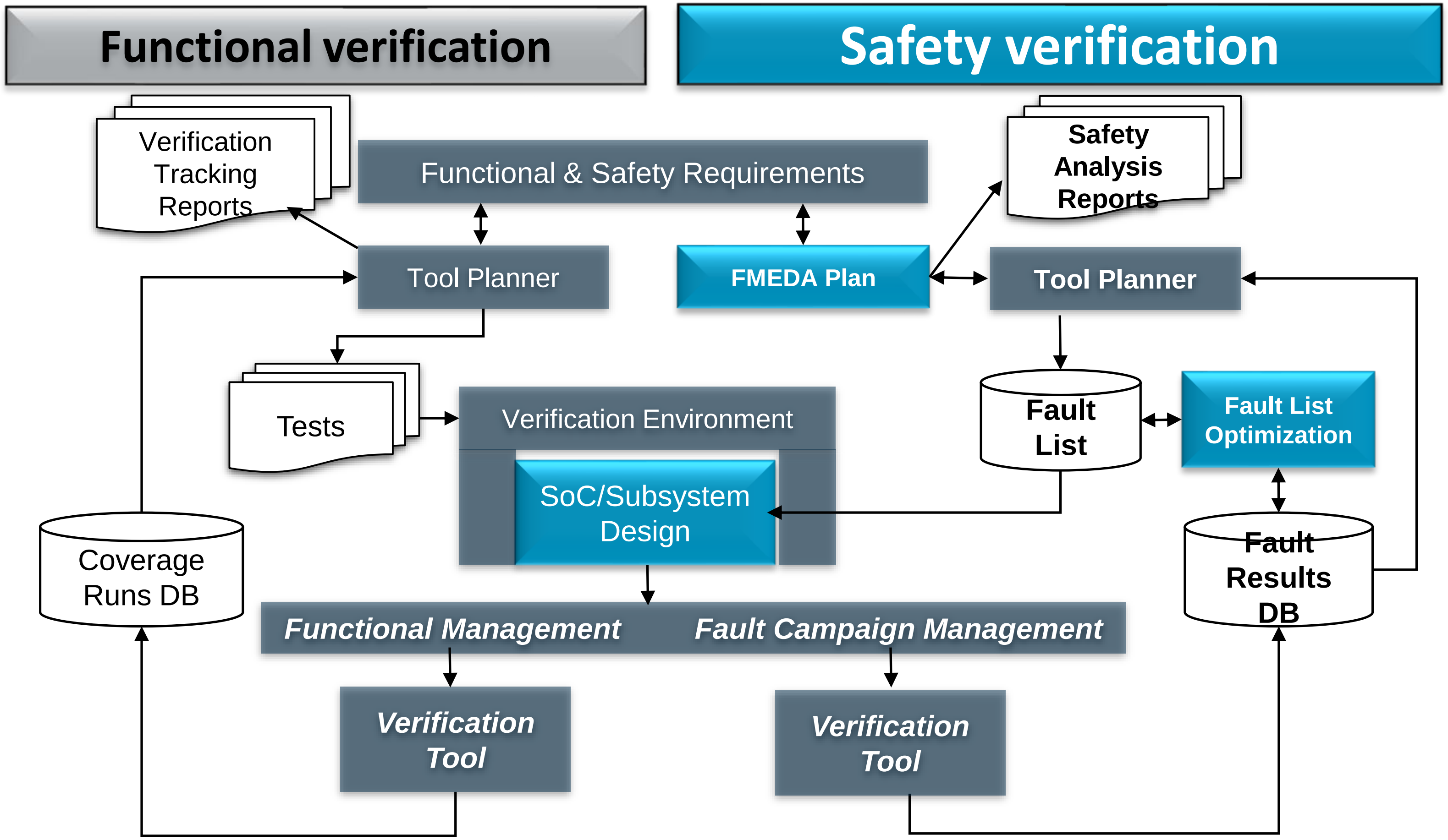
- Safety-Certified IP
- Failure Mode Definition
- Safety Mechanism Design
- Fault Campaign Planning
- Fault Reduction
- Safety Requirement Traceability
- Fault Execution
- Re-use of FV Environment
- Metric Calculation

ADAS SoC Example



Multiple verification engines and FMEDA Integration

Safety verification solution



Safe faults $\sum(\lambda_s)$	Detected dangerous faults $\sum(\lambda_{..})$	Undetected dangerous faults $\sum(\lambda_{..})$

- Unified functional + safety verification flow and engines
- Integrated fault campaign management across formal, simulation, and emulation
- Common fault results database unifies diagnostic coverage
- Proven requirements traceability, enabling FMEDA integration

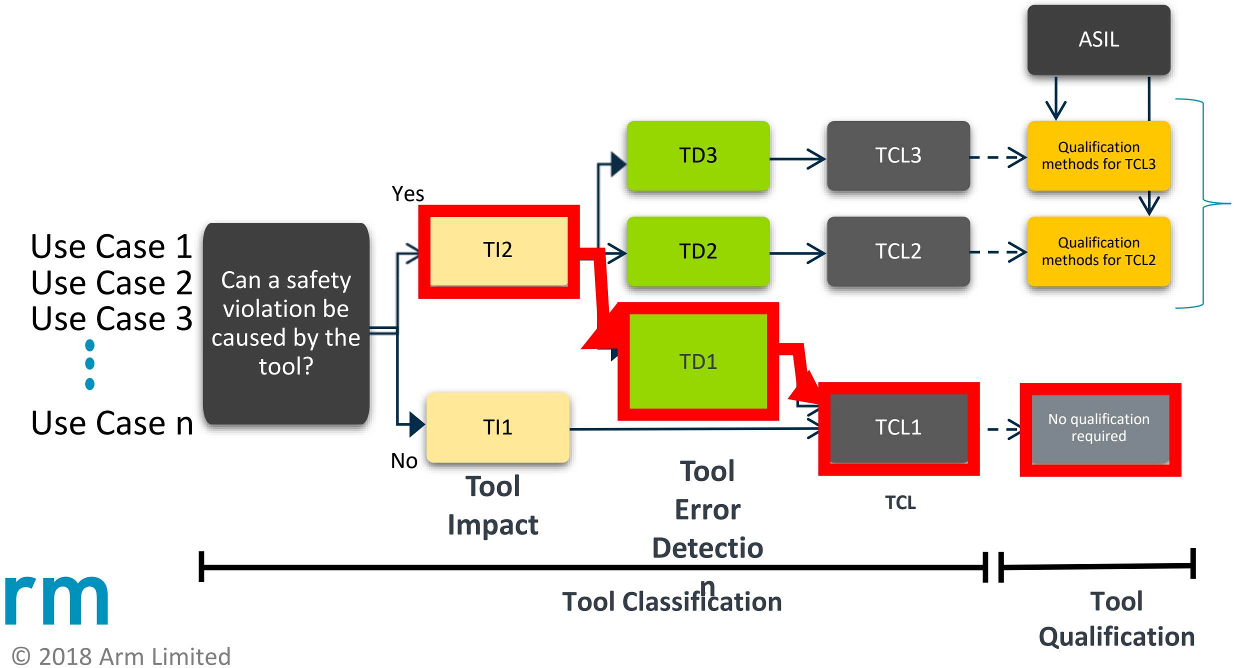
arm

cādence®

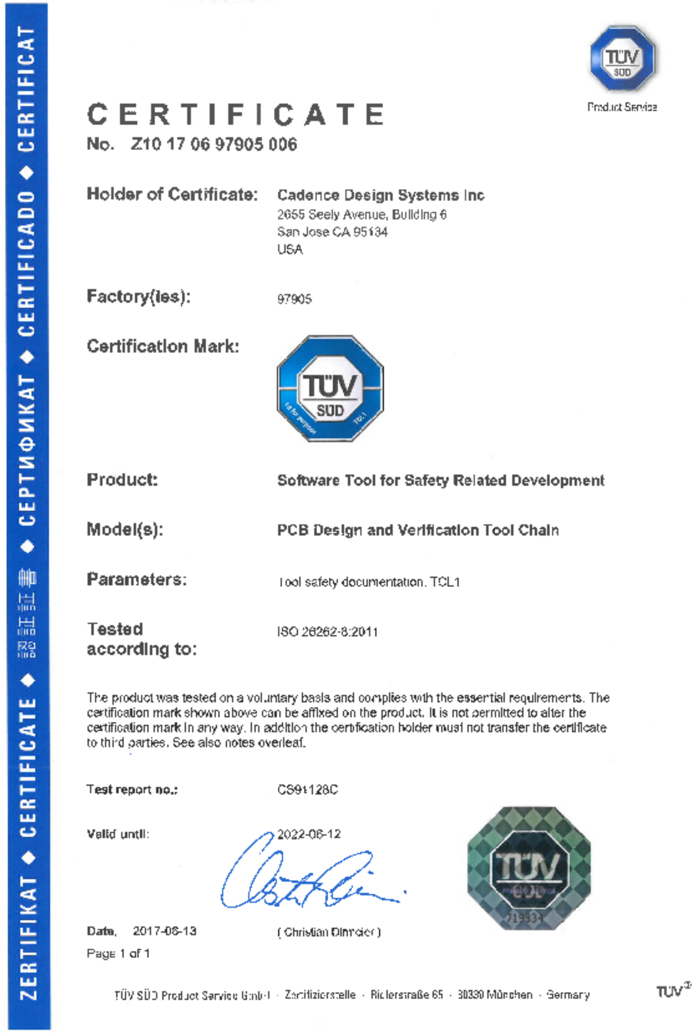
Cadence approach to ISO 26262 TCL certification

TUV SUD ISO 26262 certified documentation kits with TCL1 level confidence

- TCL1 reflects the highest confidence that tool malfunctions will not cause violations of safety requirements
- A tool-chain that evaluates to TCL1 will reduce the complexity, cost, and time required of our customers to certify their work products



Prove tools do not cause a safety issue



ZERTIFIKAT ◆ CERTIFICATE ◆ 認証書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

Product Service

CERTIFICATE

No. Z10 17 06 97905 006

Holder of Certificate: **Cadence Design Systems Inc**
2655 Seely Avenue, Building 6
San Jose CA 95134
USA

Factory(ies): 97905

Certification Mark:

Product: **Software Tool for Safety Related Development**

Model(s): **PCB Design and Verification Tool Chain**

Parameters: Tool safety documentation, TCL1

Tested according to: ISO 26262-8:2011

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: CS91128C

Valid until: 2022-06-12

Date, 2017-06-13

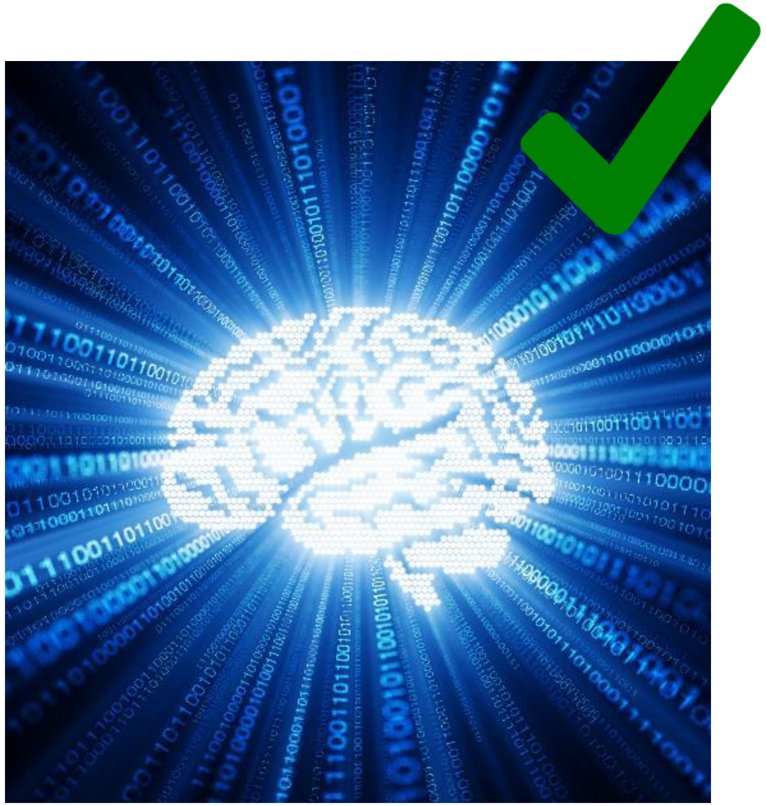
(Christian Dirmeier)

Page 1 of 1

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

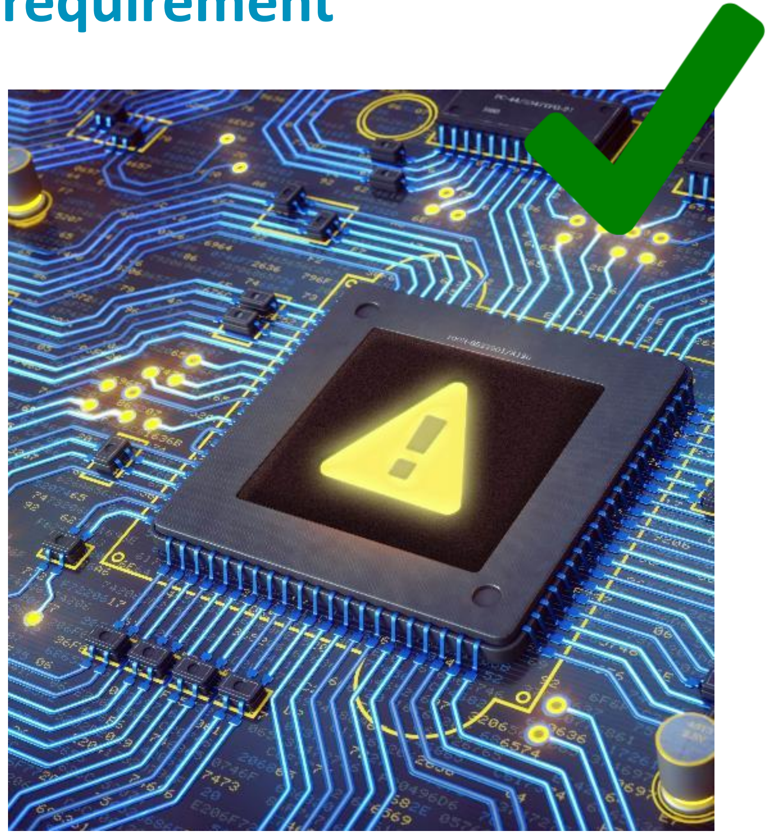
Summary

Complex and demanding compute requirements



Increasing need for security

Rising functional safety requirement



Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

감사합니다

धन्यवाद

arm

arm