Monoprocessed computer architecture for safety critical applications

J. Rady de Almeida Junior¹ & J. A. Fonseca²

¹Safety Analysis Group – GAS, Department of Computer Engineering and Digital Systems – PCS, Escola Politécnica da Universidade de São Paulo, Brazil.
²Alstom Transportes do Brasil, São Paulo, Brazil.

Abstract

Electronic systems with safety-critical requirements must be projected following the fail-safe equipment philosophy. Therefore, all operational states and all failure states must be known and analyzed. If one of these failure states has been reached, prompt action must be taken to leave it, preventing great material losses, or risk to the people or the environment.

Monoprocessed computer architecture is presented for use in safety-critical applications, such as train control systems. The main advantages in using this kind of architecture are its higher reliability and safety and lower cost, when compared to other architectures, such as duplicated architectures with a comparator, or triplicated architectures with a voter.

The proposed monoprocessed architecture is composed by a main processor and a watchdog processor. The watchdog processor has the function of monitoring the main processor behavior. The watchdog processor is much simpler than the main processor. The main processor software is designed using special coding techniques. The basic technique is the software replication that consists of two or more different implementations of the same program specification. These software versions should produce the same results and are executed sequentially in the same processor.

The target of this proposed architecture is to increase the system fault coverage factor. This factor is responsible for the detection of any fault that may be dangerous to the system, enabling the prevention of such condition.
1 Introduction

Old control and supervisory systems are being replaced more and more frequently by computerized systems. Some of the current advantages of that substitution are thoroughly the largest control capacity, the answer speed and performance, very superior to any other technique until then used. Beside this, it is possible the inclusion of new functions, until then not available with the old technologies.

Safety related critical systems are those in which a fault can have as consequence serious material damages or damages to the environment, or still represents danger to the human life, be operators, be possibly the population reached by a fault of the system. Considering that the number of those systems are growing day by day, it becomes necessary to accomplish activities that seek to obtain a prevention of the number of accidents, or still a decrease of its consequences.

The main problem is the safety subject of those new control and supervisory systems. The hardware fault modes of a computer are extremely complex and of difficult foreseeing. Software does not present wear, but it almost always contains mistakes, whose consequences are not always immediately understood or evaluated. It can be said that there is not now an accepted technique that evaluates the software reliability, in the same way that exists techniques for hardware.

The development of those systems considered critics is, usually, controlled by government regulation that establishes approaches of systems certification in each area. The efforts and resources invested in the prevention of accidents are fully justified when a loss is considered serious or important.

The safety related activities should start when the system begins to be conceived and must have sequence in the project, production, test and operation of the system. The project should already emphasize, since its beginning, the aspects related to the safety, avoiding to increase new elements subsequently. Safety should be seen as a whole. It is not enough to assure the correction of subsystems of a larger system.

It can be thought about the possibility of making exhaustive tests of software, verifying its behavior in all possible paths. This is impracticable, due to the enormous number of states that software (even not very sophisticated) can assume.

Another trend points to the use of formal development techniques. These techniques did not still reach a stage considered acceptable for wide use. They are just used in small pieces of programs, due to its application complexity. The great problem here refers to the proper training of the technical team.

2 Risk

The human being was always subject to risks. In the past, those risks were just constituted by the natural factors, such as hurricanes, earthquakes, etc. Now,
Beyond of that natural factors, there are the artificial ones, whose origin are the new developed systems, such as accidents with airplanes, explosions in chemistry and nuclear plants, among others.

The technology development allowed the decrease of the natural risks, because man learned how to foresee and to control its effects. That same progress provided great benefits to the humanity, bringing also the inherent risks, that is to say, an increase of the called artificial risk [01].

It can be said that in modern systems there is always an inherent risk because of the technology. There is a direct reduction of the risk because the control systems used and also because other means, such as modern fire combat equipment or efficient heat detectors. It always remains a risk portion, called residual risk, that can or not be accepted by the society. Accept or not the residual risk depends a lot on how developed or conscious is the society. For example, people of less developed countries tend to accept a much larger risk portion than people of first world countries.

Generally, the acceptable risk is obtained starting from a balance among the benefit brought by the new technology and the risk degree that the society is disposed to run for that benefit. That risk depends on social and cultural factors, of the value that is attributed to life, property and environment.

All modern systems have effective control systems, which have as one of its functions to reduce the inherent risk of an application at an acceptable level. An expression accepted to define the risk is the following:

\[
\text{Risk} = \text{Frequency of Dangerous Event} \times \text{Severity of Consequences}
\]

The more frequent the event or the more larger the severity of the consequences, larger the associated risk.

2.1 Acceptability of Risk

The acceptability of a given level of risk is determined by the benefits associated with that risk, and by the amount of effort that would be required to reduce it. Risk with catastrophic consequences and that could occur frequently is not tolerable. On the other hand, risks that cause negligible consequences with frequent occurrence, or risks with catastrophic consequences that are improbable or even remote to occur can be acceptable.

The IEC 61508 standard classifies the risk into three levels [02]:

- Intolerable risk: when the consequence is intolerable and its occurrence cannot be justified;
- Acceptable risk: when the consequence, though not insignificant may be acceptable under certain conditions;
- Neglected risk: when the consequence is insignificant and it can be neglected.

In the case of safety critical system, a particular risk is acceptable if it is as low as it is reasonably practicable, defining the ALARP level [02]. The risk level satisfying this criterion is named tolerable risk for one given application. Nevertheless, it is important to consider that a risk within the ALARP level is
never acceptable if it can be easily reduced. Therefore a proposed system that has a very small risk may be judged unacceptable if that risk is unjustifiable. Conversely, a system that has a significant risk may satisfy the requirement if it offers sufficient benefits, and further reduction of this risk is considered impracticable.

The requirements change from one to another application, and each electronic system with safety demand must follow certain requirements, that are project targets to be accomplished [03]. It is important to point out that a safety critical electronic system maintains its safety characteristic in the universe of its application. While the basic requirement of a train control system is to stop the trains, the basic requirement of an airplane is to maintain the flight.

In a project it is extremely difficult or almost impossible to turn compatible requirements of high reliability, safety and availability with a low cost. It must be evaluated which of them must stand out.

So, the focus of this paper is to show the positive and negative aspects of a monoprocessed architecture that does the control and safety functions, comparing it with the duplicated architecture.

The system presented in this study is applied to the control of a generic process in which the safety state is achieved when the control system is disabled, interrupting the process.

3 Comparison Between Architectures

The cost of a duplicated architecture is basically twice the cost of an architecture that uses only one processing unit. It was not considered the additional mechanical details that accommodate the duplication [04].

The failure rate of the duplicated architecture is twice the failure rate of a monoprocessed architecture [05]. The reliability of the arrangements is obtained through the expression $R(t) = e^{-\lambda t}$ and the Mean Time To Failure (MTTF) is given by

$$MTTF = \int_0^\infty R(t)dt$$

The evaluation of the architectures about safety is made through the use of models based in Markov chains [05]. The model of figure 1 is used for the safety analysis of duplicated architecture.

![Safety Evaluation Model for Duplicated Architecture](image)

© 2002 WIT Press, Ashurst Lodge, Southampton, SO40 7AA, UK. All rights reserved.
Web: www.witpress.com   Email witpress@witpress.com
Paper from: *Computers in Railways VIII*, J Allan, RJ Hill, CA Brebbia, G Sciutto and S Sone (Editors).
Where,
- **State 0**: Safe state of the system, including the states of correct operation and states with safe failures;
- **State 1**: Incorrect state – there was an unsafe failure in one of the channels;
- **State I**: Unsafe state

\[ MTTUF = \int_{0}^{\infty} S(t) \, dt \]  

\( MTTUF \) – Mean Time To Unsafe Failure

Where, \( S(t) = P_{0}(t) + P_{1}(t) \) or \( S(t) = 1 - P_{1}(t) \)  

(\( S(t) = Safety \))

The model for only one processing unit is presented in figure 2.

![Figure 2. Safety Evaluation Model for Monoprocessed Architecture.](image)

Where,
- **State 0**: Safe state of the system.
- **State I**: there was an unsafe failure

\[ MTTUF = \int_{0}^{\infty} S(t) \, dt \]  

and,

\[ S(t) = P_{0}(t) \text{ or } S(t) = 1 - P_{1}(t) \]  

where,

\[ P_{0}(t) = e^{-\lambda_{U} t} \]  

Starting from these models, it can be observed that the architecture with only one processing unit has a lower cost and a higher reliability, when compared with the duplicated architecture. But for safety critical applications, the monoprocessed architecture has a very low \( MTTUF \), when compared with the duplicated architecture.

One attempt to turn the monoprocessed architecture equivalent to the duplicated architecture in the aspect of safety is to decrease the unsafe failure rate of the processing unit. This can be achieved through an increase in the fault cover factor.

### 4 Monoprocessed Unit

In the project of a system for safety applications, there are additional cautions to be followed, when it is used an architecture with only one processor unit.

The fault cover factor is determined by the tests that exists in the software and hardware of the system. In the models of architectures with replicated units, the fault cover factor is amplified by the comparison of results among the
replicated units when confronted with architectures without replication where such comparisons are not possible.

The proposed architecture [06] has the property of increasing the fault cover factor, by the addition of a second processor, much simpler than the main processor. Therefore, the coverage is greatly increased, because one processor looks at the behavior of the other processor, signaling in case of any problem.

The proposed architecture is composed by a main processor and an auxiliary processor called watchdog processor [07], as presented in figure 3.

![Monoprocessed Architecture](image)

The main processor is responsible for the fulfillment of control and safety functions. The microprocessor architecture may be of RISC type with pipeline and internal cache to have high performance. This is necessary, so main processor must run the two replicas of software and still attend the time requirements for the applications.

The complexity degree of the main processor does not affect directly and significantly the safety, which is supervised by hardware and software mechanisms [06].

The standard MIL-HDBK-217F does not consider that a growth in the number of processor functional units increases its failure rate. This standard considers as relevant only the number of bits of the data bus.

The main processor has an output called Timestamp used by input devices, giving the process variables a stamp of time, turning dynamic its input data. The main processor checks if the watchdog processor is making correctly its processing, using as diagnosis a set of signatures generated by the watchdog processor. If it is detected any type of fault in the watchdog processor or even in the main processor, the vital comparing device outputs are disabled by signals of toggle.
The input interfaces receive data from field, making its processing in a duplicated manner, by independent input circuits. The input interfaces add the time stamp value supplied by the main processor, to the data received.

The basic function of the watchdog processor is to supervise the working of the main processor and inhibit the output of vital comparing devices, if it detects any fault, be in the main processor, be in the proper watchdog processor.

The mode of work of the watchdog processor is the same of the main processor. The watchdog processor verifies the correct operation of the main processor through a set of signatures generated by this one.

This technique of the watchdog processor is based in a more general technique called safety bag [02].

The watchdog processor is a small and simple co-processor [07]. It can be considered that there are much less failure modes in this kind of processor. The watchdog processor can also be used to detect errors caused by the software, if it is used to do checks about the consistency of results of the main processor.

The use of a watchdog processor gives diversity to hardware, adding protection against common mode errors and independence in the verification of the main processor [07].

5 Software Replication

Any method of error detection is based in the information redundancy. The same data is produced several times and any inconsistency among them implies the occurrence of an error. Replicating the equipment or the software generates this kind of information redundancy.

Software replication consists in the generation of two copies of same the software processed sequentially by only one processor unit. A data $\alpha$ is represented by $\alpha = (\alpha_A, \alpha_B)$, where $\alpha_A$ is the representation of data $\alpha$ in the replica A and $\alpha_B$ is the representation of data $\alpha$ in the replica B. The consistency criteria between the data is achieved if $\alpha_A = \alpha_B$. This principle is similar to that applied in [08] and [09].

The software replication has the ability to detect errors that affect only one replica, or both replicas in a distinct manner.

Software replication can be compared with a coding process. Coding strengthens the data by representing it with more bits than the necessary. The two replicas run in the main processor.

There are two kinds of faults that must be avoided in a project of a system for safety-critical applications [10]:

5.1.1 Common Mode Fault

A common mode fault consists in only one fault that affects both replicas, and consequently both representations of a result. This can cause, in a direct way, an undetectable erroneous result, without a preliminary latency phase, because of the correlation that exists among the replicas. Both replicas share several resources that can be grouped as:
Processor and Bus: extremely used by both replicas for calculus and transfers [11].

Memory: all data are stored in memory – in this case ROM memory for instructions and RAM memory for data. Both representations of a result are stored simultaneously in memory, in distinct addresses. References [12] and [13] describes a methodology to evaluate errors in memories.

The correlation among the replicas can be classified in:

- **Value Correlation:** With identical values for both representations of a result, a single fault has a high chance of altering them in the same way and thus making the error undetectable.
- **Timing Correlation:** Both representations can be simultaneously manipulated and stored.
- **Place Correlation:** Both representations are handled and stored in the same place.
- **Structural Correlation or Instruction Correlation:** Both replicas are manipulated by the same code structure, the same instruction since they are functionally identical.

It is possible to deal with the common mode faults if one decorrelate the effects of a fault on both representations of a result and thus make the error detectable.

The generic error identified earlier, that is, the path and block configuration errors, are a consequence of these correlations.

### 5.1.2 Dormant Faults

A fault remains dormant while it does not produce an error. During the dormancy time, an accumulation with other faults can generate an undetectable fault.

The volatile variables little called upon, either because they are rarely used or because they change slowly (rarely refreshed) are particularly threatened by fault dormancy. In both cases, the variables remain in storage a long time without being manipulated. An initial fault affects one of the two replicas and freezes its value, the results remain correct since the consistency criterion is assured. Then, a second fault affects the other replica and freezes its value too, the result is still correct but can no longer change, it will remain frozen at the previous value and the error becomes undetectable.

### 5.2 Errors

Errors are generated from faults. An error is detectable if both representations of a result $\alpha$ do not agree. The discordance can occur if an error affects only one of the representations or both in different manner. An error is undetectable if both representations of a result $\alpha$ agree, but with an unexpected result.

The error is said latent when it occurs and until it can be detected by some consistency test. A latent error is characterized by its latency time, which represents the time the error has occurred until the time the error is handled.
Latency time is dangerous, because other errors can occur, resulting in an accumulation of errors, increasing the probability of occurrence of undetectable errors. The accumulation of errors has two distinct forms:

- **Error Propagation**, that is, the propagation of the detectable nature of the errors. A detectable error remains detectable despite the transformations carried out.

- **Error Compensation**, that is, the masking of the detectable nature of the errors. A detectable error gives rise to an undetectable error. The error becomes undetectable insofar as it affects both representations of a result in the same way.

The error detection method for replicated software is based on the principles of revitalization and diversification

5.2.1 **Revitalization**

In order to resolve the problems associated with fault dormancy, the variables are revitalized by adding the timestamp of the current cycle.

5.2.2 **Diversification**

An application with error detection constituted by two replicas of the software is efficient for detecting those errors, which affect a single replica or both replicas differently, but is inefficient, on the other hand, for detecting those errors, which affect both replicas in the same way.

This problem is solved by a diversification between the two replicas:

- Value Diversification: different values between replicas;
- Execution De-Synchronization: different times for program execution and variables storing;
- Address Diversification: different address location between replicas; and
- Structure Diversification: different code structures between replicas.

6 **Conclusion**

Comparing the monoprocessed architecture with the duplicated architecture, there are two basic advantages, that is its lower cost and higher reliability. The main focus is to apply the coding process in a monoprocessed architecture that provides similar level of safety.

The proposed architecture has the merit of increasing the fault cover factor, allowing a great increase in the detection methods. This fact gives more confidence in the use of this type of architecture, using only one main processor and a small and simple auxiliary processor.

The techniques employed for error detection also have an important role in the task, increasing the fault cover factor.

**References**


[02] International Electrotechnical Commission IEC61508 Functional safety of


