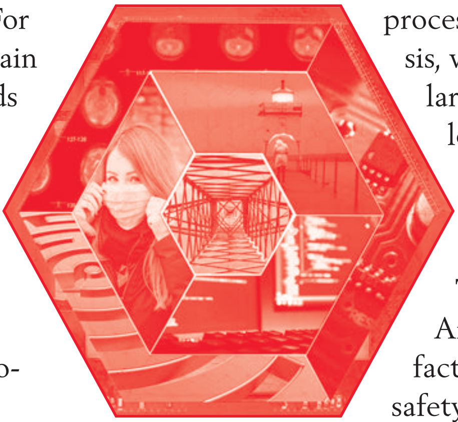
Applying Safety Concepts and Principles in Vital Controller Design

by Fenggang Shi
Toronto, Canada

A modern large-scale safety-critical system relies on multiple controllers to control devices and collaborate with each other to deliver expected functions. Some of these controllers take safety responsibilities to ensure safe system operation in the given environment. For example, a Communications-based Train Control (CBTC) system uses hundreds of computers to automate metro rail transportation and ensure it is operating safety. The safety-critical controllers are in two main classes:



- Wayside controllers for controlling railway trackside devices, providing movement authority for a train and enforcing signaling interlocking functions.
- Onboard controllers for enforcing train movement authority and safe speed limits, as well as correct train door operations.

If any of these controllers fails on delivering a safety-critical function, a possible hazard can be raised, leading to an accident with unacceptable consequences. Thus, these controllers must be designed not only to have correct reactions to hazardous conditions raised from the operating environment (such as railway switch failures), but also to be fail-safe in the operating context against their own internal failures.

Designing and upgrading safety-critical controllers is the unending endeavor of a safety-critical system supplier because customers require the most advanced electronics to be up to date on their projects or the components used in the designs of existing controllers will approach end of life. A safety-critical controller must be designed to meet the expected vitality (i.e., fail-safe) regarding detecting and mitigating all possible failures in advanced electronics. This safety objective raises challenges because electronics technology is constantly advancing and the corresponding complexity is increasing at a much faster pace than engineers' knowledge. Traditional safety engineering approaches are not efficient and effective to provide guidance in designing a fail-safe controller against controller internal failures at the early development stage. Popular safety engineering processes employ a top-down hazard analysis, which takes a significant effort and a large amount of time from the system-level hazard analysis down to the low level of controller internal hardware failure effects for identifying hazardous failure conditions of a controller. Traditional Failure Modes and Effects Analysis (FMEA) is performed after the fact at the hardware design level. Thus, safety engineers may not be able to provide valuable input in time for designers to include effective failure detection and mitigation mechanisms. In reality, it is quite frequent to find new hazardous failures after design or during later system operation, which raises significant safety concerns leading to major design modifications at high costs.

This paper proposes an approach to using a set of safety concepts as guidance for both safety-critical controller design and its safety integrity assessment, based on mitigating the generic hazard that the controller's internal failures cause unexpected outputs. The design objective is to use these safety concepts in the most effective way to achieve the least complex safety-critical controller that meets the expected highest safety integrity level. These safety concepts are categorized as *intrinsic fail-safe*, *reactive fail-safe* and *composite fail-safe* [Refs. 1 and 2]. We have practiced an effective combination of these concepts in our CBTC projects. The composite fail-safe concept in checked redundancy (a.k.a. check redundancy in some publications) techniques is used to design the architecture of a controller and to enforce the workcycle-based lockstep voting of safety-critical parameters for mitigating failures in one channel in time. The reactive safety concept in self testing and closed-loop monitoring mechanisms are used in each of the checked redundant channels for revealing dormant failures that may not show signal-level effects to the checked redundancy voting. Finally, the intrinsic fail-safe concept is used to design safe interfaces to other controllers and controlled devices. Based on our experience, a combination of these safety concepts and their application principles serves to achieve a high safety integrity level of a controller through design.

Application of Safety Design Concepts in Controller Designs

In our experience of designing vital controllers for new CBTC systems or upgrading existing CBTC products, traditional safety engineering approaches are not effective to provide guidance for safety-critical designs in time, and may result in unmitigated failures in controllers at the design stage. In our original practice, designing a safety-critical controller was guided by a systematic safety engineering program following a top-down hazard analysis to identify safety requirements for the design. The system safety engineering program enforced preliminary hazard analysis, system hazard analysis, the subsystem hazard analysis and the later hardware failure mode analysis to identify hazardous failures in a controller, after which their mitigations were specified as the safety requirements for the controller design. In this traditional approach, there are two typical problems:

- Hazards and their mitigations identified from different subsystem analyses, even though they are based on the same controller design, are normally quite different in appearance and may not be consistent and complete for covering all possible internal random failures of a controller.
- FMEA of component-level failures of a controller may not identify all hazardous failures due to unmanageable complexity of the modern electronics of intelligent processors (having numerous failure modes) and incorrect judgment of some failure effects (based on the single failure viewpoint).

Therefore, the new approach of using safety concepts as the design guideline is introduced, which has been practiced in multiple CBTC system projects and shows more effectiveness than traditional safety engineering approaches [Ref. 3]. The combination of these safety concepts focuses on generic controller internal hazards associated with hardware failures — which is that any hardware failure mode leads to unintended permissive outputs. This generic view of controller internal hazardous failures leads us to identify safety design concepts and their application principles as sound guidelines for designers and safety engineers to determine failure detection techniques to be designed into safety-critical controllers.

The safety concepts used in the CBTC designs are categorized as intrinsic fail-safe, reactive fail-safe and composite fail-safe. By applying them in an effective combination, a controller can be designed with mechanisms for detecting internal failures in time, and then enforcing safe states as the corresponding reactions to them [Ref. 4]. At an abstract level, these fail-safe concepts are viewed as the design philosophy. Each of them has application principles that are incorporated into the design of a controller so that, in the event of a failure, the controller detects the failure in time and enters (or remains in) a safe state. These fail-safe concepts are further clarified here:

- **Composite fail-safety** — With this concept, each safety-related function is performed by at least two items. Each of these items shall be independent from all others to avoid common-cause failures. Non-restrictive (or permissive) activities are allowed to progress only if the necessary number of items agree. A hazardous fault in one item shall be detected and negated in sufficient time to avoid a co-incident fault in a second item.
- **Reactive fail-safety** — This technique allows a safety-related function to be performed by a single item, provided its safe operation is assured by rapid detection and negation of any hazardous fault (for example, by encoding, by multiple computation and comparison or by continual testing). Although only one item performs the actual safety-related function, the checking/testing/detection function shall be regarded as a second item, which shall be independent to avoid common-cause failures.
- **Intrinsic fail-safety** — This technique allows a safety-related function to be performed by a single item, provided all the credible failure modes of the item are non-hazardous. Any failure mode that is claimed to be non-credible shall be justified according to the intrinsic physical properties of used components and the application environ-

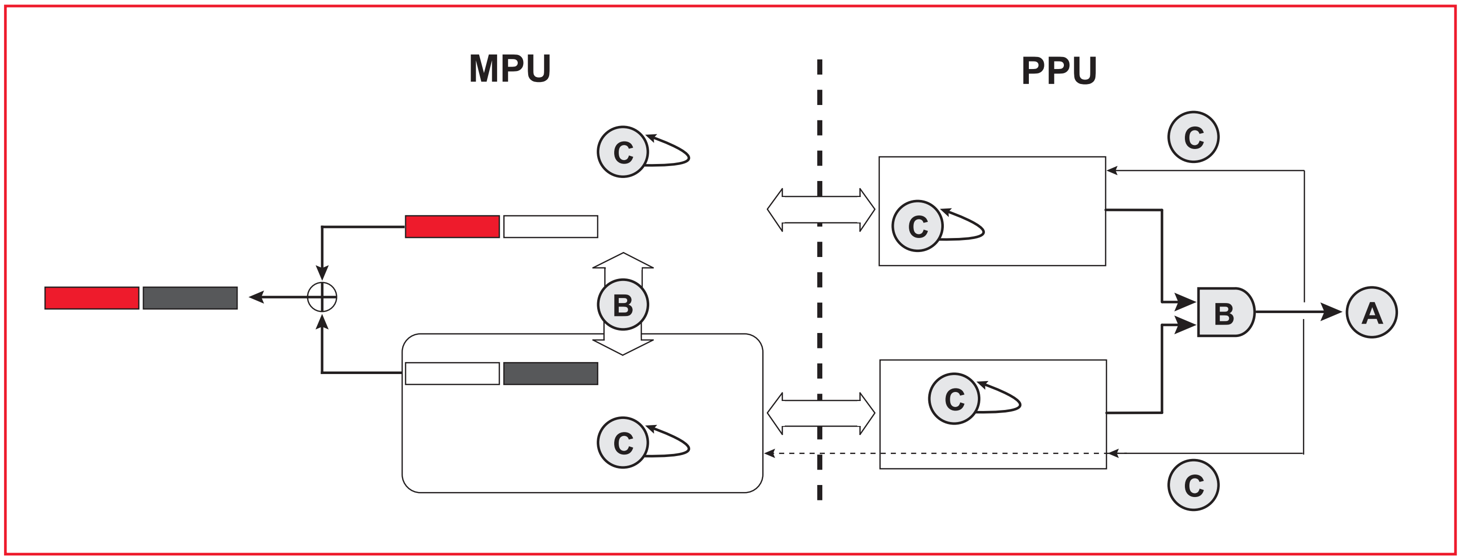


Figure 1 — Controller in 2oo2 Checked Redundancy.

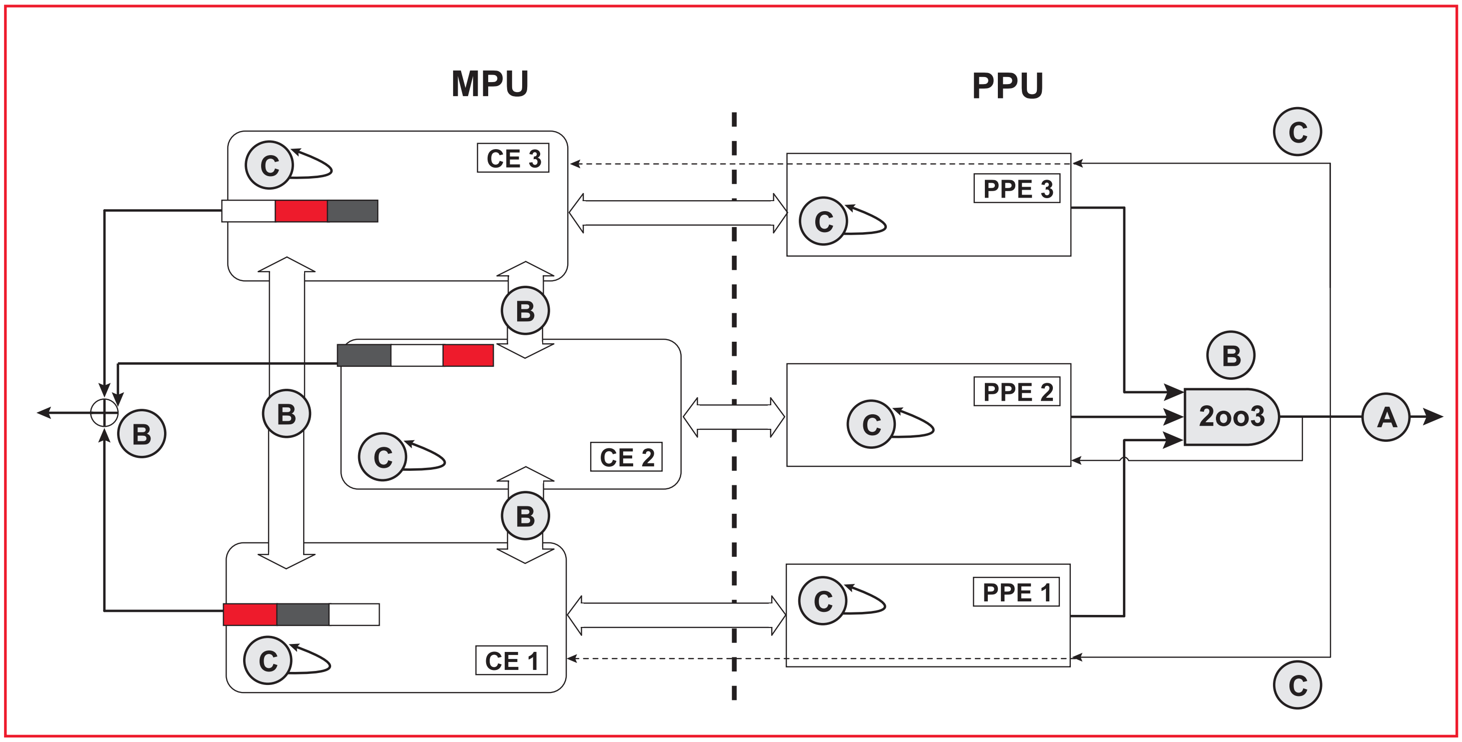


Figure 2 — Controller in 2oo3 Checked Redundancy.

ment. Intrinsic fail-safety may also be used for certain functions within composite and reactive fail-safe systems — for example, to ensure independence between items or to enforce shutdown if a hazardous fault is detected.

In the current CBTC designs, which have been implemented in multiple projects, safety-critical controllers using the determined set of safety concepts have the following four categories from an architectural viewpoint:

- Standalone Main Processor Unit (MPU) in 2oo2 (two out of two) without Periphery Processor Unit (PPU), as shown in Figure 1
- MPU in 2oo2 with its PPU (Peripheral Processor Unit, in 2oo2 voting), as shown in Figure 1
- Standalone MPU in 2oo3 (two out of three voting) without PPU, as shown in Figure 2
- MPU in 2oo3 (two out of three voting) with its PPU, as shown in Figure 2

In each diagram as shown in Figures 1 and 2, the safety concepts and the techniques for implementing them are indicated specifically by:

- "A" indicating the intrinsic fail-safe interfaces by using intrinsic fail-safe components;
- "B" indicating the composite fail-safe by using the checked redundancy techniques;
- "C" indicating the reactive fail-safe by using self-checking and self-testing techniques;
- Colored small red or gray rectangles indicating a part of a vital communication telegram. Any single computer in checked redundancy shall only contribute a part of a telegram for sending to other vital controllers (to prevent a single channel sending a complete telegram to other controllers without voting).

Each vital controller has its MPU designed by following the composite fail-safe concept in either the 2oo2 or the 2oo3 checked redundancy architecture. Also, a vital subsystem can have one or more PPUs, and each PPU is incorporated into the checked redundancy channels of MPU. Each checked redundancy channel of MPU is named as a Computing Element (CE), each performing the same safety-related function whereby each computing element must be in agreement to allow permissive activity to continue. Each PPU channel is named as a Peripheral Processing Element (PPE), which allows the associated CE to communicate with various external interfaces.

To ensure correctness in applying these safety concepts and maximizing the safety integrity of controllers with an effective combination of the concepts in the identified techniques, the principles, assumptions and effectiveness factors of each safety concept will be discussed in subsequent sections.

Principles of Composite Fail-Safety in Checked Redundancy

The checked redundancy concept requires the use of multiple (normally two or three) independent hardware units executing control logic, performing identical functions, and voting based on 2oo2 or 2oo3 logic to vote on certain internal safety parameters, inputs and outputs. Also, voting requires that these independent computation and control channels must periodically obtain agreement to assert any permissive output. If the units do not agree, safety-critical functions and outputs default to a known safe state.

Checked redundancy is an important technique to implement the composite fail-safe concept for detecting failures in complex electronic circuits and processors to support composite fail-safety. The premise of the concept is that if the channels in the checked redundancy configuration are independent, failures will not occur on more than one channel before being detected. This relies on the checking period being short enough to detect a failure, or the effects of a failure, before it results in an unsafe condition. It is important to notice that checked redundancy cannot detect common mode failures in the channels. Also, checked redundancy has the potential risk of latent failures between checking intervals. Therefore, certain latent detection mechanisms (i.e., the reactive safety concept application) in each channel must be designed in. They are responsible for detecting failures that do not make themselves evident at the points of comparison.

Critical Assumption

A critical assumption associated with this concept is independency of the parallel channels implementing the checked redundancy. This means that the hardware failures in each of the redundant units in the parallel channels that produce the same erroneous unsafe effect will not occur simultaneously between voting time points, that is, within the interval between correspondence checks.

Specifically, common mode failures in parallel checked redundant channels are the most significant factor that may defeat the safety properties assumed by the composite fail-safe concept. It is useful to note that using asymmetric channels in which diverse hardware and software are implemented in different checked redundant channels may not be the best way to achieve the desired safety confidence. Such diversity techniques largely increase the system complexity, which itself most likely defeats the safety confidence and causes a project to fail from either financial limitations or schedule constraints. Therefore, common mode failures should be prevented in a practical and cost-effective way through a sound verification process.

Checked Redundancy Design Principles

The composite fail-safe concept may be implemented in either symmetric channels or asymmetric channels (such as a computation channel and a monitoring channel). The following principles are common checking mechanisms:

1. The 2oo2 (2 out of 2) checked redundancy logic is subject to vitality of the voter/checker. In particular, the design shall guarantee that:
   - The checking process is, in itself, fail safe;
   - The checking process is sufficiently frequent that similar or identical errors or failures in redundant units cannot occur between checks;
   - The checking process is sufficiently sensitive to detect all significant errors in a single unit;
   - A failure to check as scheduled causes timely action to occur, which maintains safety;
   - Redundant units are sufficiently independent that occurrence of hazardous failure with common modes must be remote during the safety check interval.
2. The 2oo3 (2 out of 3) checked-redundancy in terms of majority voting ensures that each of the three channels (including processors) is isolated from the others and is independently capable of either enforcing a safe reaction or being masked out by other functioning channels. The design shall guarantee that:
   - The checking process is, in itself, fail safe;
   - A failure in one of the three channels shall be detected in time, resulting in the 2oo2 checked-redundant logic in the subsequent operation;
   - The voting mechanism must be monitored and periodically tested by the processors of the three channels, and that a failure in the voting mechanism shall lead the whole controller to shut down;
   - A common mode failure in more than one channel must have remote probability between 2oo3 voting checks.
3. Each checked redundancy channel must have a mechanism to detect its internal integrity failure by following the principles in the reactive fail-safe concept.
4. The practical principle for mitigating common mode failures is:
   - To enhance the techniques and mechanisms to ensure independency and isolation between checked redundant channels, so that there are no physical internal influences between them from the viewpoint of electrical and electronics circuits;
   - To enforce an effective development process for preventing hazardous (systematic) defects in the hardware and software;
   - To enforce sufficient qualification of the design and use reliable components against disturbance from the operating environment such as temperature, moisture, and Electromagnetic Compatibility (EMC)/Electromagnetic Interference (EMI) effects;
   - To prevent physical external influences from the interfaces to the checked redundant channels resulting in a common mode failure.

"Checked redundancy is an important technique to implement the composite fail-safe concept for detecting failures in complex electronic circuits and processors to support composite fail-safety. The premise of the concept is that if the channels in the checked redundancy configuration are independent, failures will not occur on more than one channel before being detected."

Checked Redundancy Design Verification

The checked redundancy design must be analyzed to verify the independency of one channel from the others, and tested for the correctness of implementing the associated principles. The qualification tests must confirm that environmental conditions have been met to prevent any environmental common factor from raising a possible common mode failure in the checked redundancy channels beyond the tolerable risk level.

The safety verification must justify the residual risk in the checked redundancy design for satisfying the quantitative safety target in the context of the checking/voting mechanisms and the application environment. The MPU has two or three Computing Elements (CEs), which are organized in a 2oo2 or 2oo3 configuration, and the checked redundant principles are applied by the combination of hardware and software. If extended I/O capability is needed, one or more PPUs are added. Each PPU channel PPE-x is only specifically connected to the corresponding CE-x. The residual hazard in the checked redundancy (i.e., either 2oo2 or 2oo3) is the condition that two channels encounter common mode failures, defeating the voting (for generating correct voted results) and resulting in either unexpected permissive outputs or incorrect parameter values for subsequent computations. The residual hazard risk level can be estimated based on the worst case by using the whole failure rate λ-c-i of any checked redundancy channel (c-i) and the total number of any possible combinations of two channels that encounter failures during the lockstep voting interval.

λ-c-i = λ-ce-i + λ-ppu1-ppe-i, if only one PPU is used; or

λ-c-i = λ-ce-i + (λ-ppu1-ppe-i + λ-ppu2-ppe-i + …), if more than one PPU is used

Here,

- λ-c-i is the total failure rate of one complete check redundant channel;
- λ-ce-i is the whole failure rate of one CE;
- λ-ppu1-ppe-i is the failure rate of one PPE in PPU1; λ-ppu2-ppe-i is the failure rate of one PPE in PPU2…

Now, during the lockstep voting interval T, two channels have failures (defeating the voting) with the probability of (T × λ-c-i) × (T × λ-c-j), I ≠ j.

The residual hazard rate is ((T × λ-c-i) × (T × λ-c-j))/T, in a viewpoint of even distribution.
In checked redundancy architecture, λ-c-i = λ-c-j.

Thus, we can use the following generic formula for the residual hazard risk assessment to demonstrate whether the controller meets its qualitative Safety Integrity Level (SIL) target:

$HR_{2ooX} = C_2^X T \times (\lambda\text{-c-i})^2$, X = 2 for 2oo2 controller design, X = 3 for 2oo3 controller design.

The risk estimation in the method stated here is specifically about the failure detection vitality through the checked redundancy lockstep voting. Further, the residual risk level on enforcing safe reactions to the failures detected through the voting is assessed by taking a count of the number of work cycles required for the reaction to be in effect. This approach of assessing the controller's residual risk is credible because it is based on the worst case that any failure in a checked redundant channel is treated as a hazard contributor. With consideration of certain I/O ports that can be in either high or low state for a long time, which can contribute to dormant failures, the reactive fail-safe concept in the self-checking and self-testing techniques is further designed in for each channel to test its I/O monitoring circuits. This is discussed in the following section.

Principles of Reactive Fail-Safety in Self-Checking and Self-Testing

The reactive fail-safe concept requires that critical components and failure detection circuits be checked and tested to reveal possible dormant failures, which the checked redundancy voting may not be able to detect in time and which could result in a condition of multiple cascade failures. Thus, the self-checking and self-testing logic and mechanisms are designed into each checked redundancy channel to be performed at controller start-up, the hardware rest and the online available time. Also, a scheduled self-testing (normally once per day) logic is designed to fully test I/O channels and power monitoring circuits, as well as watchdog circuits.

Critical Assumption

A critical assumption associated with this concept is independency between the application functions and self-testing/monitoring functions so that the testing mechanisms will not result in unintended permissive outputs.

Self-Checking and Self-Testing Design Principles

In general, the following principles are proposed for using the diversity and self-testing/monitoring safety concept:

1. The start-up testing in a safety-critical controller must fully test its integrity, including the main computation unit and the safety-related/critical input and output paths.

2. Periodic self-testing mechanisms must be designed in for detecting the controller internal safety-related/critical I/O paths, as well as all monitoring circuits
3. Online self-testing CPU and Random Access Memory (RAM)/ Read Only Memory (ROM) health status, along with safety-related/critical I/O paths must be performed. The period during which the self checking is performed must be short enough that the unsafe effect of a detected failure can be mitigated before a combination with other failures results in losing the capability to enforce safe state whenever it should.
4. Online self checking of the software image and data integrity must be completed within the predefined time window.
5. Closed-loop self checking must monitor a command and its check-back must be designed in for detecting failures in any output path in real time.
6. A watchdog as cycle self-checking monitor feature must be designed in to detect software execution locks or work-cycle overrun, and to enforce the safe state due to violating real-time constraints as determined for the system.

“The Failure Mode Effects Analysis (FMEA) and Failure Mode, Effects and Criticality Analysis (FMECA) must be performed on the usage of the expected fail-safe components in the application context in conjunction with an analysis of the systematic safety designs. The safety analysis must verify that components selected for fail-safe characteristics conform to those in current, well-known standards.”

Self-Checking and Self-Testing Design Verification

The self-checking and self-testing design in the CBTC vital controllers is analyzed and tested for verification that it satisfies the associated principles. Each CE in the MPU and each PPE in the PPU of a safety-critical controller performs CPU testing and memory checks as online background tasks. A separate watchdog is designed for each CPU to monitor software execution locks and the work-cycle time window. Each output path for commanding controlled devices has a check-back as the monitoring feedback to enable closed-loop supervision to the voter. The scheduled periodic tests fully test the input and output path, as well as the monitoring circuits. The software image, the database and the internal state parameters in each checked redundant channel are protected by safety code (i.e., data block with its cyclic redundancy check code). The online self checking can detect internal data corruption, which triggers the CE and the PPE to enforce a safe state.

Because it is possible that an input or output may rarely change its state between “low” and “high” during normal operation, the residual risk of each I/O path is assessed based on the conservative condition that both an I/O path — and its testing monitoring circuits — encounter failures during the periodic test interval. If the total independence between the command path and the monitoring circuits cannot be ensured, any safety-critical input or output in logic needs two physical paths in the concept of dual-cuts or double-cut circuits.

Principles of Intrinsic Fail-Safety

The intrinsic fail-safe design principles are embedded in the PPU interfaces and the MPU vital communication interfaces. The PPU uses vital relays organized in double-cut design, coupled with a status-back architecture, and is known to be self-revealing during failure conditions. Upon a failure in the output path, the MPU can detect and react by enforcing the output to a safe state. The vital communications between safety-related critical controllers through data communication links or networks require protocols with logical fail-safe properties of safety code to support the CBTC safety features. Also, this concept is used for designing controller watch-dog (vital supervision card) circuits for both MPU and PPU to react to processor halts or deadlock conditions.

Critical Assumption

A critical assumption associated with this concept is that multiple, independent self-revealing component failures will not occur simultaneously (within the time it takes for the first failure to occur and safe action to be taken).

Intrinsic Fail-Safety Design Principles

Fail-safe hardware must achieve accurate identification and prediction of all failure modes and characteristics within the application context. The occurrence of failure modes that could have unsafe consequences is eliminated, prevented or otherwise accounted for by design with qualitative safety mitigation in the application context. The safety analysis must justify the intrinsic fail-safety design against the dependent factors associated with the components used in the design.

Intrinsic Fail-Safety Design Verification

The intrinsic fail-safety concept depends on the intrinsic properties of the used components. This means that intrinsic fail-safety properties are the characteristic of a component, circuit or device within specific application context, such that no failure modes could cause an unsafe condition. The degree of safety achieved is dependent on:

- The correctness of selected component failure characteristics
- The comprehensive and accurate identification of all component failure modes
- The extent to which all combinations of failure modes can be analyzed

The Failure Mode Effects Analysis (FMEA) and Failure Mode, Effects and Criticality Analysis (FMECA) must be performed on the usage of the expected fail-safe components in the application context in conjunction with an analysis of the systematic safety designs. The safety analysis must verify that components selected for fail-safe characteristics conform to those in current, well-known standards.

Conclusion

Designing and demonstrating a safety-critical controller that meets expected safety properties has high cost and schedule risks under a traditional safety engineering approach to getting consistent and complete safety design requirements in time, based on top-down hazard analyses. To reduce such risks, using a set of safety concepts and their application principles as the design guideline is practical and feasible to let design and safety engineers focus on mitigating controller internal failures that result in an unexpected permissive output as a generic hazard.

This paper discussed this approach and composite fail-safe, reactive fail-safe and intrinsic fail-safe concepts, as well as their application principles. Based on our experience with developing controllers for CBTC systems, an effective combination of these safety concepts and their principles can lead to defining safety requirements for achieving the vitality of a safety-critical controller. The controller design in this approach is based on lockstep voting in checked redundancy architecture, self-checking and self-testing in each process channel to detect the first internal failure and react to it in time, which can prevent controller internal failures from contributing to system application-level hazards. Further, this approach can simplify the estimation of the controller residual risk level, and facilitate assessment of the controller safety integrity level.

About the Author

Fenggang Shi, Ph.D., is senior expert and chief safety architect at Thales Canada Transportation Solutions in Toronto, Canada. He has 25 years of experience in the field of CBTC system safety engineering. Shi has been the technical leader and supervisor of safety teams on more than 40 CBTC systems and products for future signaling.

References

1. CENELEC BS EN 50129. "Railway Applications — Communication, signalling, and processing systems — Safety Related Electronic Systems for Signalling," European Committee for Electrotechnical Standardization (CENELEC), 2018.
2. IEEE Std 1483-2000. "IEEE Standard for Verification of Vital Functions in Processor-Based Systems Used in Rail Transit Control," IEEE Standards Association, 2000.
3. Shi, F. "Defining Layered Safety Concepts based on Open System Architectures as Foundation for Multi-Suppliers to Develop Interoperable Safety Critical Systems," *Proceedings of International System Safety Conference*, 2014.
4. Shi, F. "Using Layered Safety Objectives and Concepts to Guide Large Scale System Designs for Achieving Built-in Safety Properties in Hierarchy," *Proceedings of International System Safety Conference*, 2015.