ARRL: A CRITERION FOR COMPOSITIONAL SAFETY AND SYSTEMS ENGINEERING

Eric Verhulst, Bernhard Sputh, Altreonic NV

Content

- Safety engineering and Safety Integrity Levels (SIL)
- Some issues with the SIL criterion
- Introducing the normative ARRL criterion
- Illustrated architectures
- Conclusions
- Note: Work In Progress!

Some background projects

- ASIL project
  - Project with Flanders Drive to develop a common "automotive" safety engineering methodology
  - IEC-61508, IEC-62061, **ISO-26262**, ISO-13849, ISO-25119 and ISO-15998. (+ CMMI, Automotive SPICE)
  - About 350 steps, 100 workproducts, ...
  - ASIL imported in GoedelWorks portal
- EU FP7 IP OPENCOSS
  - Project with 17 EU partners (avionics, railway, automotive) on reducing the cost and effort of certification
    - ISO-26262, DO-178C/254/..., CENELEC 50126-128-129
    - Cross-domain
    - Product families
- LinkedIn discussion groups (new: ARRL)
- => there is interest and a growing awareness

Systems Engineering vs. Safety Engineering

- System = holistic
- Real goal is **"Trustworthy Systems"**
  - Cfr. Felix Baumgartner almost did not do it because he didn't trust his safe jumpsuit
- TRUST = by the user or stakeholders
  - Achieving intended Functionality
  - Safety & Security & Usability & Privacy
  - Meeting non-functional objectives
    - Cost, energy, volume, maintainability, scalability, Manufacturability,..
- So why this focus on safety?
- User expects guaranteed "QoS" from a "Trustworthy system"

Safety and certification

- **Safety** can be defined to be the **control of *recognized hazards*** to achieve an ***acceptable level** of **risk***.
    - Safety is general property of a system, not 100% assured
    - It is complex but there are moral liabilities
- Certification: In depth review => safe to operate
    - "Conformity assessment" (for automotive)
    - Not a technical requirement: confidence, legal
- **Evidence makes the difference:**
    - Evidence is a **coherent** collection of **information** that relying on a number of **process artifacts** linked together by their **dependencies and sufficient structured arguments** provides an **acceptable proof** that a specific system goal has been reached.

Categorisation of Safety Risks

Category	**Consequence upon failure**	**Typical SIL**
Catastrophic	Loss of multiple lives	4
Critical	Loss of a single life	3
Marginal	Major injuries to one or more persons	2
Negliglible	Minor injuries at worst or material damage	1
No consequence	No damages, except user dissatisfaction	0

- SIL ≅ f (**probability of occurrence, severity, controllability**)
  - As determined by HARA
  - SIL goals ≅ Risk Reduction Factor
- Criteria and classification are open to interpretation

Safety as a goal across domains

Domain	Approximate mapping				
General (IEC-61508) Programmable electronics	(SIL0)	SIL1	SIL2	SIL3	SIL4
Automotive (26262)	ASIL-A	ASIL-B	ASIL-C	ASIL-D	-
Avionics (DO-178/254)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A
Railway (CENELEC 50126/128/129)	(SIL0)	SIL1	SIL2	SIL3	SIL4

Risk reduction factors depend on domain and usage pattern!

Detailed analysis reveals **only partial mapping!**

Problems with SIL definition

- Poor harmonization of definition across the different standards bodies which utilize SIL=> Reuse?
- Process-oriented metrics for derivation of SIL
- SIL level determines architecture (system specific)
- Estimation of SIL based on **reliability estimates**
  - System complexity, particularly in software systems, makes SIL estimation difficult if not impossible
  - based on probabilities that are very hard if not impossible to measure and estimate
  - Reliability of software (discrete domain) is not statistical!:
  - The **law of Murphy still applies**:
    - The next instant can be catastrophic

Composibility in the safety domain

- Although this is the practice in systems engineering, it is poorly addressed in the standards
- ISO-DIS-25119 (& ISO 13849): rule of composition for safety critical subsystems: weakest link gives PL (SIL)
- ISO-26262 – **SEooC**: qualification in isolation by defining boundary conditions of use
- Avionics – **IMA**: reuse is promoted by defining a common architecture
- => the principle of reuse in the safety domain exists, but still **weakly formalised**

New definition: we start from the component

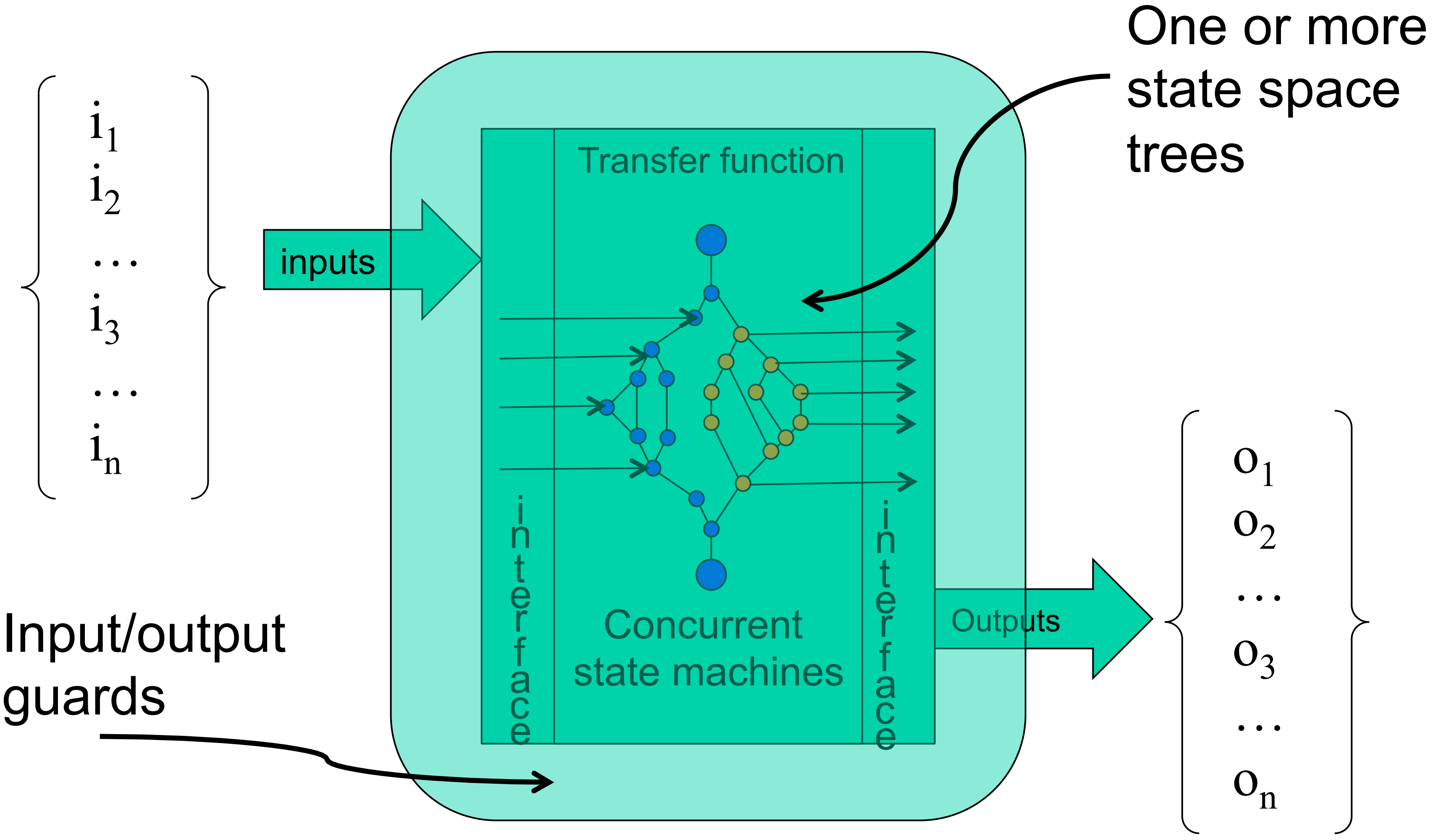
- **ARRL: Assured Reliability and Resilience Level**

ARRL 0	it might work (use as is)
ARRL 1	works as tested, but no guarantee
ARRL 2	works correctly, IF no fault occurs, guaranteed no errors in implementation) => formal evidence
ARRL 3	ARRL 2 + goes to fail-safe or reduced operational mode upon fault (requires monitoring + redundancy) - fault behavior is predictable as well as next state
ARRL 4	ARRL 3 + tolerates one major failure and is fault tolerant (fault behavior predictable and transparent for the external world). Transient faults are masked out

ARRL: what does it mean?

- Assured:
  - There is verified, trustworthy **evidence**
  - Process related and architecture related
- Reliability:
  - In absence of faults, MTBF is >> life-time: **QA aspects**
- Resilience:
  - The fault behaviour is predicted: **trustworthy behaviour**
  - Capability to continue to provide core function
- Level: ARRL is normative
  - Components can be classified: **contract**

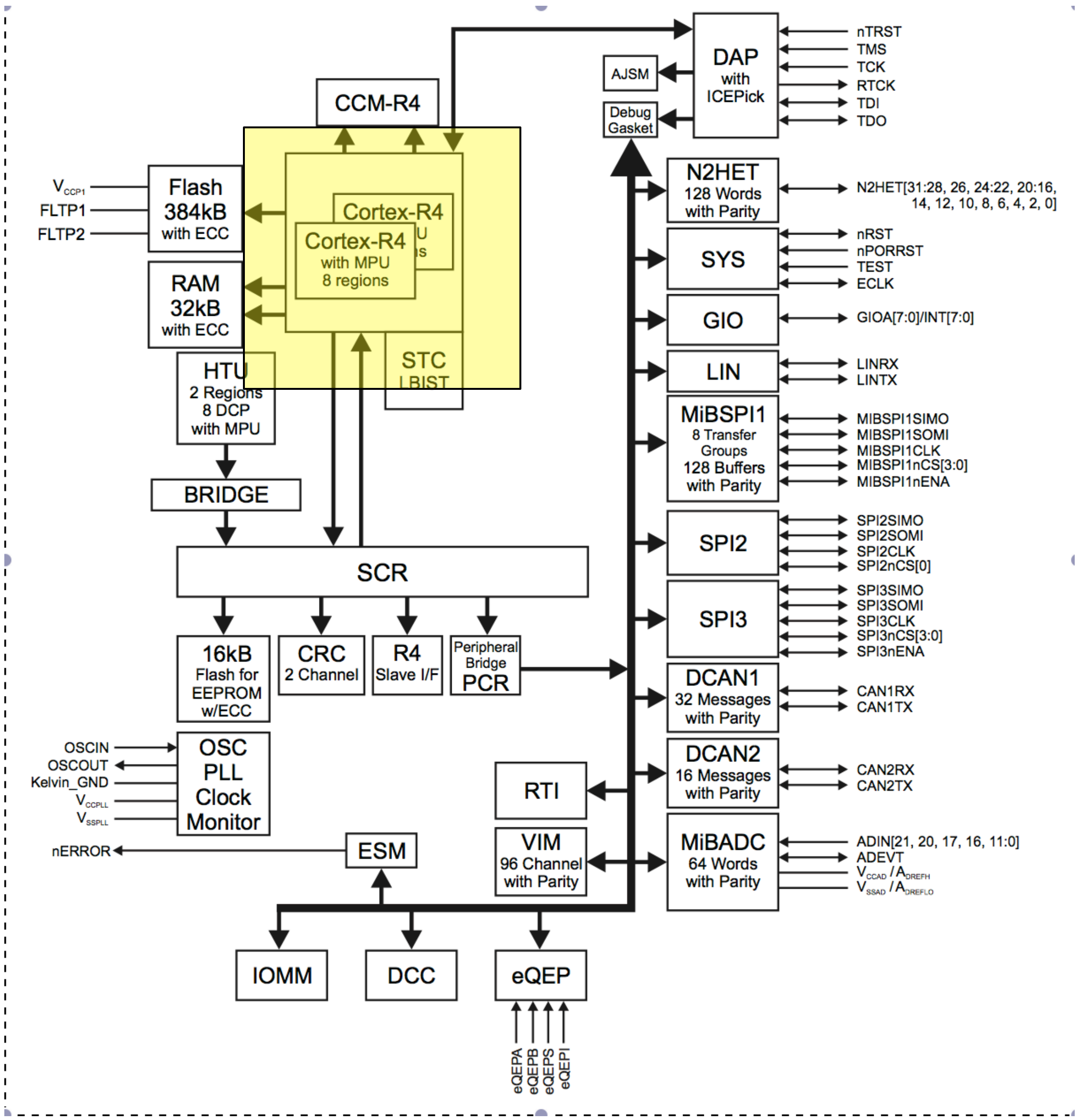
Architectural component view (discrete domain


One or more
state space
trees
$\left\{ \dot{i}_1, \dot{i}_2, \ldots, \dot{i}_3, \ldots, \dot{i}_n \right\}$
inputs
Transfer function
interface
interface
Concurrent
state machines
Outputs
$\left\{ o_1, o_2, \ldots, o_3, \ldots, o_n \right\}$
Input/output
guards

Consequences

- If a system/component has a fault, it drops into a degraded mode => lower ARRL
  - ARRL3 is the operational mode after an ARRL4 failure
    - Functionality is preserved
    - Assurance level is lowered
- SIL not affected and domain independent
  - System + environment + operator defines SIL
- ARRL is a **normative criterion**:
  - Fault behavior is made explicit: verifiable
  - Cfr. IP-norm (comes with a predefined test procedure)

Example SoC to guide the ARRL qualification



Vendor claims:

High-Performance Microcontroller for Safety-Critical Applications

– Dual CPUs Running in Lockstep

– ECC on Flash and RAM Interfaces

- MPU

– Built-In Self-Test for CPU and On-Chip RAMs – Error Signaling Module with Error Pin

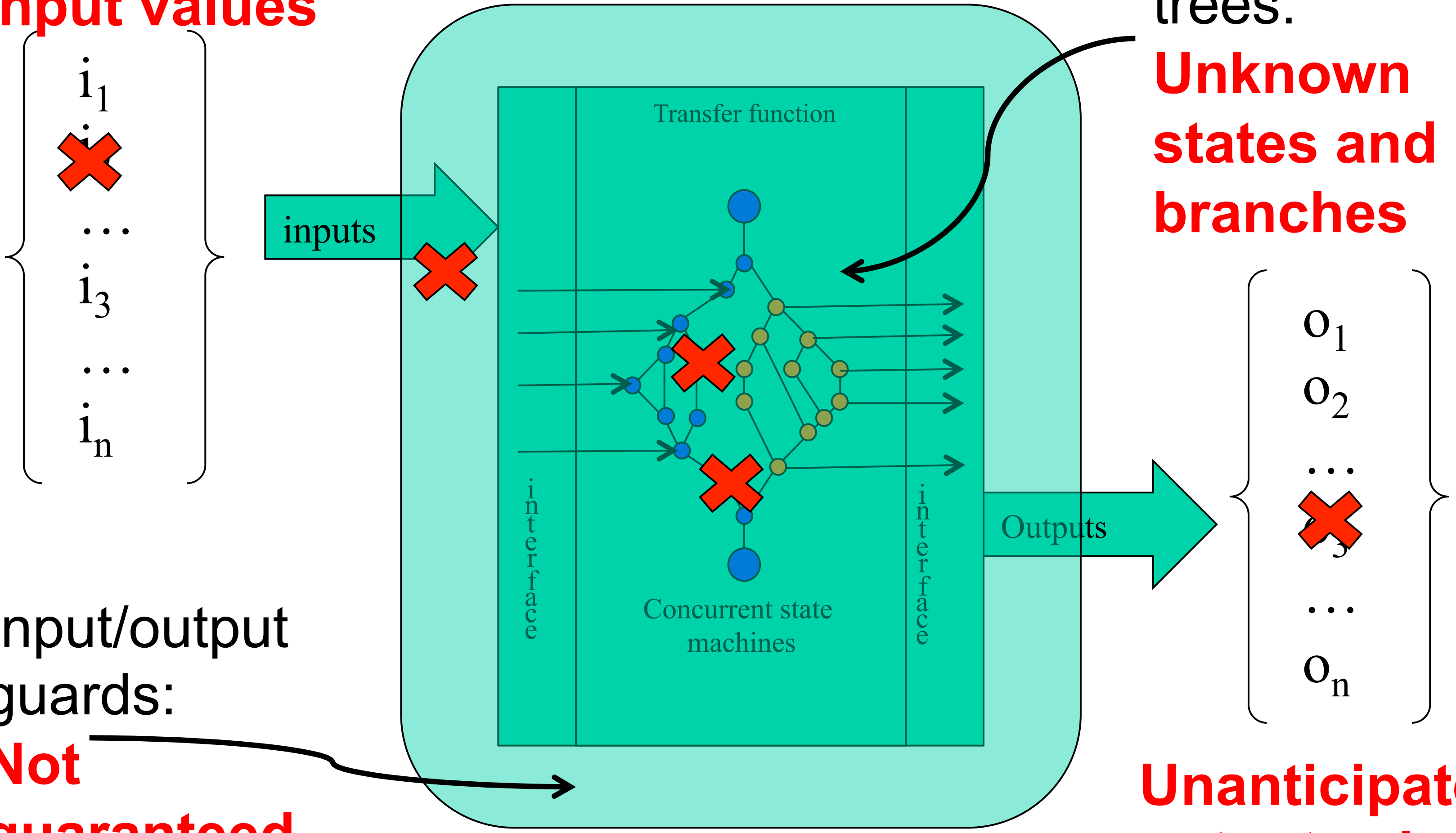
– Voltage and Clock Monitoring

What ARRL level?

ARRL-0/1

- ARRL-0: “use as is”
  - No verified contract: no assurance
  - Still needs a specification
  - Assumes QA at production
- ARRL-1: ARRL-0 + “works as tested”
  - Scope of assurance limited to test cases
  - Evidence = verified test reports
  - Absence of errors not assured
- Hence not really usable for safety critical systems

ARRL-1


Unanticipated input values
i_1
i_2
…
i_3
…
i_n
inputs
Transfer function
interface
interface
Concurrent state machines
Outputs
o_1
o_2
…
o_3
…
o_n
One or more state space trees:
Unknown states and branches
Input/output guards:
Not guaranteed
Unanticipated output values

Gaps/Risks due to erroneous specifications and incomplete testing

ARRL-2

- ARRL-2: ARRL-1 + formal evidence for all specified properties (if no fault): logically correct
- Hardware:
  - Design verification
  - Extensive testing, burn-in, etc.
- Software:
  - Formal evidence:
    - Use of FM, proven in use, …
- Process requirements:
  - Rigorous development, verification, validation, review, …
  - Stress testing to confirm corner cases are handled

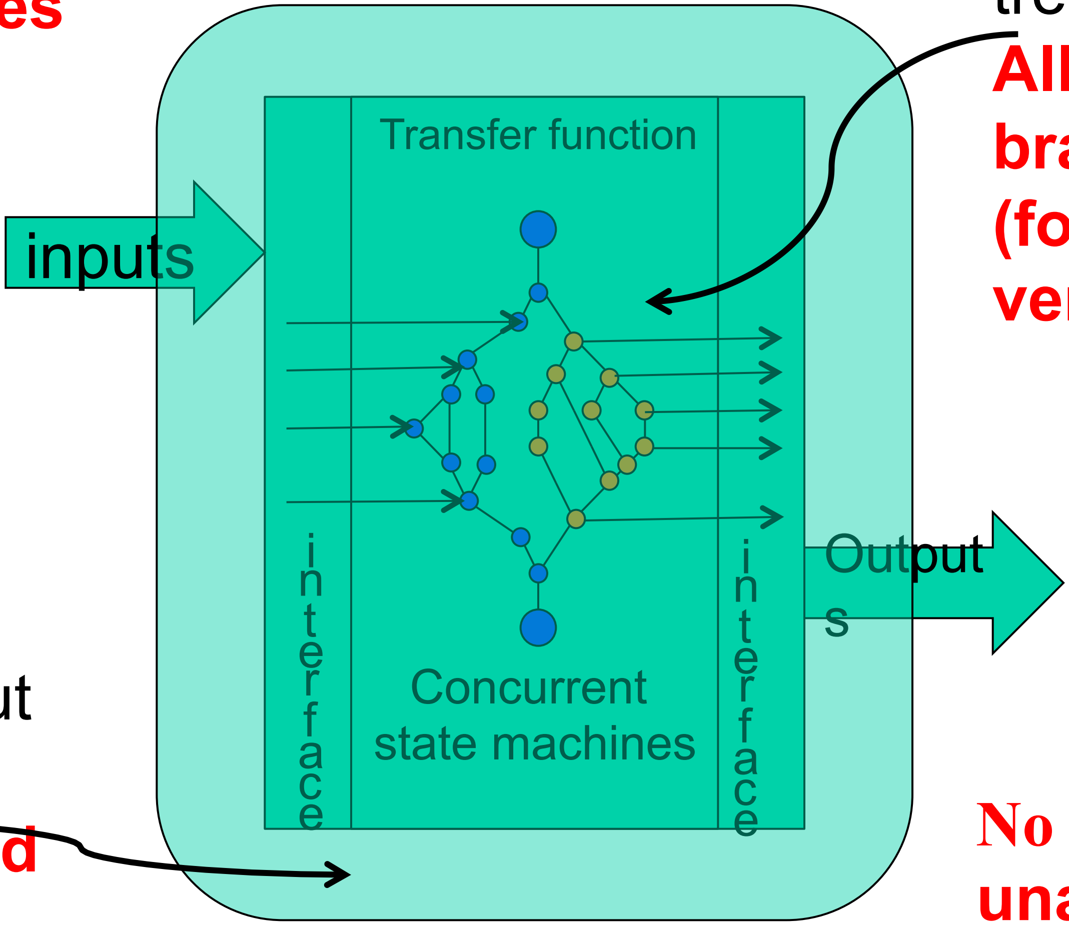
ARRL-2

No unanticipated input values

$$\left\{\begin{array}{c} i_1 \\ i_2 \\ \ldots \\ i_3 \\ \ldots \\ i_n \end{array}\right\}$$

One or more state space trees:

All states and branches (formally) verified



$$\left\{\begin{array}{c} o_1 \\ o_2 \\ \ldots \\ o_3 \\ \ldots \\ o_n \end{array}\right\}$$

Input/output guards:

guaranteed

No unanticipated output values

Normal Case specifications correct, implementation logically correct

Role of Formal techniques

- Formal evidence is wide:
  - Use of formal models at design time
  - Use of formal verification post implementation
  - Evidence of rigorous process
    - Document - Test – Verify – Validate – Review – Confirm - …
  - Proven in use (weaker argument)
  - Stress testing (weaker argument)
- Formal methods increase confidence

ARRL-3

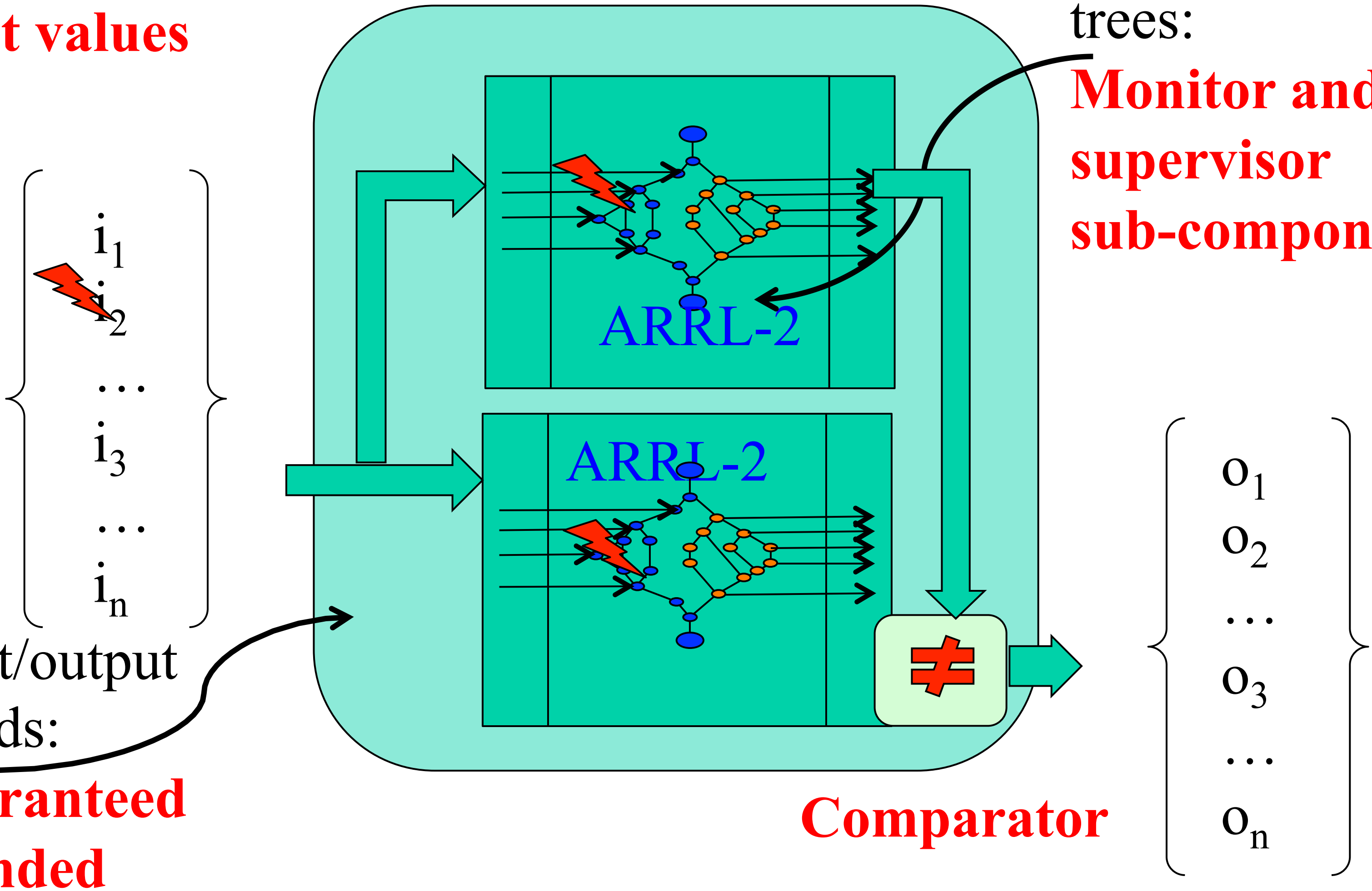
- ARRL-3: ARRL-2 + fail-safe mode upon fault
- All possible fault cases are part of specification
- Fault behavior predictable upon fault
- Fault: at micro-level (bit level state)
- Features:
  - Monitoring and redundancy for degraded mode
  - Prevent error propagation, incl. externally
  - Isolate fault area internally
  - Easier with modular architecture
  - Keeps correct functionality if possible
  - HW/SW co-design

ARRL-3

Unanticipated input values

Induced fault

One or more state space trees: **Monitor and supervisor sub-component**



Input/output guards: **Guaranteed bounded**

Comparator

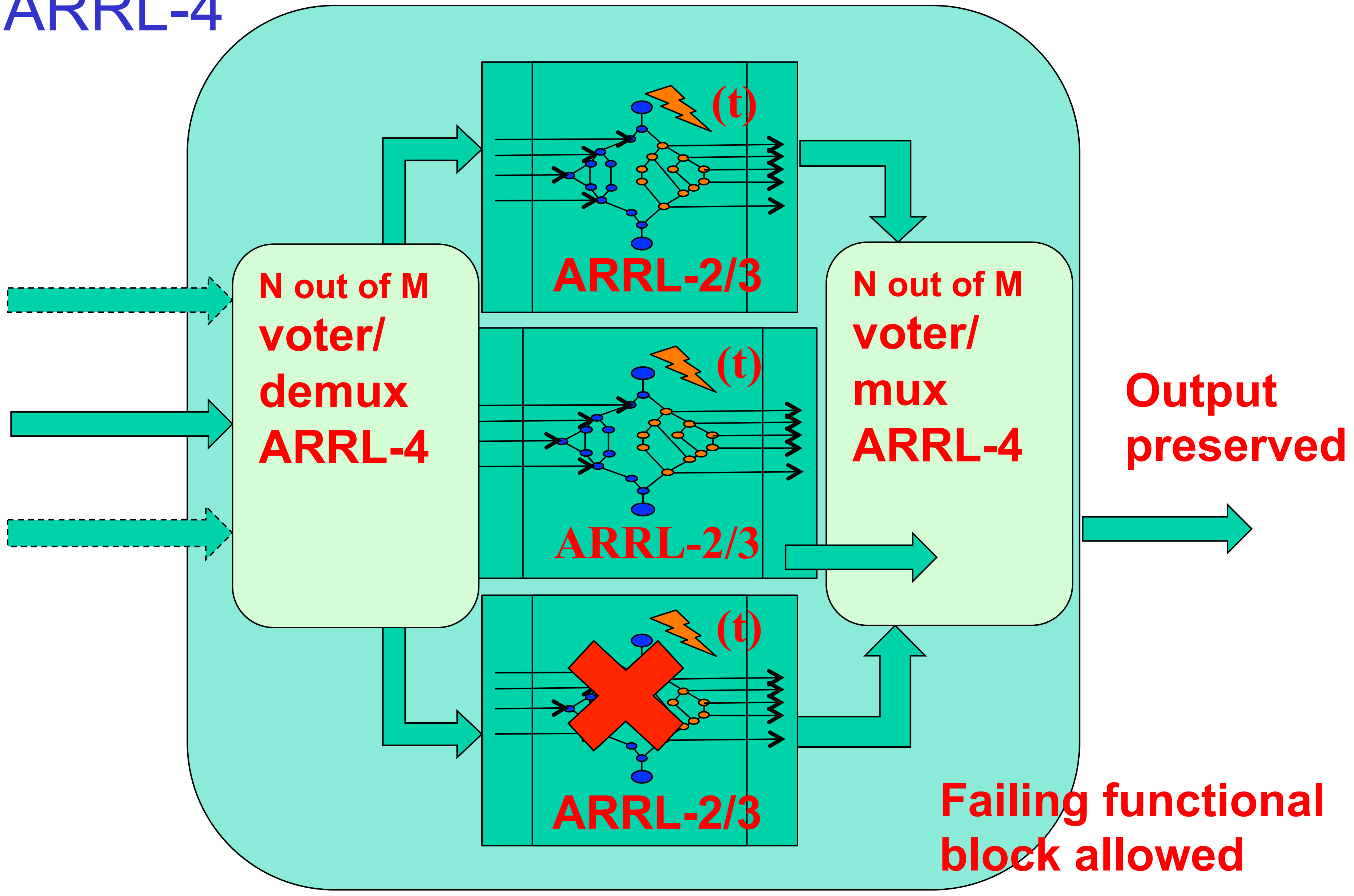
Common mode failures possible

Fail safe output

ARRL-4

- ARRL4: ARRL-3 + fault tolerance
- Fault: at macro-level (functional block)
    - What is the unit of failure?
- Requires macro-level redundancy + voting
- Interconnect needs to be ARRL-4 as well

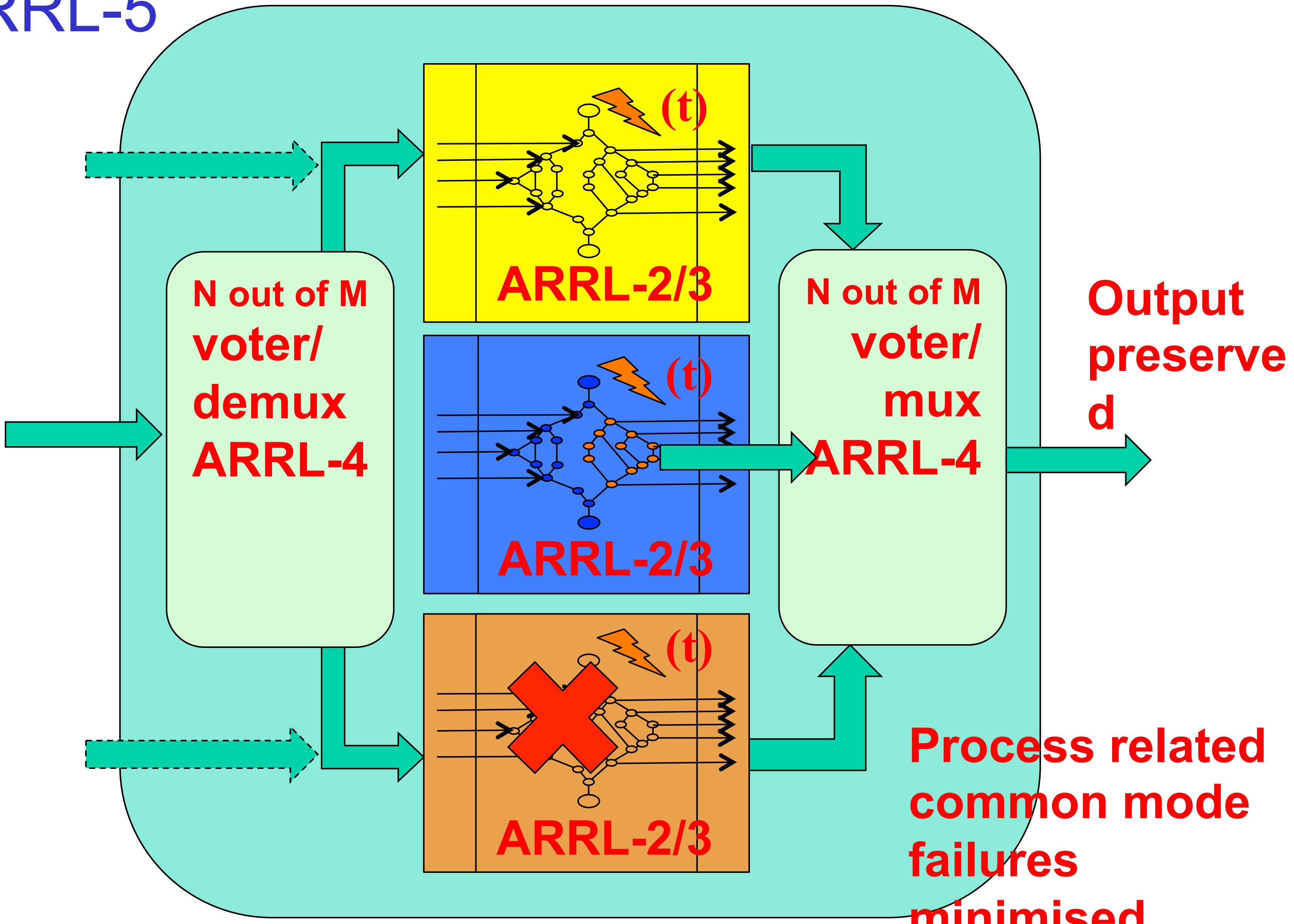
ARRL-4


N out of M
voter/
demux
ARRL-4
(t)
ARRL-2/3
(t)
ARRL-2/3
(t)
ARRL-2/3
N out of M
voter/
mux
ARRL-4
Output
preserved
Failing functional
block allowed

Residual common mode failures => **ARRL-5**

- ARRL-4 assumes independence of faults in each redundant channel
- Covers only a subset of the common mode failures
- Often residual ones are process related
- Less visible are e.g. common misunderstanding of requirements, translation tool errors, time dependent faults => require asynchronous operation and diversity/heterogenous solutions
- Hence we can define an ARRL-5 as well

ARRL-5



Process related common mode failures minimised

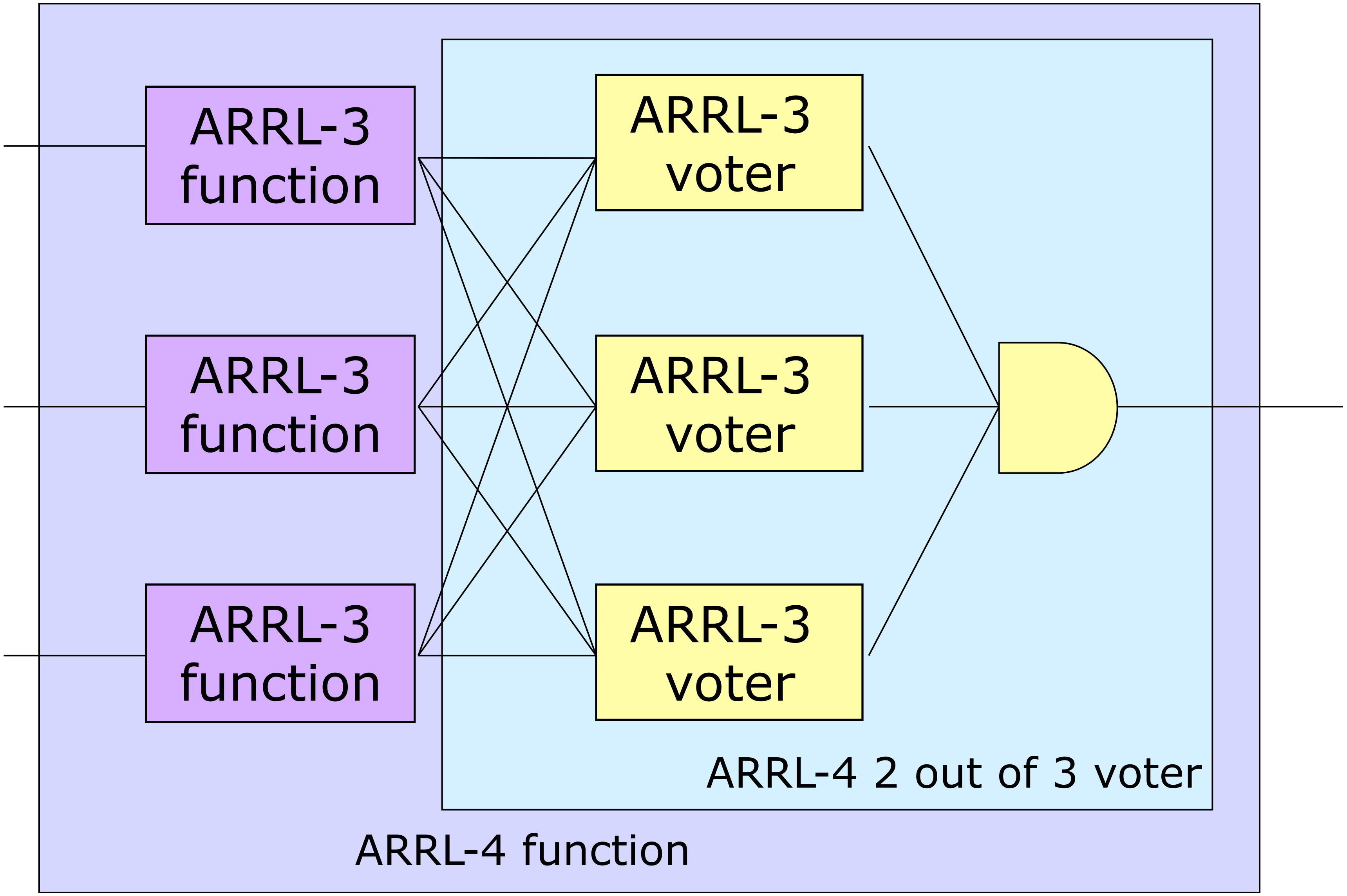
ARRL-5

- ARRL5: ARRL-4 + design diversity
- Focus is on common mode failure at design level
- Requires rigorous interface specification
- Best use asynchronous interactions
- Can still affect real-time capabilities

Composition rule:

- **A system can only reach a certain SIL level if all it components are at least of the same ARRL level.**
  - This is a necessary condition, not a sufficient condition
  - Redundancy can compose ARRL 4 components out of ARRL 3 components (needs an ARRL 4 voter)
  - ARRL3 component can use ARRL 2 components (>2)
  - In line with architectural recommendations based on SIL levels
- Consequences:
  - Interfaces and interactions also need ARRL level!
  - Error propagation is to be prevented => partitioning architecture (e.g. distributed, concurrent)
  - Using ARRL-3/4 components means that the system becomes resilient: run-away situations leading to critical states are contained.

Generic example



So what about the lock stepped SoC?

CPU +memory = ARRL-3, but

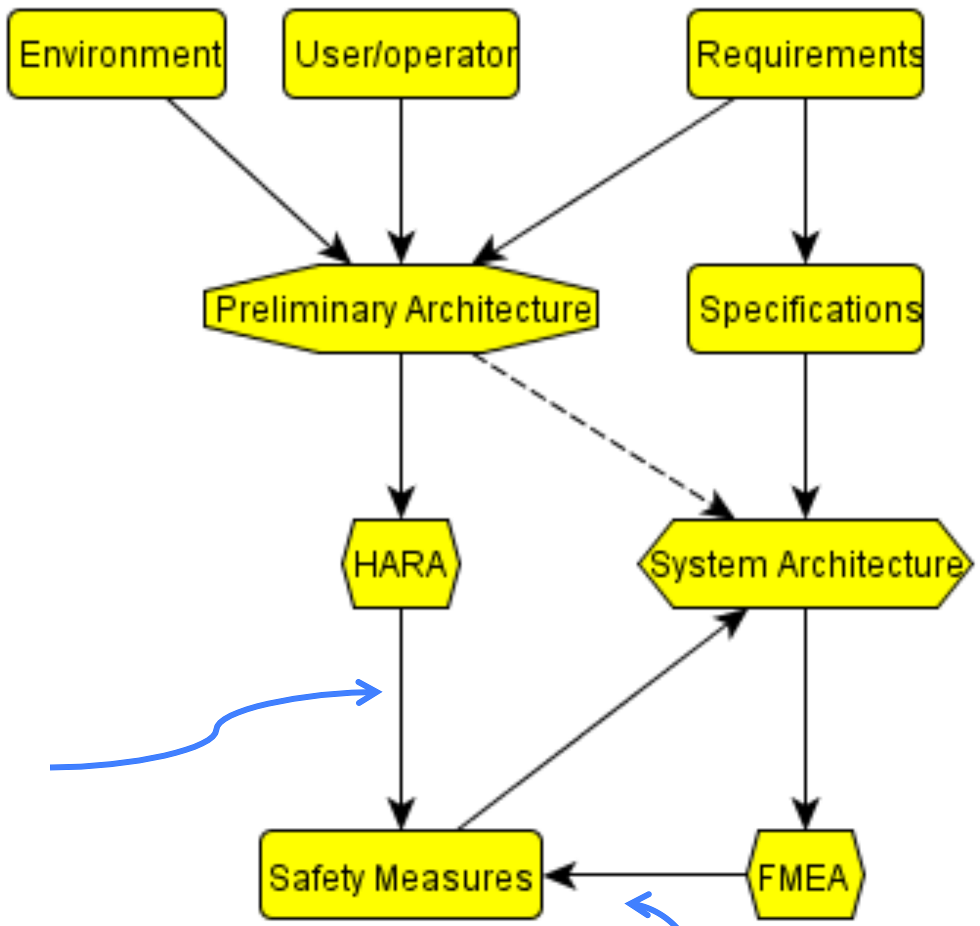
Multiple Communication Interfaces

- **Two CAN Controllers (DCANs)**
  - **DCAN1 - 32 Mailboxes with Parity Protection**
  - **DCAN2 - 16 Mailboxes with Parity Protection**
- **Multibuffered SerialMibSPI Module**
  - **28 Words with Parity Protection**
- **Two Standard SPI Modules**
- **UART (SCI) Interface with LIN 2.1**
- **High-End Timer (N2HET) Module**
  - **19 Programmable Pins (internal microsequencer)**
  - **128-Word Instruction RAM with Parity Protection**
  - **Each Includes Hardware Angle Generator**
  - **Dedicated High-End Timer Transfer Unit**

Overall, quite good (better than most)

- **Some peripherals are only ARRL-1/2**
- **If used => ARRL-1/2 for full SoC**
- **Mitigation needed in SW or at system level**

SIL and ARRL are complementary



SIL driven

ARRL driven

Conclusions

- Unified system and safety engineering is feasible
- Unified safety certification is not yet feasible (standards and SIL differ too much)
- ARRL concept allows compositional safety engineering with reuse of components/subsystems
- More complex systems can be safer
- A unified ARRL aware process pattern can unify systems and safety engineering standards

More info:

www.altreonic.com

White paper as work in progress available

Further work

- Making ARRL normative and applicable
  - Refinement and Completeness of criteria
  - Normative: components carry contract and evidence
    - Independent of final use or application domain
    - Process evidence + validated properties
    - ARRL-3 and higher: HW/SW co-design?
  - Study link with a system's critical states
  - Apply it on real cases:
    - OpenComRTOS (formally developed)
    - ARRL-awareness for projects developed in GoedelWorks
- Input and feedback welcome