DEVELOPING AN AUTOMOTIVE SAFETY ISLAND

Vehicle Architecture Trends

Electrification	ADAS / Autonomy	Connectivity

Increased compute requirements

Increased software content

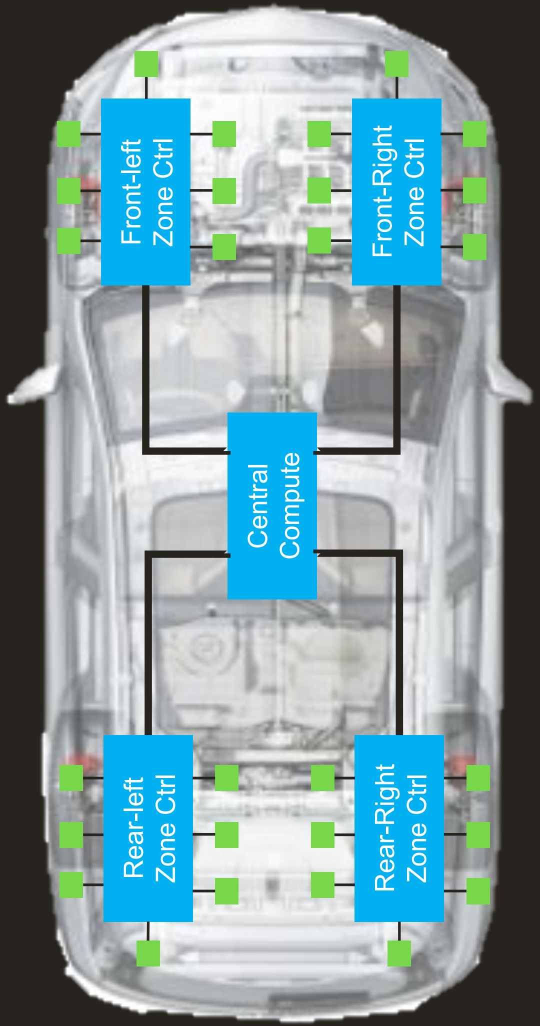
Safety remains paramount

Security becoming critical

Consolidation of compute resources

Drive for standardisation

Discrete ECUs → Domain Controllers → Zonal Controllers



Vehicle Architecture Trends

Electrification	ADAS / Autonomy	Connectivity

Increased compute requirements

Increased software content

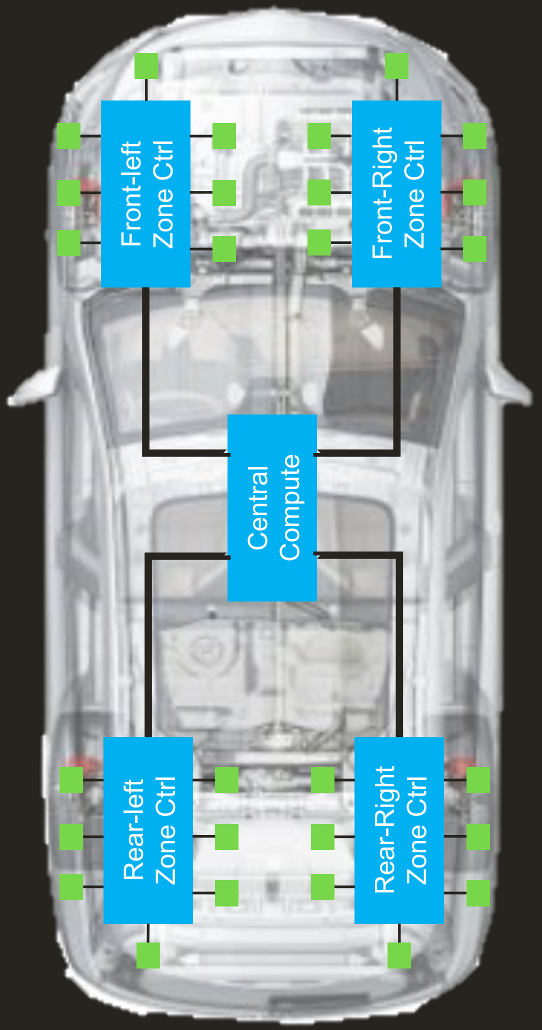
Safety remains paramount

Security becoming critical

Consolidation of compute resources

Drive for standardisation

Discrete ECUs → Domain Controllers → Zonal Controllers



Functional safety

Compliance with Safety Standards

Failure Avoidance and Detection

Risk Reduction and Hazard Analysis

System Reliability and Redundancy

FuSa ensures reliable and safe operation

Real-Time CPU

Deterministic Execution

Interrupt Handling

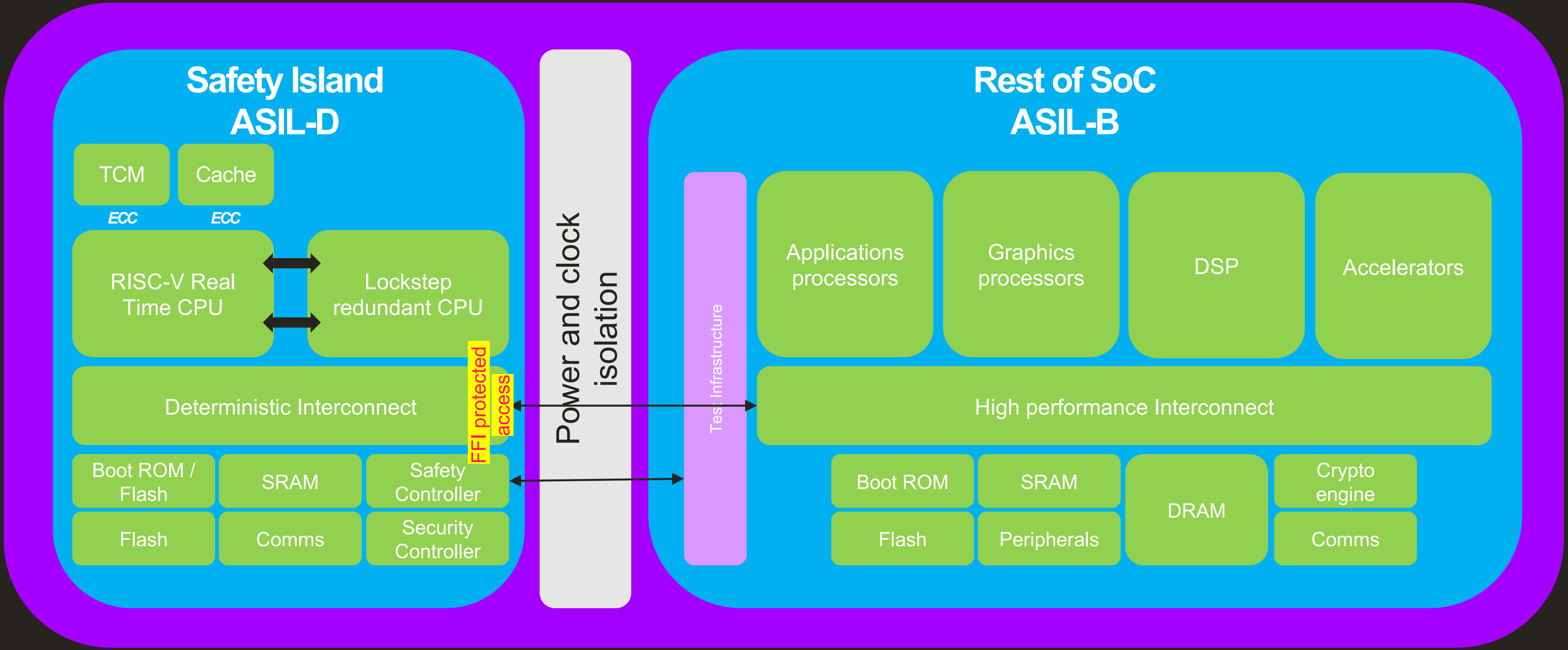
Task Scheduling

Fast Context Switching

Low Latency I/O

Predictable Performance

Example Safety Island SoC



Freedom From Interference requires that a failure in the 'Rest of SoC' (ASIL-B) must not be able to cause a failure in the Safety Island (ASIL-D)

- **Timing and execution**
  - Execution of an ASIL-B function being blocked must not block an ASIL-D function executing
  - Made easier as only ASIL-D functions run on the Safety Island
  - Safety Island code must not block waiting on an action from ASIL-B software
- **Memory**
  - Memory corrupted by faulty execution on the ASIL-B side must not affect Safety Island software
  - Generally, use separate memories with no access to the Safety Island memory from Rest Of SoC
  - Any shared buffers should be in a constrained area in the Safety Island side
    - If accessibility from Rest of SoC is programmable, must be configured by Safety Island software
- **Exchange of information**
  - Safety Island software must treat any data from the Rest-Of-SoC as unreliable (maybe in shared buffer)
  - Validate integrity, ensure corrupted data does not cause failure

The Safety Island

Characteristics

- Physically Isolated (power and clock) from Rest of SoC (to provide protection from common mode failures)
- Keep as simple as possible – less components, easier to analyse, less opportunity for failures
- Real time CPU (Typically TCMs and no MMU)

Functions

- General ASIL-D workloads
- Control reset and clocks for Rest of SoC
- Monitor the rest of the SoC for safety failures
- Provide resilient communication to other ECUs
- Coordinate in-service BIST
- Security monitoring

Summary

- Industry trends driving move to more compute, and much more software
- Architecture moving from separate ECUs, to Domain controllers, to Zonal /Centralised controllers
- Increased need to mix safety criticality on a single SoC
- Best achieved using a high-safety Island

THANK YOU