A Transient-Resilient System-on-a-Chip Architecture with Support for On-Chip and Off-Chip TMR

R. Obermaisser, H. Kraut and C. Salloum
Vienna University of Technology, Austria

Abstract

The ongoing technological advances in the semiconductor industry make Multi-Processor System-on-a-Chips (MPSoCs) more attractive, because uniprocessor solutions do not scale satisfactorily with increasing transistor counts. In conjunction with the increasing rates of transient faults in logic and memory associated with the continuous reduction of feature sizes, this situation creates the need for novel MPSoC architectures. This paper introduces such an architecture, which supports the integration of multiple, heterogeneous IP cores that are interconnected by a time-triggered Network-on-a-Chip (NoC). Through its inherent fault isolation and determinism, the proposed MPSoC provides the basis for fault tolerance using Triple Modular Redundancy (TMR). On-chip TMR improves the reliability of a MPSoC, e.g., by tolerating a transient fault in one of three replicated IP cores. Off-chip TMR with three MPSoCs can be used in the development of ultra-dependable applications (e.g., X-by-wire), where the reliability requirements exceed the reliability that is achievable using a single MPSoC. The paper quantifies the reliability benefits of the proposed MPSoC architecture by means of reliability modeling. These results demonstrate that the combination of on-chip and off-chip TMR contributes towards building more dependable distributed embedded real-time systems.

1 Introduction

Due to diminishing returns from uniprocessor optimizations, chip designers are facing the need for new computer architectures. According to Pollack's rule [12], the increase in performance of a uniprocessor is only about the square root of the increase in the number of devices, which implies that doubling the transistor count will lead to a performance improvement of about 40%.

Multi-core architectures in the form of Multi-Processor System-on-a-Chips (MPSoCs) are a solution to circumvent Pollack's rule. If an application can be partitioned into a set of nearly autonomous concurrent functions, then a nearly linear performance improvement could be achieved by assigning a dedicated processing element to each of these concurrent functions. MPSoCs combine multiple heterogeneous processing elements, which can be interconnected by a NoC. However, the key to high performance in multiprocessor systems-on-a-chip are applications with inherent parallelism. Fortunately, the inherent concurrency in a typical embedded application (e.g., automotive electronics, avionics) satisfies this requirement.

Another challenge in the development of new processors is the increasing importance of transient faults. The types and causes of failures for electronics have changed over the years. Failure analysis in recent years has revealed that permanent failures have been reduced by improvements in technology but due to the higher level of complexity and downsizing other failure classes have emerged. The tremendous improvements made by the IC industry with respect to permanent failure rates are extenuated by increasing transient failure rates for instance due to semiconductor process variations, shrinking geometries, and lower power voltages [26, 6]. These result in higher sensitivity to neutron and alpha particles, and consequently have an impact on dependability by increasing the transient failure rates [13]. These technological effects affect in particular upcoming generations of MPSoCs manufactured using Very Deep Sub-Micron (VDSM) processes [32].

Due to these technological constraints, there is a growing importance of self correcting intelligence embedded into MPSoCs specifically targeting transient faults. Solutions for self correcting intelligence have emerged at different abstraction levels. For example, in [14] chip-level redundancy through a redundantly threaded multiprocessor with recovery is presented. This solution provides fault tolerance by replicating an application into two communicating threads, namely a leading thread executing ahead of a so-called trailing thread. The execution of the trailing threads is delayed by a predefined number of instructions in order to enable the trailing thread to use memory load values and branch outcomes of the leading thread.

Another approach for tolerating transient faults at chip-level is time redundancy [3, 8, 27]. Detection and masking of transient faults occurs based on the comparison of the computational results that are gained through redundant computations performed in temporal succession. Disadvantages of time redundancy are a performance penalty and constraints on the duration of the transient pulse that can be detected and corrected.

The focus of this paper are distributed real-time systems, where the correct behavior depends not only on the logical results but also on the points in time at which these results are produced. This real-time requirement particularly applies to fault scenarios. Hence, the computer system has to be fail operational by continuing to provide the specified application services (e.g., engine control in a car) in the value and time domain despite the occurrence of transient faults.

Consequently, we have selected spatial redundancy using Triple Modular Redundancy (TMR) for the proposed transient-resilient MPSoC architecture. The temporal behavior of a Triple Modular Redundancy (TMR) configuration does not change as long as only one of the replicas is affected by a transient fault.

In contrast to other solutions employing TMR within a chip (e.g., LEON3FT [11]), we perform replication of complete IP cores instead of introducing spatial redundancy at transistor level or at the level of logic circuits. The decision of selecting TMR at the level of IP cores is motivated by the following reasons:

- *Superior time and energy efficiency.* The voting in a TMR configuration involves an inevitable overhead. For TMR at the level of small logic circuits, this overhead occurs whenever this logic circuit is used. For example, in [2] an overhead of up to 23% has been determined for voting in TMR in different types of arithmetic circuits. Therefore, the solution presented in this paper performs incoming voting at a coarser level. Voting occurs on the final computational results of IP cores and not on the intermediate computational results. For example, consider an IP core for FFT that is implemented in a TMR configuration. Only the final Fourier coefficients are voted upon and not the results of each intermediate computation. Hence, there is less timing overhead and energy consumption.
- *Ability to use standard libraries.* Voting at the level of IP cores allows to use standard IP core libraries. In contrast, the introduction of fault-tolerant logic circuits (e.g., fault tolerant adder [2]) would require developers to construct completely new IP cores.
- *Higher resilience against spatial proximity faults.* TMR at the level of IP cores enables designers to physically separate the replicas on the chip. Hence, the probability of correlated transient faults affecting multiple replicas is significantly diminished. According to [7, 33] the probability of particle-induced multi-bit errors and soft errors within the combinatorial logic is going to increase.
- *Foundation for design diversity and heterogeneity.* The proposed solution allows to exploit diverse replicas, i.e., different IP cores that provide the same services, while having been designed by different developers in order to reduce the probability of design faults. For example, a TMR configuration with three different implementations of a control algorithm would allow to tolerate a design fault restricted to one of the three replicas.

In this paper, we present a System-on-a-Chip (SoC) architecture that supports both on-chip and off-chip TMR. For on-chip TMR, the replicas are three IP cores that are interconnected by a NoC. Incoming voting occurs at the IP cores that receive the redundant messages from the replicas.

The major contributions of the paper are as follows:

- *SoC architecture with inherent support for transient-resilience through TMR.* The presented architecture provides the foundations for TMR (e.g., by being design for replica determinism [31]). Furthermore, generic IP cores for voting enable designers to rapidly construct TMR configurations.
- *Complementation of on-chip and off-chip TMR.* Redundancy only leads to reliable systems if the constituting replicas are reliable [25]. Hence, the effectiveness of off-chip TMR can be improved when increasing the reliability of individual chips using on-chip TMR. This complementarity of on-chip and off-chip TMR is quantitatively evaluated in the paper through reliability modeling.
- *Spatial redundancy at level of IP cores.* Raising the granularity for TMR from logic circuit to IP core level brings the above mentioned benefits, such as superior time and energy efficiency, the ability to use standard libraries, higher resilience against spatial proximity faults, and support for design diversity.

The paper is structured as follows. Section 2 gives an overview of the proposed SoC architecture. The realization of on-chip and off-chip TMR is the focus of Section 3. Section 4 describes a reliability model of the different TMR configuration. Using the Moebius tool, we quantitatively analyze the reliability of an individual SoC and of clusters with multiple SoCs. The results of this analysis are presented in Section 5.

2 Time-Triggered System-on-a-Chip Architecture

The central element of the presented SoC architecture is a time-triggered NoC that interconnects multiple, possibly heterogeneous IP blocks called micro components (see Figure 1). The SoC introduces a *trusted subsystem*, which ensures that a fault (e.g., a software fault) within the host of a micro component cannot lead to a violation of the micro component's temporal interface specification in a way that the communication between other micro components would be disrupted. For this reason, the trusted subsystem prevents a faulty host within a micro component from sending messages during the sending slots of any other micro component.

Furthermore, the time-triggered SoC architecture supports integrated resource management. For this purpose, dedicated architectural elements called the Trusted Network

Authority (TNA) and the Resource Management Authority (RMA) accept resource allocation requests from the micro components and reconfigure the SoC, e.g., by dynamically updating the time-triggered communication schedule of the NoC.

2.1 Micro Component

The introduced SoC can host multiple application subsystems (possibly of different criticality levels), each of which provides a part of the service of the overall system. An example of an application subsystem in the automotive domain would be a braking subsystem. A nearly autonomous and possibly heterogeneous Intellectual Property (IP)-block, which is used by a particular application subsystem is denoted as a *micro component*. A micro component is a self-contained computing element, e.g., implemented as a general purpose processor or as special purpose hardware. An application subsystem can be realized on a single micro component or by using a group of possibly heterogeneous micro components (either on one or multiple interconnected SoCs).

The interaction between the micro components of an application subsystem occurs solely through the exchange of messages on the time-triggered NoC. Each micro component is encapsulated, i.e., the behavior of a micro component can neither disrupt the computations nor the communication performed by other micro components. Encapsulation prevents by design temporal interference (e.g., delaying messages or computations in another micro component) and spatial interference (e.g., overwriting a message produced by another micro component). The only manner, in which a faulty micro component can affect other micro components, is by providing faulty input to other micro components of the application subsystem via the sent messages.

Due to encapsulation, the SoC architecture supports the detection and masking of such a failure of a micro component using TMR. Encapsulation is necessary for ensuring the independence of the replicas. Otherwise, a faulty micro component could disrupt communication or communication of the replicas, thus causing common mode failures.

Also, encapsulation is of particular importance for the implementation of SoCs encompassing application subsystems of different criticality levels. In such a mixed criticality system, a failure of micro components of a non safety-critical application subsystem must not cause the failure of application subsystems of higher criticality.

For the purpose of encapsulation, a micro component comprises two parts: a *host* and a so-called *Trusted Interface Subsystem (TISS)*. The host implements the application services. Using the TISS, the time-triggered SoC architecture provides a dedicated architectural element that protects the access to the time-triggered NoC.

2.2 Requirements for the Time-Triggered Network-on-a-Chip

The time-triggered NoC interconnects the micro components of an SoC. The purposes of the time-triggered NoC encompass clock synchronization for the establishment of a global time base, as well as the predictable transport of periodic and sporadic messages.

Clock Synchronization The time-triggered NoC performs clock synchronization in order to provide a global time base for all micro components despite the existence of multiple clock domains. The time-triggered NoC is based on a uniform time format for all configurations, which has been standardized by the OMG in the smart transducer interface standard [28].

Predictable Transport of Messages Using Time-Division Multiple Access (TDMA), the available bandwidth of the NoC is divided into periodic conflict-free sending slots. We distinguish between two utilizations of a periodic time-triggered sending slot by a micro component. A sending slot can be used for the *periodic transmission of messages* or the *sporadic transmission of messages*. In the latter case, a message is only sent if the sender must transmit a new event to the receiver. If no event occurs at the sender, the reserved bandwidth remains available.

The allocation of sending slots to micro components occurs using a communication primitive called *pulsed data stream* [16]. A pulsed data stream is a time-triggered periodic unidirectional data stream that transports data in pulses with a defined length from *one* sender to *n* a priori identified receivers at a specified phase of every cycle of a periodic control system.

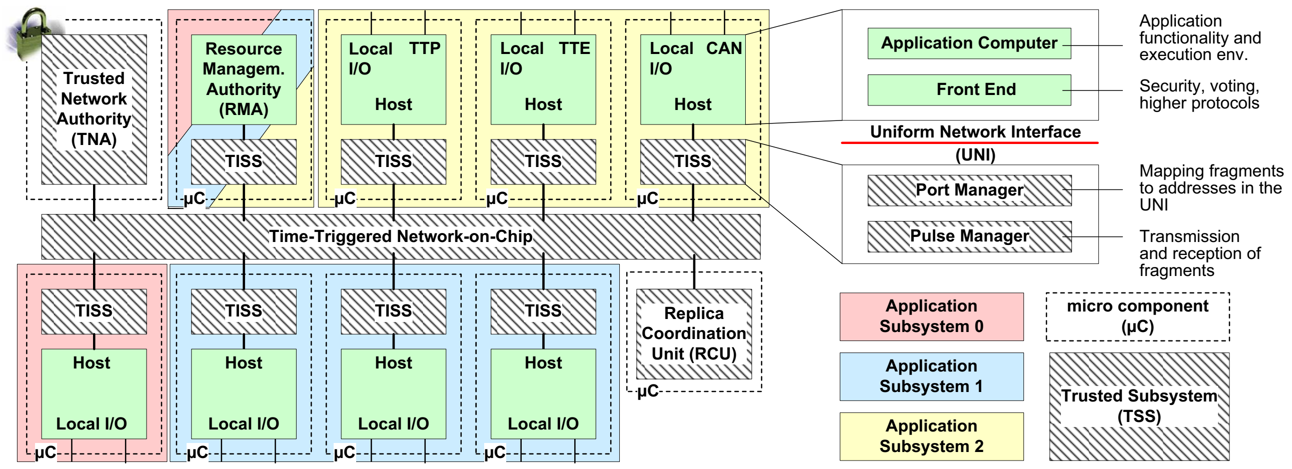
A pulse consists out of one or more *fragments* of variable size, which are sent successively during the duration of a single pulse. The defining parameters of a pulsed data stream (i.e., pulse period, pulse phase) determine the allocation of TDMA slots to the micro component that sends the pulsed data stream. TDMA slots are allocated to the sender micro component in a time interval, which periodically recurs with the pulse period and has a length controlled by the pulse duration.

2.3 Host

The host performs the computations that are required to deliver the intended application service of a micro component. In general, the host is not assumed to be free of design faults. The host is divided into an *application computer*, which implements the intended application services and the so-called *front end*. The front-end provides a domain-specific network interface to the application computer (e.g., temporal firewall interface [24] or event queues). It can incorporate middleware bricks that extent the communication services that are provided by the TISS. Examples of such services are security functionality (e.g., encryption, authentication) or fault-tolerance mechanisms (e.g., voting).

2.4 Trusted Interface Subsystem

The TISS manages the host's access to the underlying NoC. Each TISS contains a table which stores a priori knowledge concerning the global points in time of all message receptions and transmissions of the respective micro



Trusted Network Authority (TNA)
Resource Managem. Authority (RMA)
Local I/O
TTP
TTE
CAN
Host
TISS
µC
Time-Triggered Network-on-Chip
Replica Coordination Unit (RCU)
Application Computer
Application functionality and execution env.
Front End
Security, voting, higher protocols
Uniform Network Interface (UNI)
Port Manager
Mapping fragments to addresses in the UNI
Pulse Manager
Transmission and reception of fragments
Application Subsystem 0
Application Subsystem 1
Application Subsystem 2
micro component (µC)
Trusted Subsystem (TSS)

Figure 1. Structure of Time-Triggered SoC Architecture

component. Since the table cannot be modified by the host, a design fault or a hardware fault restricted to the host of a micro component cannot affect the exchange of messages by other micro components.

As depicted in Figure 1, the TISS is structured into two parts, namely a port manager and a pulse manager. The pulse manager is responsible for transmitting fragments of pulses, while providing media access control and logical link control. The port manager maps each fragment that is received or sent by the pulse manager to a corresponding address in the uniform network interface.

Port Manager The port manager implements the transport layer in the OSI reference model [17]. The port manager maps each fragment that is received or sent by the pulse manager to a corresponding address in the uniform network interface. Furthermore, the port manager manages the control fields of the ports (e.g., write and read position of event ports, synchronization fields for state ports). In addition to the ports, the port manager provides a programmable timer-interrupt service, a watchdog service and a power control service.

Pulse Manager The pulse manager accesses the TTNoC by sending and receiving single fragments of a pulsed data stream according to the message definitions stored in the MEDL. Whenever a fragment is received by the pulse manager the fragment number which identifies the fragment within a message, the associated port number and the fragment itself are handed over to the port manager. On the other hand, when a fragment is scheduled to be transmitted by the pulse manager, the pulse manager issues a request to the port manager which includes the fragment number and the port number of the requested fragment. The port manager will then pass the requested fragment to the pulse manager.

2.5 Architectural Elements for Resource Management

The purpose of the integrated resource management in the SoC architecture is to dynamically assign computational resources (i.e., micro components) to application subsystems and to grant communication resources and power to the individual micro components.

We distinguish two fundamentally different types of application subsystems: *Safety-critical applications subsystem* need to be certified to the highest criticality classes (e.g., class A according to DO-178B). Non safety-critical applications subsystems, on the other hand, do not require certification to the highest criticality classes. In general, these two types of applications subsystems will involve fundamentally different design paradigms. The focus of safety-critical applications lies on simplicity and determinism in order to facilitate thorough verification and validation. In contrast, non safety-critical applications can provide more complex application services (e.g., need to deal with insufficient a priori knowledge about the environment) and dynamism to handle the challenges of evolving application scenarios and changing environments.

The architectural elements for resource management follow this bivalent distinction of application subsystems. We provide two different architectural elements for enabling integrated resource management, namely the Trusted Network Authority (TNA) and the Resource Management Authority (RMA). The RMA computes new resource allocations for the non safety-critical application subsystems, while the TNA ensures that the new resource allocations have no adverse effect on the behavior of the safety-critical application subsystems. As depicted in Figure 1 the TNA is part of the trusted subsystem of the SoC, whereas the RMA is not. By splitting the entire resource management into two separate parts, where only one is part of the trusted subsystem, the certification of the time-triggered SoC is significantly simplified, since the checking of the correctness of a resource allocation through the TNA is significantly simpler than its generation at the RMA.

2.6 Replica Coordination Unit

The purpose of the Replica Coordination Unit (RCU) is the detection of host failures caused by transient faults and the subsequent reseting of the faulty hosts. The reseting of hosts through the RCU is part of the architectural mechanisms for recovering from transient faults. The RCU detects host failures by comparing the redundant computational results in TMR configurations. In case a divergence of these results occurs, the RCU executes a reset operation at the TISS of the respective micro component.

2.7 Gateways

The proposed SoC architecture supports gateways for accessing chip-external networks (e.g., TTP [23] or TTE [22] in Figure 1). The benefits of gateways include the ability to interconnect multiple SoCs to a distributed system, which enables applications based on the SoC architecture for ultra-dependable systems [34]. Since component failure rates are usually in the order of 10^{-5} to 10^{-6} (e.g., [29] uses a large statistical basis and reports 100 to 500 failures out of 1 Million Electronic Control Units (ECUs) in 10 years), ultra-dependable applications require the system as a whole to be more reliable than any one of its components. This can only be achieved by utilizing fault-tolerant strategies that enable the continued operation of the system in the presence of component failures.

In case the chip-external network is also time-triggered (e.g., TTP [23], TTE [22]), the TDMA scheme of the NoC can be synchronized with the TDMA scheme of the chip-external network. The periods and phases of the relayed pulsed data streams on the NoC can be aligned with the transmission start instants of the messages on the time-triggered chip-external network. Consequently, a message that is sent on the chip-external network is delivered to the micro components within a bounded delay with minimum jitter (only depending on the granularity of the global time base). The alignment between pulsed data streams and messages on time-triggered networks ensures that replicated SoCs perceive a message at the same time, i.e., within the same inactivity interval of the global sparse time base [20]. This property is significant for achieving replica determinism [30] as required for active redundancy based on exact voting. Without synchronization between the NoC and the chip-external network, there could always occur a scenario in which one SoC forwards the message to the micro components in one period of the pulsed data stream, while another SoC would forward the message in the next period.

3 Fault Tolerance

The Time-Triggered System-on-a-Chip (TTSoC) architecture employs micro components in TMR configurations in order to tolerate transient faults. We denote three replicas in a TMR configuration a Fault-Tolerant Unit (FTU). In addition, we consider the voter at the input of a micro component and the micro component itself as a self-contained unit, which receives the replicated inputs and performs voting by itself without relying on an external voter. We call this behavior *incoming voting*.

With respect to voting, one can differentiate between two kinds of strategies: exact voting and inexact voting [21]. Exact-voting is preferred in the TTSoC architecture, because it is transparent to applications and generic (i.e., independent from any specific application service). The underlying assumption of exact voting is, that the replicas show *replica-deterministic* behavior [31]. Replica determinism requires that all correct replicas produce exactly the same output messages that are at most an interval of d time units apart (as seen by an omniscient outside observer). In a time-triggered system, the replicas are considered to be replica deterministic, if they produce the same output messages at the same global ticks [21]. A common source of replica indeterminism is in an inconsistent message reception order at the replicas within an FTU.

A TMR-based FTU is a fail-operational component, which means that it continues to deliver a correct service despite the failure of a single replica. Nevertheless, the reliability of an FTU is reduced, if a failed replica remains in an erroneous state. In fact, the reliability of a TMR-based FTU in which one of the three replicas has already failed is actually lower than the reliability of a single replica since the FTU requires both remaining replicas to stay operational. In order to obtain the original reliability of an FTU after the failure of a replica, one of the following actions can be taken:

- In case of a transient fault where the hardware is still operational, a valid state in the replica can be reestablished by adequate recovery mechanisms.
- In case of a permanent hardware fault, the hardware of the replica can be repaired or replaced during the next scheduled maintenance service.
- In case of a permanent hardware fault where a repair action cannot be taken due to temporal or physical constraints (e.g., a faulty IP core within a chip cannot be repaired), the role of the failed replica can be taken over by a spare IP core.

In the following, we will explain TMR-based FTUs at the on-chip and off-chip level. This description will focus on the following categories:

- *Encapsulation:* Common mode failures have to be avoided by *encapsulation* mechanisms that establish a dedicated Fault Containment Region (FCR) for each replica. *An FCR is a collection of components that operates correctly regardless of any arbitrary logical or electrical fault outside the region [19].* Without encapsulation it cannot be guaranteed that the replicas fail independently. If a shared communication medium is employed, *error containment* mechanisms (e.g., the bus guardian in TTP [23]) are needed to ensure that a fault in a replica cannot disrupt the communication of the other replicas.
- *Replica determinism* has to be supported by the architecture to ensure that the replicas of an FTU observe the received messages in a consistent order.
- *Temporal predictability:* The communication service has to be *temporally predictable* in order to guarantee

that the resulting system can meet its deadlines even in peak-load scenarios.

- *Recovery and repair.* The architecture should provide adequate *recovery* and *repair* mechanisms, in order to reestablish the original reliability of an FTU after the failure of a replica.

3.1 On-chip TMR

The purpose of on-chip TMR is to increase the reliability of services residing on a single chip. For ultra-high reliability, however, off-chip TMR has to be employed (cf. Section 3.2). Therefore, FTUs are constructed by mapping the replicas to individual hosts which are interconnected by the time-triggered NoC (see Figure 2(a)). The following paragraphs describe how the required architectural properties for TMR are met w.r.t. on-chip TMR.

Encapsulation The FCRs for *physical faults* are the individual hosts and the Trusted Subsystem (TSS) which consists of the NoC, the TISSs and the TNA. Contrary to an operating system that provides dedicated FCRs for multiple tasks on a single processor by complex memory protection mechanisms and preemptive scheduling strategies, the TTSoC architecture naturally provides fault containment by the physical separation of the individual hosts and the TSS.

The independence of the host-FCRs is guaranteed by the fact that hosts can interact with each other exclusively via the exchange of messages over the NoC. There are no other hidden channels (e.g., shared memory) through which a host can interfere with any other host.

The independence of the TSS-FCR is guaranteed by the fact that the hosts have no possibility to directly interfere with the operation of the TSS. The TSS transports messages from one host to another host according to a predefined time-triggered message schedule. This schedule can be exclusively configured by the TNA which is itself part of the TSS. Thus, it is guaranteed that a faulty host cannot disrupt the communication among other hosts.

Contrary to fault-tolerant off-chip networks like TTP [23] which are partitioned into multiple FCRs, the TSS is a single atomic FCR, which means that a failure of one of its elements (i.e., the NoC, the TNA or one of the TISSs) can potentially cause a failure of the entire chip. Since in a typical SoC, the die area consumed by the TSS is relatively small compared to the area consumed by the rest of the chip, we expect the failure rate of the TSS to be relatively low (cf. Section 5). Therefore, for on-chip TMR we assume that the TSS does not fail during the mission time of the system.

To conclude, the underlying assumptions for on-chip TMR with respect to physical faults, are that the hosts fail independently and that the TSS does not fail during the system's mission time. Considering the fact, that a single chip is susceptible to common mode failures caused by disruption of the single power supply, particle induced multi-bit errors, extensive EMI disturbances or physical damage, the assumption coverage for these assumptions will not satisfy the requirements for ultra-high dependable systems. Nevertheless, for many applications with less stringent dependability requirements, on-chip TMR can be a cost-effective alternative to increase the reliability of systems realized on a single SoC. Furthermore, in safety-critical systems on-chip TMR can be used in conjunction with off-chip TMR to further improve reliability.

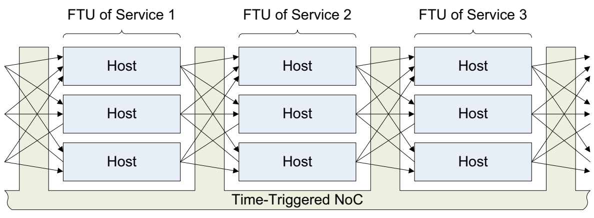
With respect to *design faults*, a host constitutes an FCR as long as no piece of the design is used in any other host. If pieces of a design (e.g., library functions or IP cores) are used in a set of hosts, the entire set has to be considered as a single atomic FCR. The TSS is assumed to be free of design faults. It has to be certified at least to the same criticality level as the most critical host in the entire SoC.

Replica determinism The TTSoC architecture supports different topologies with respect to the NoC which range from simple shared buses to complex mesh structures supporting multiple channels and concurrent message transfer. In advanced topologies the paths between different sender and receiver pairs can have different length (i.e. they can include a different number of hops), and a message on a short path can be received before a message that has been sent earlier, but over a longer path. Furthermore, multi-cast communication can be temporally asymmetric since also the paths from a single sender to the individual receivers may have different length. Therefore, messages are potentially received in an inconsistent order.

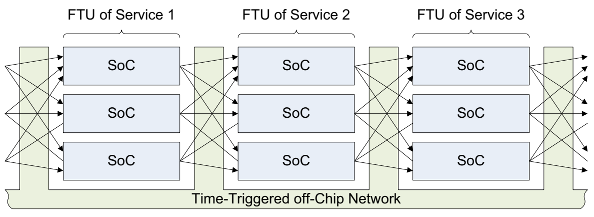
In the TTSoC architecture, replica determinism is established by exploiting the global time base in conjunction with time-triggered communication and computational schedules. Computational activities are triggered after the last message of a set of input messages has been received by all replicas of an FTU. This instant is a priori known due to the predefined time-triggered schedules. Thus, each replica wakes up at the same global tick, and operates on the same set of input messages. The messages in this set are treated as if they would have been received simultaneously, which means that a potential inconsistent reception order of the messages in this set is irrelevant.

Temporal predictability The predefined message schedule of the NoC assures that each micro component can use its guaranteed reserved bandwidth independently of the communication activities of the other micro components. Furthermore, the concept of a *pulsed data stream* supports the reservation of a defined bandwidth within a periodically recuring interval. The recurring intervals are called the pulses of a pulsed data stream. This concept fits perfectly to TMR in a time-triggered system. A replica of a typical FTU will periodically read the replicated inputs, perform incoming voting, do the application specific processing on the voted input data and send its output value to the next FTU.

The reactivness of the overall system can be optimized if the execution of the FTUs and the transmission of the mes-



(a) On-Chip TMR



(b) Off-Chip TMR

Figure 2. Fault Tolerant Units

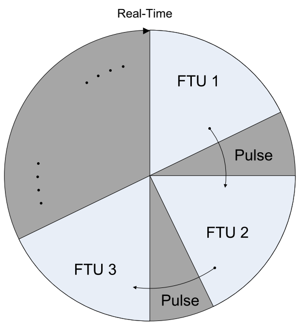


Figure 3. Voting on a Circular Time Model

sages are temporally aligned. Figure 3 depicts the execution of three FTUs in the periodic time model. The individual replicas of an FTU can execute in parallel since they reside on dedicated micro components. The pulses of the pulsed data streams are perfectly aligned with the execution of the replicas, and thus increase the reactivness of the system by minimizing the end-to-end latency from the first to the last FTU in the chain.

In addition to minimizing the end-to-end latency, the concept of a pulsed data stream increases the resource efficiency since the communication bandwidth is only reserved fore those intervals in which its actually needed.

Recovery and Reintegration For on-chip TMR, the RCU – being a trusted component – coordinates the recovery actions of an FTU. Therefore, the messages of all replicas of an FTU are routed to the RCU where they are compared to each other. If one replica deviates from the other two replicas, the RCU sends to the TNA a *restart-request message* for the host on which the erroneous replica resides. Since the TNA has direct access to the TISSs and the TISSs controll the reset lines of the attached hosts, the TNA can restart the corresponding host.

After the restart of a replica, the replica has to build up a valid internal state which has to be in perfect synchrony to state in the other replicas of the FTU. To facilitate state recovery, each replica periodically sends out its internal state via a so-called *history state message*. With the same period, each replica votes over the three history state messages (the own history state message and messages of the other two replicas) and overwrites its internal state with the voted result at the same global tick of its local clock. We call these periodic global ticks, the *reintegration points* of an FTU. Since the state of a replica after a reintegration point is exclusively determined by its inputs (data inputs plus the history state messages), a replica can be considered as stateless at the reintegration point. Therefore, a restarted component has simply to wait for the next reintegration point, to reach a consistent state.

If the replica has failed due to a transient fault, a restart of the corresponding host will be sufficient to reintegrate the replica. In order to detect a permanent or intermittent faults, the RCU can make use of threshold schemes like the α-count [4]. Therefore, the RCU holds an failure counter for each replica, which is increased every time the replica deviates from the other replicas in an FCR, and which is decreased as time goes on. If the counter reaches a defined threshold, the host on which the replica resides is considered to be permanent faulty. In this case, the RCU can send a reconfiguration request to the TNA to remove the faulty host from the FTU and replace it by a spare host. The spare host has to hold the images of all the replicas for which it should potentially act as a substitute.

3.2 Off-chip TMR

To achieve ultra-high dependability, the TTSoC architecture supports the construction of FTUs where the individual replicas are mapped on the hosts of *distinct* SoCs which are interconnected by a fault-tolerant off-chip network like TTP [23], FlexRay [9] or Time-triggered Ethernet [22] (see Figure 2(b)). Thus, the SoCs form network nodes of a fault tolerant distributed system. Since each replica of an FTU is located on a distinct network node, an FTU will still stay operational despite the failure of an entire node. We will now show how the required architectural properties for TMR are met with respect to off-chip TMR.

Encapsulation In ultra-high dependable systems common mode failures have to be considered that can cause an entire chip to fail. Examples are disturbances in the power supply, particle induced multi-bit errors, extensive EMI disturbances, or physical damage of the chip. Thus, a single host in an SoC can no more be regarded as an FCR.

For off-chip TMR we consider an entire SoC as an FCR for physical faults. The coverage of the assumption that the nodes in a distributed system fail independently is much higher than for hosts within a single SoC. Contrary to host in an SoC, the network nodes of a distributed system do not reside on the same die nor in the same package. They can be physically separated over large distances (e.g., on the opposite sides of a car or an airplane), and can have individual power supplies.

The TTSoC architecture requires that faulty nodes cannot interfere with the correct operation of the off-chip network. Furthermore, the off-chip network has to be able to tolerate internal faults in order to meet the requirements for ultra-high dependability. Therefore, it has to be partitioned into multiple FCRs, which are integrated in a way that the network can deliver a correct communication service despite of a failure of a single internal FCR.

An example of an ultra-high dependable network that meets these requirements is TTP. TTP provides error containment via local or centralized bus guardians which electronically connect each node only during its specified time slot to the shared communication bus. Thus, a node which is violating its temporal specification cannot disrupt the communication among the other nodes. Furthermore, TTP was constructed to tolerate any arbitrary single fault within the network itself, by providing two redundant communication channels (with dedicated bus guardians) forming independent FCRs.

As for on-chip TMR, we consider a host as an FCR for design faults. If pieces of a design are used in a set hosts, the entire set has to be considered as a single atomic FCR. The off-chip network is considered to be free of design faults (e.g., TTP is certified for the usage in ultra-high dependable systems).

Replica determinism As described in section 3.1, replica determinism in the TTSoC architecture is based on a consistent view of the global time. Time-triggered networks like FlexRay, TTP or TTE provide fault-tolerant clock synchronization to establish a system-wide global time. To facilitate the temporal coordination of hosts residing on different nodes, the *chip-wide* global time in each SoC is synchronized to the *system-wide* global time established by the off-chip network.

Temporal predictability Temporal predictability is provided by the combination of the on-chip and the off-chip time-triggered network. Due to the fact that the time in the NoC is synchronized to the system-wide time of the off-chip network, the message schedule of both networks can be aligned which makes the exchange of messages at the on-chip/off-chip gateways temporally predictable.

Recovery and Reintegration For off-chip TMR there is no central unit that can trigger a recovery action of replicas that are distributed across multiple SoCs. The restart or the migration of a replica can be exclusively triggered by the RCU within the SoC on which the replica resides. In order to enable recovery for off-chip TMR, the output messages off all three replicas of a distributed FTU have to be routed to the three RCUs of the SoCs where the replicas reside. By comparing the output messages, a RCU can decide whether the local replica functions correctly or whether it is affected by a transient or permanent fault. In case of transient fault the replica can be restarted, and in case of a permanent fault of the corresponding host, the replica can be migrated to a spare host.

Thus, from a global point of view, an SoC in a distributed system, is a self-checking component in which recovery actions are exclusively triggered by the local RCU. The limitation of this approach is, that the recovery actions can only be taken as long as the TSS in the SoC is still operational. The advantage is that a central coordination unit for recovery is not required, which would constitute a single point of failure and would not be acceptable for ultra-high dependable systems.

4 Reliability Modeling

This section describes the model and the evaluation approach for the quantitative reliability assessment of a single SoC, as well as for different TMR implementations. The model takes into account the increasing importance of transient faults by Single Event Upsetss (SEUs) due to shrinking semiconductor geometries and lower power voltages. In addition, the reliability assessments focus on the consequences of design faults in the context of replicated application computers and their design fault correlation.

For the evaluation of the TTSoC architecture, the Mobius dependability modeling tool [5] has been used. Mobius is a software tool for modeling the performance and dependability of complex systems. It uses an integrated multi-formalism, multi-solution approach, i.e., the overall model can be divided into smaller pieces and treated with different model formalisms and solution techniques. The classical modeling work is done in atomic and composed models. Atomic models are the basic modeling elements which consist of states, state transitions and parameters. Composed models are used to form more complex models by assembling atomic- or other composed models.

4.1 Chip-Model

4.1.1 Fault Model of the TTSoC

The fault model for a SoC component is partitioned into a design fault model and a physical fault model. The design fault model comprises hardware and software design faults and the physical fault model describes faults caused by physical exposures. Because of the limited independence of IP cores within an SoC, the conventional approach for a fault hypothesis would be the consideration of the complete chip as a single FCR for physical- and design faults, as outlined in [21]. In our case, justified by the distributed, fault-tolerant architecture of the TTSoC architecture, our fault hypothesis exhibits a more detailed structuring of FCRs. As a result the probability for common mode failures is higher than the probability when considering a complete SoC as an FCR.

Design fault model The design fault model contains the following FCRs:

- **Replicated Host**: This FCR includes the application computer and the front end. In case of replication without diversity, all replicas (possibly on different SoCs) comprise the same FCR.
- **TSS**: This FCR consists of the RCU, the NoC, the TNA and the TISSs. The TSS is assumed to be free of design faults. The simplicity of the TSS's design in the TTSoC architecture facilitates a thorough validation (e.g., by means of formal verification) which substantiates this assumption.
- **RMA**: The RMA forms its own FCR, because the RMA will usually be developed by different vendors than the TSS and exhibits a lower rigidity in the development process. Due to the high complexity of the RMA which has to dynamically compute time-triggered communication schedules from the reconfiguration requests of the application, validation to the highest criticality levels, e.g., class A according to DO-178B will generally be infeasible. The potential residue of design faults in the RMA is taken into account by means of a resource allocation protection by the TNA. The TNA protects safety-critical application subsystems from being disturbed by reconfiguration actions by a faulty RMA. Thus, only non safety-critical application subsystems can be affected by an RMA failure.

The corresponding failure rates can be estimated by means of the desired Safety Integrity Levels (SILs) (according to the IEC61508 standard) per FCR. When using design diversity each application computer can be considered as own FCR. The according assumption coverage is bounded by the design fault correlation which is determined by the usage of common resources like development tools, compiler, specification representation etc. ([1]). Figure 5(b) summarizes the design failure rates.

Physical fault model The physical fault model contains the following FCRs:

- **Host**: For physical faults, a host in conjunction with the port manager constitutes an FCR. Neither the port manager nor the host can interfere with the correct operation of the pulse manager or the NoC.
- **Pulse managers + TNA + RCU + NoC:** The pulse manager is the only part of a micro component which has direct access to the NoC. So a failure of one of the pulse managers can cause an overall SoC failure. Also the TNA can cause a common mode failure and therefore its reasonable to combine all pulse managers and the TNA together into one FCR.
- **RMA:** Since the resources of safety-critical application subsystems are protected by the TNA, a fault in the RMA can affect exclusively non safety-critical application subsystems.

We estimated the transient failure rates over the resource usage (ram, flip flops) of a TTSoC prototype implementation on an Altera Cyclone II Field Programmable Gate Array (FPGA). The method is described in [10] and the necessary SEU radiation tests can be found in [18]. Figure 5(a) summarizes the physical failure rates.

Repair and recovery durations In the fault model we distinguish between repair and recovery in that way that repair removes permanent faults and is executed by an external force while recovery happens autonomously after transient faults. We got concrete values for the durations from typical values of the aviation and the automotive domain: For the Mean Time To Repair (MTTR) we took the typical airplane mission time of 10 hours and for the recovery time we took the maximum allowed activator freezing time of 50 ms for automotive systems [15].

4.1.2 Mobius Model of the TTSoC

To describe the TTSoC system two different model formalisms are combined: the Rep & Join formalism for an abstract (composed) view of the SoC and the SAN formalism to describe the sub-models in detail. Generally, the SoC model is a composition of one atomic SoC-infrastructure model (describes RMA, RCU, TNA), a gateway model (models the interconnection to an off-chip network) and an application-dependent number of micro component models. Within the composed model the atomic models share informations by means of shared state variables which distribute,e.g., the state of the TSS or the number of failed hosts etc..

Atomic Model of SoC RMA, RCU and TNA A collective model describes the failures and recoveries of TNA, RMA and RCU. In case of a TNA failure the whole SoC is considered to be failed. When a RMA failure happens all non safety-critical application subsystems are considered to fail too. A RCU failure causes a failure of all safety-critical application subsystems which are part of the FTU.

Atomic Model of a Micro Component The SAN model captures a replicated micro component used for TMR. For this purpose the model share a state variable to count the number of failed replicas and a state variable to set or reset the state of the assigned Distributed Application Subsystem (DAS).

The model covers the following events:

- Physical faults in the TISS–pulse manager: A physical fault in the pulse manager results in a NoC failure and therefore to an overall SoC failure. Therefore a TSS failure is communicated to all micro component model instances which logical belong to the same SoC.
- Physical faults in host and TISS–port manager: Cause a failure in the value domain and therefore only concern the micro component model.

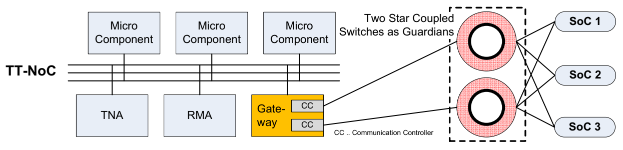


Figure 4. Fault tolerant SoC Cluster

- Application computer design faults: Determined by the coverage of the design diversity, a design fault causes either the failure of one or the simultaneous failure of all replicated micro components. Within the model this circumstance is modeled by a transition which possesses two cases, one with the probability of the design-fault-correlation and one with the probability 1 - design-fault-correlation.

- TSS failures caused by the RCU+RMA+TNA model or other micro component models: The model gets the TSS state by a state variable shared with all other models logical belonging to the same SoC.

- Gateway failures: The model gets the state of the off-chip interconnection by a state variable shared with the Gateway model.

4.2 Model of Multiple SoCs with Gateways

For our distributed model the interconnection takes place over a fault tolerant star-coupled time-triggered network (e.g., TTP [23] or TTE [22]), see Figure 4. Switches are enclosed from Guardians which protect the whole network from temporal domain switch failures. Each Guardian has exact knowledge about the sending times of each SoC and acts as temporal firewall in the same way as the TISS for the Network-on-Chip does. The connection between a SoC and the cluster network is made by the gateway micro component (see Figure 4). The gateway consists of a TISS, a gateway host as intelligent mediator between intra and inter-SoC network and two redundant communication controllers.

The gateway fault model consists of the following FCRs:

- **Switch + Guardian:** Physically the Guardian is a part of the switch and in this way together they form a FCR.

- **Physical Link:** A physical link is a bidirectional physical connection between a SoC and a Switch. Because the two pysical links to the redundant off–chip network are spatial close together the assumption coverage is bounded by the probability of correlated channel failures (e.g. caused by Electro Magnetic Interference (EMI)).

- **Communication Controller:** A communication controller is part of the gateway micro component.

- **Gateway Application Computer:** A gateway application computer has the purpose of mediation between the NoC and the off-chip network.

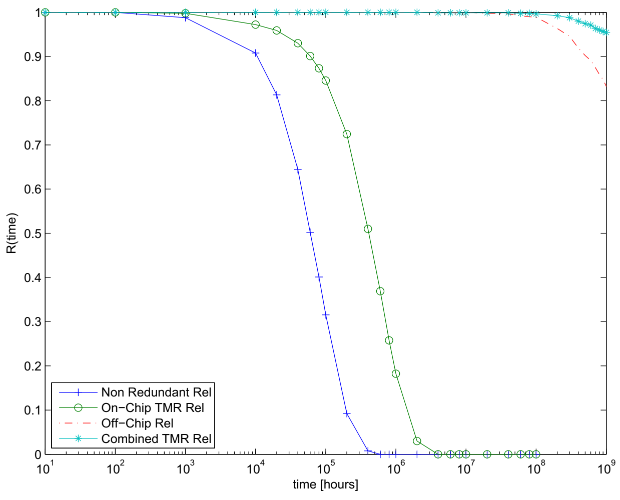


Figure 6. Reliabilities of TMR *(application computer failure rate of 10000 FIT)*

The failure rates for communication controller and gateway application computer were estimated over their resource usages as mentioned for the SoC fault model. The physical link failure rate was taken from typical EMI rates and the failure rate for the switch was taken from typical assumptions about electrical devices (see table 5(a)). For the repair and recovery durations the same assumptions as in the chip fault model were used.

5 Results

This section compares the simulation results of four different system configuration: non redundant, on-chip TMR, off-chip TMR, and combined on-chip and off-chip TMR. Figures 5(a) and 5(b) depict the model parameters, while Figures 7 and 6 show the resulting Mean Time To Failures (MTTFs) and reliabilities for the different TMR approaches.

5.1 Comparison of On-chip, Off-Chip and Combined On-chip/Off-Chip TMR

As can be seen in Figures 7 and 6, on-chip TMR clearly outperforms a non-redundant solution for host failure rates of 500 FIT or worse. In case of host failure rates better than 500 FIT, the failure rate of the TSS is dominant and undermines potential reliability gains through active replication of hosts. Also, in case of low host failure rates compared to the failure rate of the TSS, on-chip TMR does not contribute towards a better reliability, because each additional TISSs worsens the reliability of the TSS. In Figures 7 and 6, even a degradation of reliability can be observed for host failure rates smaller than 100 FIT. Nevertheless, the failure rate of the TSS is assumed to be much lower than the failure rate of a host due to the smaller area consumption. This assumption has been confirmed by first prototype implementations of the TTSoC architecture.

Figures 7 and 6 also demonstrate that off-chip TMR is the basis for supporting ultra-dependable applications,

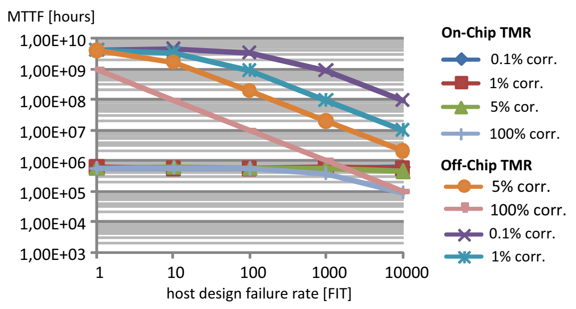
FCR	Transient Failure Rate	Permanent Failure Rate
Host	10^4 FIT	1 FIT
TISS-Port manager	170 FIT	0.02 FIT
TISS-Pulse manager	267 FIT	0.03 FIT
RMA	404 FIT	0.04 FIT
TNA	144 FIT	0.02 FIT
Switch	1000 FIT	0.1 FIT
Gateway Host	100 FIT	0.01 FIT
Communication Controller	700 FIT	0.07 FIT
Physical Link	10^4 FIT	1000 FIT

(a) Physical Fault Model

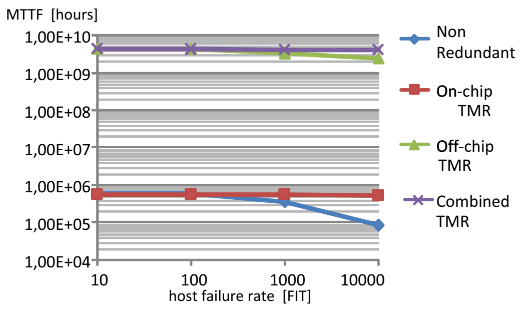
FCR	SIL	Failure Rate
Safety Critical Application Computer	4	10 FIT
Non Safety Critical Application Computer	1	1000 FIT
RMA	2	100 FIT
Trusted Subsystem	4	1 FIT

(b) Design Fault Model

Figure 5. Failure Rates for the Physical and Design Fault Model in FITs (failures in 10^9 hours)



(a) Different Application Computer Failure Rates



(b) Different Design Fault Rates and Correlations

Element	MTTF in Hours
Non redundant host	$8.48 \cdot 10^4$
On-chip TMR	$5.44 \cdot 10^5$
Off-chip TMR	$2.00 \cdot 10^9$
Combined	$4.13 \cdot 10^9$

(c) Default parameters

Figure 7. Comparison of MTTF in Different TMR Configurations

i.e., reliability requirements in the order of 1 FIT. Without off-chip, the reliability of a cluster follows from the reliability of a single SoC and does not exceed 1000 FIT.

The performance of off-chip TMR can be increased by combining on-chip and off-chip TMR. This effect gains particular significance for high host failure rates, e.g., as can be expected from transient faults in future VDSM technology.

5.2 Application Computer Design Failure Rate & Diversity Coverage

Figure 7(b) depicts the MTTF (caused by physical and design faults) of a system with one or more SoCs. Along the horizontal axis of Figure 7(b), different rates of failures caused by design faults are distinguished. The figure also shows the effect of different correlations between FCRs (between 0.1% and 100%). The rate of failures caused by physical faults is not varied.

For on-chip TMR, the reliability improvements of design diversity is less significant because of the dominant failure rates due to physical faults. For off-chip TMR, on the other hand, diversity has a significant positive impact for increasing failure rates due to design faults.

6 Conclusion

Future MPSoCs will have to cope with the increasing transient failure rates induced by VDSM technology. This paper has presented a solution for addressing this challenge based on TMR of IP cores in a novel MPSoC architecture with a time-triggered on-chip network. Compared to previous approaches for mitigating transient faults, the presented solution offers superior time and energy efficiency, enables the use of standard IP core libraries, improves resilience against spatial proximity faults, and establishes the foundation for design diversity and heterogeneity. Key mechanisms for TMR of IP cores are the inherent fault isolation and the determinism of the time-triggered on-chip network. The latter property (also known as replica determinism) ensures that correct replicas always reach the same computational result within a bounded time interval, which is required for exact voting. The former property is important for preserving the independence of replicas by preventing common mode failures of replicas.

Another focus of the paper has been the combination of on-chip TMR with off-chip TMR. Since individual MPSoCs cannot be expected to achieve a reliability as required for ultra-dependable systems (i.e., in the order of 10^{-9} failures/hour), active redundancy employing multiple MPSoCs is necessary.

Both the benefits of on-chip TMR, as well as the combination of on-chip and off-chip TMR have been analyzed quantitatively using reliability modeling. In particular, the results have demonstrated that on-chip TMR contributes significantly towards improving reliability of MPSoCs in the presence of high host failure rates caused by transient faults in VDSM technology.

7 Acknowledgments

This work has been supported in part by the European IST project ARTIST2 under project No. IST-004527 and the national FITIT project TT-SoC under grant number 813299/7852.

References

[1] W. S. A. Avizienis, M.R. Lyu. In search of effective diversity: a six-language study of fault-tolerant flight control software. *Eighteenth International Symposium on Fault-Tolerant Computing (FTCS-18)*, 1988.

[2] L. Anghel, D. Alexandrescu, and M. Nicolaidis. Evaluation of a soft error tolerance technique based on time and/or space redundancy. In *SBCCI '00: Proceedings of the 13th symposium on Integrated circuits and systems design*, page 237, Washington, DC, USA, 2000. IEEE Computer Society.

[3] L. Angheland and M. Nicolaidis. Cost reduction and evaluation of a temporary faults detecting technique. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pages 591–598, mar 2000.

[4] A. Bondavalli, S. Chiaradonna, F. D. Giandomenico, and F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *Transactions on Computers, Vol.49, Iss.3, Mar 2000*, 49:230–245, 2000.

[5] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. Doyle, W. Sanders, and P. Webster". "the mobius modeling tool". In *"Proc. of 9th International Workshop on Petri Nets and Performance Models"*, pages 241–250, sept 2001.

[6] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *Proc. of the Int. Conference on Dependable Systems and Networks*, pages 205–209. IEEE, 2002.

[7] C. Constantinescu. Trends and challenges in vlsi circuit reliability. *IEEE Micro*, 23(4):14–19, 2003.

[8] E. Dupont, M. Nicolaidis, and P. Rohr. Embedded robustness IPs for transient-error-free ICs. In *Proc. of the Conference on Design, Automation and Test in Europe*, page 244, Washington, DC, USA, 2002. IEEE Computer Society.

[9] FlexRay Consortium. BMW AG, DaimlerChrysler AG, General Motors Corporation, Freescale GmbH, Philips GmbH, Robert Bosch GmbH, and Volkswagen AG. *FlexRay Communications System Protocol Specification Version 2.1*, May 2005.

[10] M. T. G. Asadi. Soft Error Rate Estimation and Mitigation for SRAM-Based FPGAs. *International Symposium on Field Programmable Gate Arrays*, 2005.

[11] J. Gaisler. The leon3ft-rtax processor family and seu test results. In *Proc. of the 9th Annual Military and Aerospace Programmable Logic Devices International Conference*, Washington, D.C., Sept. 2006. Gaisler Research.

[12] P. Gelsinger. Microprocessors for the new millenium, challenges,opportunities, and new frontiers. In *Proc. of the Solid State Circuit Conference*. IEEE Press, 2001.

[13] P. Gil, J. Arlat, H. Madeira, Y. Crouzet, T. Jarboui, K. Kanoun, T. Marteau, J. Duraes, M. Vieira, D. Gil, J. Baraza, and J. Gracia. *Fault Representativeness*. LAAS-CNRS, Toulouse, France, 2002. Deliverable (ETIE2) of the European Project Dependability Benchmarking Dbench (IST-2000-25425).

[14] M. Gomaa, C. Scarbrough, T. Vijaykumar, and I. Pomeranz. Transient-fault recovery for chip multiprocessors. *IEEE Micro*, 23(6):76–83, 2003.

[15] G. Heiner and T. Thurner. Time-triggered architecture for safety-related distributed real-time systems in transportation systems. In *Proc. of the Twenty-Eighth Annual Int. Symposium on Fault-Tolerant Computing*, pages 402–407, June 1998.

[16] H.Kopetz. Pulsed data streams. In *IFIP TC 10 Working Conference on Distributed and Parallel Embedded Systems (DIPES 2006)*, pages 105–124, Braga, Portugal, Oct. 2006. Springer.

[17] Int. Standardization Organisation, ISO 7498. *Open System Interconnection Model*, 1994.

[18] iRoC Technologies. Radiation results of the ser test of actel fpga december 2005. Technical report, Dec. 2005.

[19] L. J.H. and H. R.E. Architectural principles for safety-critical real-time applications. In *Proceedings of the IEEE*, volume 82, pages 25–40, jan 1994.

[20] H. Kopetz. Sparse time versus dense time in distributed real-time systems. In *Proc. of 12th Int. Conference on Distributed Computing Systems*, Japan, June 1992.

[21] H. Kopetz. *Real-Time Systems Design Principles for Distributed Embedded Appl ications*. Kluwer Academic Publishers, Boston, 1997.

[22] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The Time-Triggered Ethernet (TTE) design. *Proc. of 8th IEEE Int. Symposium on Object-oriented Real-time distributed Computing (ISORC)*, May 2005.

[23] H. Kopetz and G. Grünsteidl. TTP – a protocol for fault-tolerant real-time systems. *Computer*, 27(1):14–23, Jan. 1994. Vienna University of Technology, Real-Time Systems Group.

[24] H. Kopetz and R. Nossal. Temporal firewalls in large distributed real-time systems. *Proc. of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, 1997.

[25] R. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200, Apr. 1962.

[26] S. Mishra, M. Pecht, and D. Goodman. In-situ sensors for product reliability monitoring. In *Proc. of SPIE*, volume 4755, pages 10–19, 2002.

[27] M. Nicolaidis. Time redundancy based soft-error tolerance to rescue nanometer technologies. In *Proc. of the 17th IEEE VLSI Test Symposium*, pages 86–94, Apr. 1999.

[28] OMG. Smart Transducers Interface. Specification ptc/2002-05-01, Object Management Group, May 2002. Available at `http://www.omg.org/`.

[29] B. Pauli, A. Meyna, and P. Heitmann. Reliability of electronic components and control units in motor vehicle applications. In *VDI Berichte 1415, Electronic Systems for Vehicles*, pages 1009–1024. Verein Deutscher Ingenieure, 1998.

[30] S. Poledna. Replica determinism in distributed real-time systems: A brief survey. *Real-Time Systems*, 6:289–316, 1994.

[31] S. Poledna. *Fault-Tolerant Real-Time Systems: The Problem of Replica Determinism*. Kluwer Academic Publishers, 1996.

[32] Semiconductor Industry Association (SIA). International technology roadmap for semiconductors – 2006 update. Technical report, 2006.

[33] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 389–398, Washington, DC, USA, 2002. IEEE Computer Society.

[34] N. Suri, C. Walter, and M. Hugue. *Advances In Ultra-Dependable Distributed Systems*, chapter 1. IEEE Computer Society Press, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264, 1995.