The Educational Review, USA, 2017, 2(1), 147-154
http://www.hillpublisher.com/journals/er

Functional Diversity Design of Safety-Related Systems

Ivan Malynyak

Volonterska Street 63, Apt. 133 Kharkiv 61093, Ukraine

How to cite this paper: Malynyak, I. (2017). Functional Diversity Design of Safety-Related Systems. *The Educational Review, USA*, 2(1), 147-154.
http://dx.doi.org/10.26855/er.2018.01.004

Corresponding author: Ivan Malynyak, Volonterska Stree 63, Apt. 133 Kharkiv 61093, Ukraine.

Abstract

Traditionally, the usage of widespread safety voted-groups architectures is a matter of redundancy, where hardware and software components are replicated which leads to drastically decreased system reliability; therefore necessity of functional diversity is become essential. Well known process of N-version programming minimizes the probability of producing similar erroneous results, but In this paper the combined software and hardware methods to achieve safety system requirements without enlarged implementation price is proposed. Avoidance of redundant complexity with limitation the number of system's internal states is naturally led to functional diversity with residual and common cause faults are decreased to achievable level.

Keywords

Fault Folerant Architectures, 1oo2D, 2oo3, Redundancy, Complexity, Control systems, Diversity

1. Introduction

Nowadays technology of system building is believed to be effective and safe, where making essential decisions are sufficiently exploited (IEC 61508-3, 2010). The current generation of safety systems is highly integrated digital complexes which offer better performance and additional diagnostic capabilities in comparison with aged analog systems. But as far as system is going to be intricately big (like the voted-groups architectures) the questions of reliability and possibility of latent faults becomes a challenge.

As shown in Fig. 1, two types of well-characterized architectures (2oo3 and 1oo2D) is built under equal functional structure schemes (A≡B≡C) and therefore synchronization philosophy of all operations within units in each system cycle is required. As a result, extra data connection lines between units are used to establish complete diagnostic and information support. Although projects very often become fed up with wiring and software redundancy managing, the approach of managing identical units' functional algorithm is widely used and implemented.

As a result, digital system could be triggered to perform unexpected actions or failed to provide safety functions due to injected faults during the design or the manufacturing process.

The redundancy alone may not be sufficient way to achieve safe and reliable operation when considering the effects of common cause failures (NP-T-1.5, 2009). In many safety systems diverse approach is required to provide alternative functions that will mitigate the harmful effects of an accidents. However, in most architecture, the units of a redundant system have identical functionalities and designs. The concurrent triggering of multi-channel mechanism could be potential failure and it's like group of blind man walking simultaneously around a pit.

It is recognized that it is not possible to prove that all potential faults couldn't be identified and that's why different physical methods or different design approaches has to be implemented to perform safety functions. It's all about matter of diversity, whether it would be human, functional or design one (Avizienis, 2001).

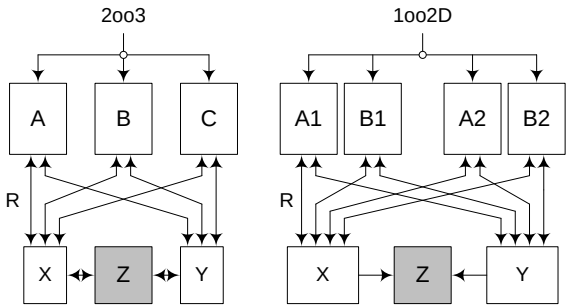

Figure 1. Two types of well-characterized architectures 2oo3 and 1oo2D.

In spite of the fact that diversity questions are deeply concerned and surveyed, the biggest issue still lies in assumption of equal "units", where the fundamental problem took their roots. Of course, it's easy to fit out the system with all kinds of equipment, but absence of diverse feedback can't guarantee prolonged operation without common cause failures.

The answer could be found in hardware reduced channel design as "satellite"-type (sequential) architecture, where two parallel equal units are substituted with different types of ones with minimizing redundant wiring as shown in Fig. 2. The development of such safety redundant control systems could be similar to nature evolution with satellite design, where necessity of redundancy implements as sequential algorithm in embedded bundle of diverse units instead of parallel working of identical ones (Mukai, 1974).

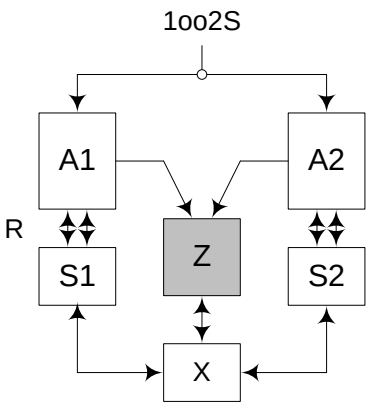

Figure 2. Satellite architecture of safety redundant control systems 1oo2S.

Data lines in satellite architecture 1oo2S are minimized to essential level with safety functions are guaranteed to be executed correctly. Although output safety approach remains the same for all architectures (X∩Y=>Z), the involvement level of output components (X, Y) are differed on the case of functionality. Herein, component (Y) is removed in 1oo2S architecture and is modified as a part of executive mechanism (Z) for safeguard with ability to stop. By contrast component (X) is supposed to drive output solely with all necessity requirements of dependability.

The satellite-type architecture isn't widespread despite simplicity and natural application because of general established approach used to cover safety issues. Nevertheless, this direction would have its future in the light of diversity and necessity of dropping system complexity to widespread acceptable level.

2. Redundancy

The challenge of providing redundancy management to meet certain requirements of fail safe for different application is complicated by the critical constraints of market cost and schedule. Evolution of redundancy management with its intricate synchronization interfaces and overall complexity were found to be potentially catastrophic (Boykin, 1983). This approach obviously was costly in cost of wiring and components but on the contrary is offered rather simple and attractive macro level "black box" design.

The first step down from simple "black boxes" could be avoiding some needless redundancies. As shown in Fig. 3, 1oo2D architecture is looked safe due to transparent functional scheme. Indeed, all inputs are duplicated and could be verified by other neighbor. Moreover, output is drove from two units, where both of them are received feedback data and could stop forming the output.

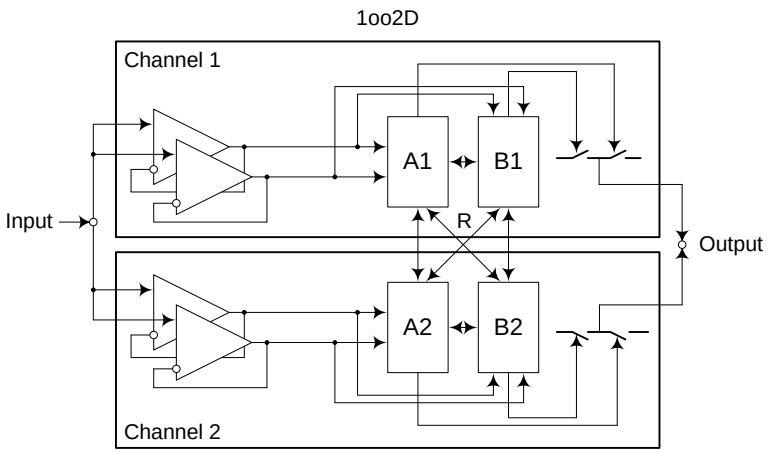

Figure 3. Decomposition of 1oo2D architecture.

The classical design to achieve necessary safety level in each channel leaded out into duplicating functions almost at every step. Firstly, the safe input means two signals to be monitored and safe output is based on the same repeating principal. Secondly, to achieve availability requirements the "shape" just went on adding next "channel 2", but of course it could be much more channels. And finally, when all components are successfully placed in detailed places, the intricately net of data communicating has to be put over.

As one could see, the classical approach is well known and understandable, where failures of elements could be easily found by comparison. For simple functions it works greatly and reliability decreasing is not matter of consideration. But when dozens of input and output were designed with redundancy, cost of developing and physical implementation tended to increase exponentially. The result of complexity already known and contained to be in faults propagation which leads to unsafe concurrent failures of two or more identical units (Madden, 1984).

On the contrary, the satellite architecture approach could change the whole appearance of system design, where redundancy eliminating is the first step to be done. As shown in Fig. 4, the point of view lied in the basic idea sequential approach, where elements of system are presented with non-symmetrical faults property. For example, if "input" is implied

as thermocouple, than implementing fixed current with periodical caliber testing would guarantee possible faults detection. Indeed, the only fault is not detected by this approach is a rare gradual measurement drift and by using high quality thermocouple with periodical external checkup could solve particular problem due to low possibility of such fault. As a result, a thermocouple with measuring circuits is treated as single element with non-symmetrical types of faults, where most likely defects in circuit break or rapid measurement drift (op-amps or connecting line issues) are easily detected. The non-symmetrical fault type output is treated in the same way, where satellite unit (S) drives relay (K) safely through transformer to eliminate shortcut failures. The main unit (A) is connected to satellite with two communication lines where first one is performed both algorithm timing checking and another one is used to analyze units' functional integrity (R). Either unit could run safety function by stopping transformer in case of discrepancy.

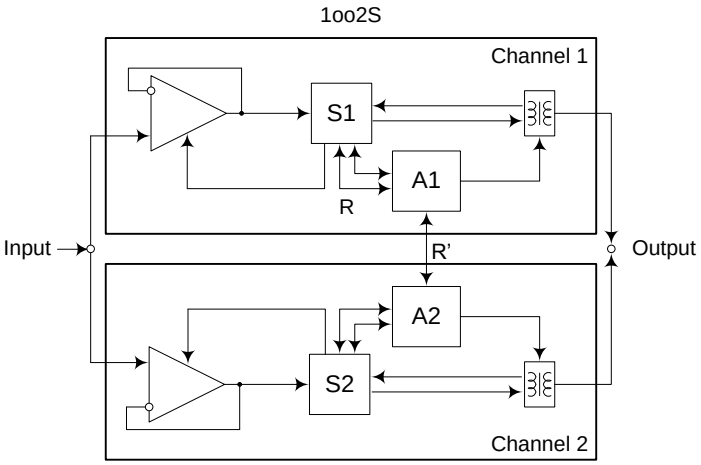

Figure 4. Decomposition of satellite architecture 1oo1S.

As a result, sequential approach is constrained to build the system with strong separation between core algorithm and routine input/output tasks, whereupon the fast and simple satellite unit is prepared to deal only with input/output where other unit is managed with functional algorithm. The effect would come out of hardware complexity reduction, common cause failures elimination and reliability enlargement due to algorithms simplification and functional diversification.

3. Hardware Diversity

The main aim of diversity is perform safety functions. In most situations, safety function is achieved by a monitoring the behavior of number of functional units and is executed by at least two hardware channels (monitored redundancy). The way safety ideas are implemented reminds "fractal" pattern, where general method of reiteration is repeated at every scale of a project. It could be seen in software, where developer has no idea how his source code would be implemented into machine language. The same thing is happened in hardware, where safety issues are implemented by pure repeating elements based on monolithic integrated circuits without clear 100% understanding of its practical functionality. As a result, each level of system decomposition is considered with increasing uncertainty. The core problem of fractal pattern is laid in the mistaken understanding principle of aggregation where uncontrollable cascading of simple elements may not be as reliable as it expected.

But what could be done if system has to be reliable and safe altogether? For example, safety-related diversity in relay scheme would be useless because its fully determined and only problem of reliability needs careful consideration. One of

the approaches is hidden in sequential architecture which leads to functional diversity. Evolution of Nature and species shows that repeated and uncontrolled multiplication ultimately leads to disaster, just look back on dinosaurs or cancer. So, the main idea of safety lies not in redundancy issues as one may think, but in capability of system predicting its behavior with minimum complexity, for example "As large as you need and as small as you can" for implication of the brain size (Davidson, 2007).

The result of uncontrolled "fractal" implementation is come out as necessity of unit diversity due to providing predictable system behavior, where growing software and hardware is turned to multiplication its complexity inside two essential system degrees of freedom which have to be carefully verified (Astrom, 2008). Hereafter, when system is installed, a well known golden rule is remained - "don't change anything".

From other hand, when sufficient data of failures is gathered, system availability could be increased due to overall simplifying. For example, if it's clear how light bulbs in mines are breaks and what circumstance is preceded, then particular algorithm with appropriate feedback could lead to only essential abundance of lighting control system or even eliminate redundancy at all due to accurate prediction of failures with using certain planned actions.

After thorough revision of prevail systems with 1oo2D architectures it could be transformed into simplified 1oo2S architecture, where all inputs and outputs are designed with feedback (shown in Fig. 5).

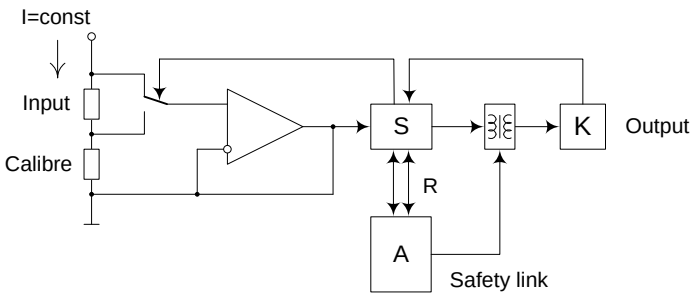

Figure 5. Decomposition of 1oo2S architecture.

It is recognized that it is not possible to prove that all potential faults had caused by unpredictable uncertainties are identified and successfully captured by eliminating or mitigating functions. Avoidance of unnecessary complexity with limitation the number of internal states and complete set of functional diversity would decrease residual and common cause faults to achievable level (NP-T-1.5, 2009).

In order to comparing 1oo2D and 1oo2S types of architectures a number of variables have been putted in:

- $\boldsymbol{R}$ – set of results of software blocks
- $\boldsymbol{A}$ – safe output finite state of unit $\boldsymbol{A}$
- $\boldsymbol{B}$ – safe output finite state of unit $\boldsymbol{B}$
- $\boldsymbol{S_A}$ – set of normal responses of unit $\boldsymbol{A}$ placed in unit's $\boldsymbol{S}$
- $\boldsymbol{A_S}$ – set of normal responses of unit $\boldsymbol{S}$ placed in unit's $\boldsymbol{A}$
- $\boldsymbol{P}$ – set of proper events $\boldsymbol{p}$
- $\boldsymbol{Q}$– set of fault events $\boldsymbol{q}$
- $\boldsymbol{Z}$ – set of safe output finite states
- $\boldsymbol{Z'}$ – set of unsafe output finite states

For the preliminary evaluation of 1oo2D safety function it's assumed that proper $\boldsymbol{p} \in \boldsymbol{P}$ or fault $\boldsymbol{q} \in \boldsymbol{Q}$ inputs are used for all units $\boldsymbol{A}$, $\boldsymbol{B}$ with equal results of software blocks $\boldsymbol{r} \in \boldsymbol{R}$ to implement safety function $\boldsymbol{Z}$ (shown in Fig. 1). For the se-

quential safety function inputs is used only for unit ***S*** which has both diagnostic and data communication lines ***R*** with unit ***A*** (shown in Fig. 2).

As discussed above, formal specifying of criteria is used to analyze fault-tolerance of two safety function forms. For 1oo2D architecture $\boldsymbol{A} \Leftrightarrow \boldsymbol{B}$ is assumed where software could be diversified or not.

Working states where function operates normally:

$$(\forall r, \forall p: Arp=Brp)\ (X=Y \Rightarrow Z)\ (Z \wedge Z'=\varnothing) \quad (1)$$

Failure states where units' outputs synchronously being changed by faults (common cause failure):

$$(\exists r, \exists q: Arq=Brq)\ (X=Y \Rightarrow Z)\ (Z \wedge Z' \neq \varnothing) \quad (2)$$

Failure states where units' outputs are unsafe and equal (safety case issues):

$$(A \Leftrightarrow B)\ (X=Y \Rightarrow Z)\ (Z \wedge Z' \neq \varnothing) \quad (3)$$

Software for sequential form is considered to be easier and diverse with $\boldsymbol{A} \wedge \boldsymbol{S}=\varnothing$ equation is assumed. Each unit in the sequential architecture has acceptable set of responses applicable to other one.

Working states where function operates normally:

$$(\forall r, \forall p: Arp \subseteq S_A \wedge Srp \subseteq A_S)\ (X \Rightarrow Z)\ (Z \wedge Z'=\varnothing) \quad (4)$$

Failure states where units' outputs are unsafe and equal (safety case issues):

$$(A \Leftrightarrow S)\ (X \Rightarrow Z)\ (Z \wedge Z' \neq \varnothing) \quad (5)$$

Equation 5 is very rare to happen due to absence of equal functional in ***A*** and ***S***:

$$(\forall r, \forall p: Arp \wedge Srp \neq \varnothing) \quad (6)$$

4. Functional Diversity

Reliable operation is the most important attribute of safety system where replacing the faulty parts with fixed ones associates with great cost resulted ranging from disgruntled users to financial damage and loss of life. Fundamental solution in hardware faults characterization is realized as early as 1956 by Von Neumann by introducing redundancy with substantial increases in reliability (Von Neumann, 1956). But to be broadly deployable, the hardware reliability solutions must incur low overheads, precluding use of expensive redundancy.

These observations is motivated a strategy where faults are detected by watching for anomalous software behavior. One of the well-known techniques was introduced as a dynamic implementation verification architecture or "DIVA" (Austin, 1998), where processor is created by splitting a traditional hardware design into two parts: the speculative core and the functionally robust checker. The approach was robust enough for masking functional bugs that were introduced during the simulator's development due to functional verification stage was added at retirement.

This idea was further enlarged by resilient software anomaly treatment project "SWAT" where software-centric detection is provided mechanisms for diagnosis and repair based on "symptoms" of faults approach, using low-cost hardware and software monitors (Li, 2008). SWAT system is assumed a multicore design where a fault-free core is always available and hardware ability to repair by performing rolling-back execution on a different core is processed by checkpoint/replay mechanism. Further development of SWAT strategy is found in likely program invariant solution iSWAT, where program variables ***x*** during all executions are assumed to be in certain interval $\boldsymbol{Min} \leq \boldsymbol{x} \leq \boldsymbol{Max}$ inferred from additional modeling and training (Sahoo, 2008).

And of course N-version program methodology (NVS) as the independent generation of functionally equivalent programs from the same initial specification is concerned in a way of unique advantages of fault-tolerant diversified software (Avizienis, 1995).

By looking at the basic ideas of each design from diversity point of view it could be seen that relay, voting-architectures and N-version software systems have no hardware (HW) diversity (do not muss up with redundancy). The other systems which are based on DIVA have great HW diversity in processor which is split into Core and Checker parts to be able to detect errors in own computation. Also DIVA software (SW) diversity is assumed to have minor differences due to necessity of Checker verifying the correctness of all Core computation with pre-computed inputs and outputs.

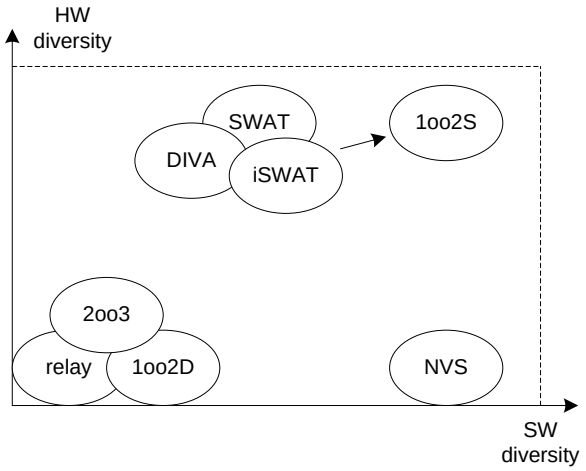

Figure 6. Functional HW/SW diversity system design comparison

Despite the fact, those modern fault detectors are able to identify more than 98% of hardware errors in many structures, for the safety issues this level is not enough (Sahoo, 2008). There are still problem with residual 2% of non-masked faults, which could hardly been identified due to defective software design and insufficient diversity. Indeed, if software faults are undetected, than voting-architectures systems without SW diversity could move to potentially unsafe state ***Z'*** (7). In case of NVS or DIVA system design most of common cause failures are uncovered, but as far as overall diversity is far from 100 %, than the possibility of system unsafe reaction is have to be counted (8).

$$(A \Leftrightarrow B)\ (X=Y \subseteq Z') \tag{7}$$

$$(A \wedge B \neq \varnothing)\ (X \wedge Y \neq \varnothing)\ (Z \wedge Z' \neq \varnothing) \tag{8}$$

In order to avoid potentially unsafe states, a number of conditions must be met (NP-T-1.5, 2009). This paper is regarded only to conceptual software and hardware design differences, which have to base on distinguish functional specifications. Let us assume software algorithms of ***A*** and ***B*** with symmetric difference feature, and then in theory, it's possible to reach necessary system safety level (9).

$$(A \wedge B \approx \varnothing)\ (X \wedge Y \approx \varnothing)\ (Z \wedge Z' \approx \varnothing) \tag{9}$$

One of the approaches to achieve units' software symmetric difference $(A \wedge B \approx \varnothing)$ could be based on iSWAT "likely program invariants" idea with substitution invariants property of critical parts for negations ones $(a \Leftrightarrow \neg b)$**.** For example, if unit's ***A*** algorithm has ***CASE*** operator for "what must be done", than unit's ***B*** algorithm to control this part of software should have operator for "what is not supposed to be done in any case".

The idea of software symmetric difference could potentially cover 99.9% faults without overhead in performance, throughput and power consumption, but the problem of sufficient proofs must be confirmed with a positive exploration.

5. Conclusions

Advanced safety redundant control system architecture 1oo2S combines the benefit of 1oo2D and 2oo3 systems with higher availability and higher safety levels. The price of this innovation lies in additional functional requirements to successfully analyzing integrity of worker unit running kernel algorithm. As a result of reduced architecture implementation with simplified satellite units, the higher availability and higher safety could be reached comparing to prevalent systems.

The approach to the architecture design outlined above would place all of the safety functions in a primary safety unit A where subset of functions is implemented in a smaller independent unit S (satellite). Because it's obvious that units A and B would not process the same variables, the potential for common signal trajectory have been adequately addressed. In this case, all standard measures of diversity are recommended to reduce the chance of a common fault.

References

IEC 61508-3. (2010). Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems–Part 3: Software Requirements.

NP-T-1.5. (2009). Protection Against Common Cause Failures in Digital I&C Systems of Nuclear Power Plants. IAEA Nuclear Energy Series.

Avizienis, A., Laprie, J.-C., & Randell, B. (2001). Fundamental Concepts of Dependability. Research Report No 1145, LAAS-CNRS.

Mukai, Y., & Tohma, Y. (1974). A Method for the Realization of Fail-safe Asynchronous Sequential Circuits. *IEEE Trans. Computer*, 23(7), 736-739.

Boykin, J., Thibodeau, J., & Schneider, H. (1983). Evolution of Shuttle Avionics Redundancy Management/Fault Tolerance. Space Shuttle Technical Conference, NASA Conference Publication 2342. Part 1, Johnsons Space Center, Texas, 1-18.

Madden, W., & Rone, K. (1984). Design, Development, Integration: Space Shuttle Primary Flight Software System. *Communications of the ACM*, 27(9), 914-925.

Davidson, I. (2007). As Large as You Need and as Small as You Can: Implications of the Brain Size of Homo Floresiensis. In Schalley, A., Khlentzos D., Mental States. V.1, Evolution, Function, Nature, 35-42,

Astrom, K., & Murray, R. (2008). Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press.

Von Neumann, J. (1956). Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. Automata Studies. *Annals of Mathematical Studies*, 34, 43-98.

Austin, T. (1998). DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In International Symposium on Microarchitecture (MICRO).

Li, M., Ramachandran, P., Sahoo, S., Adve, S., Adve,V., & Zhou, Y. (2008). Understanding the Propagation of Hard Errors to Software and Implications for Resilient System Design. In International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS, Seattle, Washington, USA.

Sahoo, S., Li, M., Ramachandran, P., Adve, S., Adve,V., & Zhou, Y. (2008). Using Likely Program Invariants to Detect Hardware Errors. In Conf. Dependable Systems and Networks–DSN, 70-79.

Avizienis, A. (1995). The Methodology of N-Version Programming. In Book: Lyu, M., Software Fault Tolerance. Wiley & Sons Ltd, 23-46.