Redundancy for safety-compliant automotive & other devices

Deepak Negi, Neha Bagri, VIKAS AGARWAL - March 12, 2014

A system's quality, maintainability, reliability, availability, and safety give a measure of how well it performs in its environment. Functional safety is the overall safety of a system against likely operator errors, hardware failures, and environmental changes. That depends on the system or equipment operating correctly in response to its inputs, including the safe management of likely operator errors, hardware failures, and environmental changes. Freedom from unacceptable risk of physical injury or of damage to the health of people either directly or indirectly is the objective of functional safety.

Various standards have been evolved for ensuring the safety function performs to the design intent, including under conditions of incorrect operator input and failure modes. IEC61508 is one of the international functional safety standards applicable to all types of industries. It is titled Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES). ISO 26262 is the Functional Safety Standard derived from IEC 61508, applicable to the automobile industry, and is titled Road vehicles -- Functional safety. ISO 26262 defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

There are various design techniques which are followed in an SoC for making it “Safety Compliant”, which are described in the ISO 26262 standard. Redundancy, self-test, signal monitoring, voltage supply monitoring, and watchdog timers are few of the techniques which are used in safety compliant devices. Redundancy is the main ingredient for these devices. There are various ways in which redundancy is used in the automobile devices. Many safety compliant devices use lockstep, ECC, CRC, and checksum as redundancy techniques. This article gives an overview of the redundancy techniques used in SoCs in the form of hardware, software, information, and time redundancy.

1. Redundancy

Redundancy is the inclusion of extra components in addition to those being used for the critical functionality to increase the reliability and availability of the system. The redundancy can be added in the form of hardware (e.g., dual core lockstep), in the form of software, in the form of information (e.g., ECC added to memory), and time redundancy. In general, redundancy works on the MooN concept.

1.1 M-out-of-N System Concept

An M-out-of-N (MooN) system contains N identical components and works on the principle that if at
least M out of N components are functioning correctly, then the system is error-free. An example of this is Triple Modular Redundancy (TMR), which is actually a 2oo3 system. If at least two components (a majority) out of three are working correctly, then the system is considered functional.

MooN systems are used in hardware as well as software. In hardware, critical components are replicated and on the basis of majority voting, the decision is taken.

In software, a task is repeated several times and the results are compared. The final result is generated based on the majority.

1.2 Hardware Redundancy

In SoCs, the redundancy in hardware can be in many forms: Replication of cores performing safety critical tasks (a.k.a. Lockstep), Delayed lockstep (1oo1 system), Asymmetrical lockstep, and Triple-voting (2oo3 system).

1.2.1 Lockstep

In high-level safety compliant devices, the cores performing the safety-critical tasks are replicated and the applications operate both cores in lockstep mode, comparing the results to make sure the redundant processing yields identical results. If the identical result is not obtained then it is assumed that a fault has occurred.

In lockstep, the same set of inputs is sent to both the cores at the same time, which perform the same calculations in the same clock cycle. The results are compared regularly to detect when a failure, whether transient, intermittent, or permanent, has occurred. On a mismatch in the outputs, generally a fault is flagged and/or a restart is performed.
1.2.2 Delayed Lockstep

Delayed lockstep is the variant of lockstep in which the inputs of one core are delayed by N clock cycles, the outputs of the other core are delayed by same duration, and then the results are compared. With this approach, time diversity is obtained. Since one core would be performing the same operation after N clock cycles, the probability of a noise pulse hitting both cores and affecting their functionality in the same way would be much reduced.

Delayed lockstep configuration. The data fed to core-2 is delayed by 2 clock cycles. The reset of core-2 is also 2 cycles delayed. The output of core-1 is delayed by 2 clock cycles and then compared by the checker circuit.

1.2.3 Asymmetric Redundancy

In Asymmetric Redundancy, instead of replication of identical cores, diverse cores are used. The diverse and dedicated core is tightly coupled with the main core by means of an interface enabling a step-by-step comparison of internal and external results. The interface reduces complexity and shortens error detection latency. The main core executes the critical task, while the dedicated diverse core replicates just enough of the execution of the main core to ensure that failures are detected or to ensure the safe operation of the main core. The hardware diversity provides effective coverage for common cause failures and systematic failures. Different structures of cores will lead to different ways in which the cores react and better diagnostic coverage of the common cause failures, since the same type of fault occurring in both the cores will be less. No separate code is needed for the parallel path, and the dedicated core is smaller than the main one. Sometimes, the area difference with respect to main core can reach up to 50% or even more. The disadvantage of this approach is that it can require a detailed analysis to prove the diagnostic coverage.

1.2.4 Triple Voting

This type of redundancy is implemented at the bit level. The register bits being used in safety-critical tasks are replicated twice, and the output is generated based on majority voting logic. This is an
example of 2oo3 majority voting system where, if any one of the three bits fails, the other two will mask the fault. Assuming that it is highly unlikely that two or all bits can be corrupted at the same time, this technique can keep the system operational. Since the area penalty is 2x in this case, it requires thorough inspection of the design configuration bits to identify the safety-critical configuration information to avoid any unnecessary area overhead.

1.3 Software Redundancy

In software redundancy, the task is performed multiple times in the software. The task may be performed multiple times using different code (different algorithms used for same task), and then the comparison of results takes place. It will lead to improved diagnostic coverage.

1.3.1 Software diversified redundancy (one hardware channel)

Here, two diverse software implementations are used in the design (i.e., different algorithms are used for executing the same task in a single processing unit). The figure below explains the implementation. The primary path is responsible for the calculations that, if calculated erroneously, can cause a hazard. The redundant path is responsible for verifying the primary path’s calculations and taking action if a failure is detected. The redundant path is implemented using separate algorithm designs and code to provide for software diversity. Once both paths are complete, a comparison of the output data of the two redundant software implementations is carried out. Detected differences lead to a failure message.

Examples of algorithm diversity are: A+B=C versus C-B=A – one path using normal calculations and the other path using two's complement mathematics.

![Diagram showing primary and redundant paths with comparison and remedial action](image)

1.3.2 Reciprocal comparison by software in separate processing units

Here, two processing units exchange data (including results, intermediate results, and test data) reciprocally. A comparison of the data is carried out using software in each unit, and detected differences lead to a failure message. The figure below shows the implementation. In this case, the primary path and redundant path are executed in different processing units using different software algorithms. This approach allows for hardware and software diversity if different processor types are used as well as separate algorithm designs, code, and compilers.
1.3.3 Peripherals Replication

Replication of peripherals is not same as that of cores. The peripherals share the same inputs from the outside world, and the outputs of the peripherals are compared in software.

For example, multiple ADCs are used in SoCs. Input channels are replicated to most ADCs and their outputs can be compared in software.

1.4 Information Redundancy

Information redundancy is the redundant data added to the actual data for purposes of fault detection or correction when communicating the information over a noisy channel. A few information redundancy schemes are described below.

1.4.1 Parity Bit

A parity bit is added to a binary word, indicating whether the number of ones in the data is odd or even. Even parity is the scheme in which a ‘1’ is added to data if the data has odd number of ones. Odd parity is the scheme in which a ‘1’ is added to data if the data has even number of ones. For example, if the actual data was ‘1111 0000’ and an even parity bit is added at its end, then ‘1111 0000 0’ will be transmitted over the communication channel. At the receiver’s end, any odd number of flipped bits will be detected.

1.4.2 Checksums

A checksum is a value calculated over a data block using some function. It is transmitted along with the information data over the noisy channel. At the receiver end, the calculated checksum should match the received checksum. The error detection capability depends on the number of redundant bits, size of the data, and the algorithm used to generate the checksum. Parity, modular sum, and position dependent checksum are a few examples of checksums that can be used for error detection. The transmitter can be signaled to retransmit the data in case an error is detected.

1.4.3 Error correcting codes

Certain checksum functions are capable of not only detecting the errors, but also of correcting data without the data being retransmitted. These are called error correcting codes. These also have a limit up to which the error can be corrected. For example, Hamming codes can detect double bit
errors and correct single bit errors in the data. Golay codes can detect up to four bits of errors but correct three bit errors. BCH, Goppa, Reed-Solomon, Reed-Muller, and Hadamard are other error-correcting codes named after their inventors. Various such coding techniques can be studied as part of information and coding theory, an intriguing and vast branch of applied mathematics, electrical engineering, and computer science.

1.5 Time Redundancy

Time redundancy involves performing safety-critical tasks redundantly, spread over time. Because these redundant tasks are spread over time, it helps in eliminating transient faults.

1.5.1 Time redundancy over single hardware channel

The safety-critical tasks are performed multiple times on common hardware using the same software. The results of the multiple operations are compared, and if differences are found, relevant corrective operations are performed. The section “Software diversified redundancy (one hardware channel)”, described under Software Redundancy above, is an extended version of this type of redundancy.

1.5.2 Time redundancy over parallel hardware channels

Safety-critical tasks, when performed over parallel channels, but not all at the same time, help in creating redundancy over time. A transient fault will not affect the operations in same way even if the parallel channels are symmetric hardware. The section “Delayed Lockstep”, described under Hardware Redundancy above, is an example of this type of time redundancy.

2. Conclusion

There are many design techniques used in the automobile industry for achieving safety. The use of redundancy in one way or another is essential for achieving safety to ensure that the device is more reliable and robust upon occurrence of faults.

Hardware redundancy has the advantage of detecting the fault as early as possible at the cost of increased hardware. On the other hand, software redundancy might be useful when the system cost is the restriction. We can make use of both hardware and software redundancy along with time redundancy to make the system more robust.

It would not be wrong to say that: Redundancy is the key to safer automobile products.

3. References

1. ISO26262 Specs
2. Wikipedia – “IEC 61508”
4. Wikipedia – “Redundancy (information theory)”
5. Redundancy (engineering)

Also see:

- Safety & security architecture for automotive ICs
- Testing safety-critical automotive parts
- ISO26262 in automotive IC development: is it just a tick-box exercise, or does it induce
manufacturers to make safer components?