

COTS-Based Fault Tolerance in Deep Space: A Case Study on IEEE 1394 Application

Ann T. Tai
IA Tech, Inc.
10501 Kinnard Avenue
Los Angeles, CA 90024

Savio N. Chau Leon Alkalai
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

Among the COTS applications in the X2000 architecture for deep-space missions, the use of commercial bus standards is the highest-payoff COTS application since a bus interface has a global impact on system cost and capability. While COTS bus standards enable significant cost reductions, it is a great challenge for us to deliver a highly-reliable long-term survivable system employing COTS standards that are not developed for mission-critical applications. The essence of our solution to the problem is to exploit the pertinent standard features of a COTS product to circumvent its shortcomings, including those standard features that are not originally designed for reliability purpose. In this paper, we discuss our experiences and findings on the design and assessment of an IEEE 1394 compliant fault-tolerant bus architecture. We first derive and qualitatively analyze a “stack-tree topology” that not only complies with IEEE 1394 but also enables the implementation of a fault-tolerant bus architecture without node redundancy. We then present a quantitative evaluation that demonstrates significant reliability improvement resulting from the COTS-based fault tolerance. We also describe the benefits from using COTS intellectual properties for hardware implementation of the bus architecture and discuss our general approach to COTS-based fault tolerance.

Keywords: COTS-based fault tolerance, IEEE 1394, deep-space applications, tree topology, bus network reliability

Principal Contact: Ann T. Tai, a.t.tai@ieee.org

1 Introduction

The X2000 system architecture that is being developed by NASA/JPL is a distributed, scalable, fault-tolerant avionics architecture for multiple deep-space missions [1, 2, 3]. The architecture is currently the baseline for the Europa Orbiter mission which is scheduled for launch in year 2003 [4]. In the X2000 architecture, the multiple computing nodes and devices are symmetric, meaning that the roles of computing nodes are interchangeable while devices are treated as intelligent nodes in the network. Moreover, in order to meet the challenging performance, cost, and schedule requirements, this architecture stresses using *commercial-off-the-shelf* (COTS) products, standards and intellectual properties (IPs).

Among the COTS applications in the X2000 architecture, the use of commercial bus standards is the highest-payoff COTS application. This is due to the fact that, unlike a system component or subsystem, a bus interface is normally used throughout the entire system architecture and thus has a *global* impact and enabling effect on system cost and capability, respectively. After extensive survey and rigorous evaluation efforts, the Peripheral Component Interface (PCI) [5], IEEE 1394 [6, 7], and I2C [8] have been selected to implement the local computer bus, system bus, and engineering bus, respectively, in the X2000 architecture [9]. While COTS bus standards enable significant cost reductions, it is a great challenge for us to deliver a highly-reliable long-term survivable system employing COTS standards that are not developed for mission-critical applications. The central purpose of this paper is to report our experiences and findings on the design and assessment of the IEEE 1394 compliant fault-tolerant bus network for the X2000 architecture.

The IEEE 1394 standard has two implementations, namely, *cable implementation* and *backplane implementation* [6]. The backplane implementation that adopts the multi-drop topology has been investigated by the aerospace industry [10]. Yet, we have selected the cable implementation for the X2000 architecture because of its higher data rate, lower power consumption, and significantly more substantial commercial support. Nonetheless, in terms of fault tolerance, the tree topology criterion for the cable implementation makes it formidable for a non-redundant bus network to tolerate node/link failures. Namely, any single node or link failure may result in tree partitioning such that fault-tolerant routing will not be possible. A brute-force approach is to duplicate the circuitry of each node and to cross-strap them to the bus network. However, this approach is not feasible for the X2000 architecture because the significant increases in power consumption, mass, and volume would violate the constraints. Although IEEE 1394 has been enjoying fast-growing popularity, how to circumvent its limitations in fault tolerance for implementing highly reliable systems has not yet received enough attention. The essence of our solution to the problem is to exploit the

standard features of COTS products and standards in an innovative manner to circumvent their shortcomings, though these standard features may not be originally designed for fault tolerance. In this paper, we derive and qualitatively analyze a special realization of tree topology, called the “stack-tree topology,” which not only complies with the IEEE 1394 standard but also enables the implementation of a fault-tolerant bus architecture [11]. In particular, the bus network architecture based on stack-tree topology enables us to exploit a standard feature of IEEE 1394 called “port-disable” [12], which ensures the normal communications among nodes to continue so long as no non-clustered multiple cut-type failures occur (see Section 3). We also present a reliability evaluation that quantitatively compares the bus network architectures that use the stack-tree topology and its variants, namely, CST_s , CST_d and CST_r . Due to the complex failure scenarios of the network architecture CST_r , to take into account exhaustively and exclusively for all the failure scenarios based on straightforward use of combinatorics would be very difficult and inefficient. Therefore, we construct an analytic model based on recursive functions that traverse the entire network and enumerate all the concerned scenarios, yielding exact solution for the reliability measure. The evaluation results demonstrate that the proposed COTS-based fault-tolerant bus architecture leads to significant reliability gain for long-life deep-space missions.

The remainder of the paper is organized as follows. In Section 2, we explain why we have selected IEEE 1394, and describe its advantages and restrictions. In Section 3, we elaborate and qualitatively analyze the stack-tree topology that complies with IEEE 1394 and facilitates the use of its “port-disable” feature for fault tolerance, followed by Section 4 which presents the quantitative methods and results of reliability evaluation for the resulting fault-tolerant bus architecture. Section 5 describes the benefits from using COTS IPs for hardware implementation of the bus architecture and discusses the essence of our general approach to COTS-based fault tolerance. In the concluding remark, we summarize the significance of this effort.

2 IEEE 1394: Selection, Advantages, and Limitations

In the process of selecting the high-speed and low-power buses, we examined many commercial bus interfaces. The candidates included IEEE 1394, Fiber Channel, Universal Serial Bus (USB), Fast Ethernet, Serial Fiber Optic Data Bus (SFODB), ATM, Myrinet, FDDI, AS1773, and SPI. Many of them (e.g., USB, AS1773, and SPI) fail to meet the requirements on data rate; some of them are not suitable for real-time applications because of the indeterminacy of bus latency; whereas others have high power consumption which is unacceptable for deep-space applications (e.g., Fiber Channel, SFODB, ATM, and Myrinet). Moreover,

unlike prior space applications such as the Mars Pathfinder case, for which “COTS-based” means the direct use of commercial parts, components, or subsystems, the term COTS has a special meaning for the X2000 architecture. Since at least one of the prospect X2000 customers, namely, Europa, requires to survive in a high-radiation environment, all the critical electronic components must be fabricated on specialized semiconductor foundries. Therefore, another important selection criterion is the availability of the radiation-hardened components for the COTS bus interface and COTS ASIC core designs (COTS IPs) portable to a radiation-hardened foundry. A rigorous evaluation based on these criteria resulted in the selection of IEEE 1394. Similar criteria were given to the engineering bus selection while the low-power and high-performance requirements were further stressed and deemphasized, respectively. Our tradeoff study concluded that I2C was the best compromise. The I2C bus plays an important role in assisting the 1394 system bus for fault detection and recovery. Nonetheless, the detailed discussion about this bus is beyond the scope of this paper (but can be found in [9]). The selection of 1394 and I2C enables the X2000 Program to procure COTS ASIC core designs, which can be integrated into a single chip. It is projected that this approach will reduce the total design effort by at least 30% when compared with the Cassini ASIC design [13], while the ASIC complexity is increased by 400%. Moreover, COTS products required by the IEEE 1394 and I2C implementation, such as bus monitors, prototype boards and device drivers, are highly available, which in turn, leads to further savings.

IEEE 1394 is originally designed for commercial applications such as multimedia and mobile phones. The current version of IEEE 1394 can support data rates of 100 Mbps, 200 Mbps, and 400 Mbps for the *cable implementation* that is based on a tree topology, and 50 Mbps and 100 Mbps for the *backplane implementation* that is a multi-drop bus. Indeed JPL designers were more familiar with the backplane implementation because of its resemblance to the MIL-STD-1553 bus that was used in the Cassini project [13]. Nonetheless, we selected the cable implementation due to its high speed and extensive commercial support which enable us to maximize the benefits from using COTS. Accordingly, unless it is explicitly stated, all discussions on IEEE 1394 in this paper refer to the cable implementation.

Although there are various types of tree structure that satisfy the topology requirement of IEEE 1394, it is preferred to have a “regular topology” for space applications. By a “regular topology,” we mean a structure that is topologically simple and can be easily maintained as nodes are added to or deleted from the system such that testing and integration can be accomplished efficiently at low cost. Therefore, the stack-tree topology depicted in Figure 1 is proposed, where a node is either a flight computer or a device. There are three physical layer ports in each node. For each branch node, at least two of these ports are connected to other nodes, while a leaf node has only one connected port. Figure 2 depicts the baseline

X2000 First Delivery avionics architecture that applies the IEEE 1394 bus with a variant stack-tree topology CST_r (see Section 3).

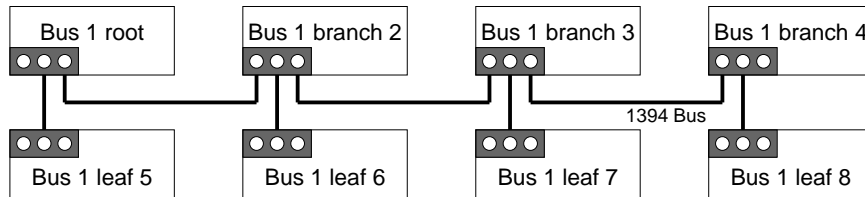


Figure 1: Bus Network based on Stack-Tree Topology

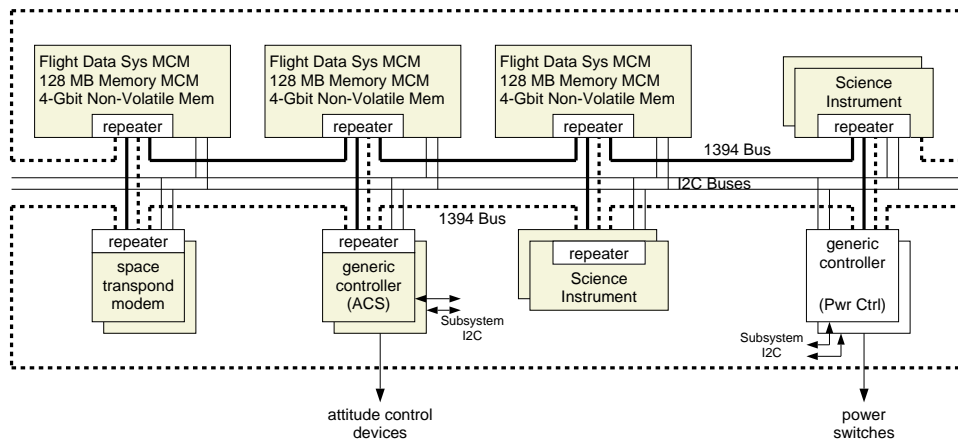


Figure 2: Baseline X2000 First Delivery Architecture

Like any tree structures, the stack-tree topology shown in Figure 1 has a potentially serious drawback. Namely, a tree structure by itself is not fault tolerant as any single node or link failure may result in tree partitioning such that fault-tolerant routing will not be possible. What makes the design more difficult is that to duplicate and cross-strap nodes for fault tolerance purpose cannot be justified due to the constraints on power, mass, and volume. Although various schemes of fault-tolerant bus network have been proposed in research literatures (see [14, 15], for example), the restrictions from 1394 and from our application prevent us from utilizing those schemes since the majority of them involve either loops or redundant nodes.

The 1394 standard has some error detection provisions such as CRC, they are nonetheless inadequate to ensure reliability for long-life missions. On the other hand, IEEE 1394A [12] provides an employable feature called “port-disable,” which allows us to implement a 1394 compliant reconfigurable bus architecture, though this feature is not intentionally designed for fault tolerance. In the following section, we describe and analyze, the stack-tree topology and its variants based on which we design the IEEE 1394 compliant fault-tolerant bus architecture.

3 Stack-Tree based Bus Architecture

3.1 Concepts

In the interest of bridging the terminology between network topology and the X2000 MCM-stack packaging technology [16], we call the proposed topology “stack-tree topology.”

Definition 1 *A stack tree is a tree where each branch node is connected to at most three other nodes among which at most two are branch nodes.*

For example, the trees in Figures 3(a), (c) and (d) are stack trees while that in Figure 3(b) is not (as the right node at the first level below the root is connected to three branch nodes).

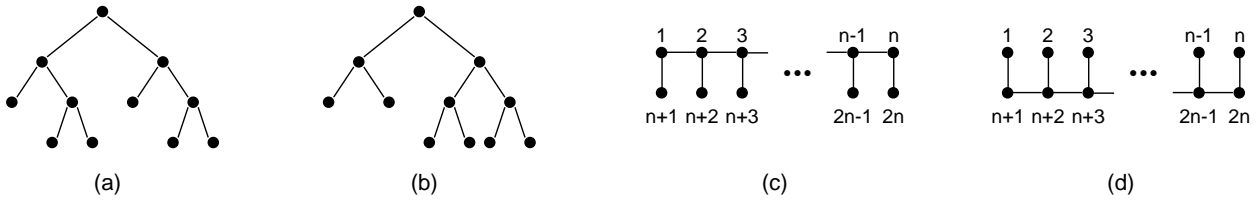


Figure 3: Trees

Definition 2 *A complete stack tree is a stack tree where each branch node is connected to exactly one leaf node.*

Figure 3(c) depicts a complete stack tree (CST) with n branch nodes. We call this topology *simplex complete stack tree* which is denoted as CST_s . Note that the nodes are labeled such that the branch nodes have the ID numbers from 1 to n , while the leaf nodes have the ID numbers from $n + 1$ to $2n$. This labeling scheme will be used in the remainder of the paper. Further, we use n , the number of branch nodes in a CST, to denote the *size* of the tree. Note also that the trees in Figures 3(c) and 3(d) are both CST_s . Based on the CST in Figure 3(c), we can define *CST mirror-image* as follows.

Definition 3 *The mirror-image of a complete stack tree is a tree obtained by (1) removing the edges connecting the branch nodes with the ID numbers i and j which satisfy the relation $|i - j| = 1$; (2) adding edges to connect the leaf nodes with the ID numbers k and l which satisfy the relation $|k - l| = 1$.*

Clearly, the CST shown in Figure 3(d) is a mirror image of that in Figure 3(c). It is worth to note that, if we connect $2n$ nodes with a CST-based network and its mirror image, then the two networks will not have any branch nodes in common.

3.2 Applications

3.2.1 The CST_d Scheme

The performance of the X2000 avionics system needs to be scalable and gracefully degradable. Accordingly, our objective is to develop a fault-tolerant bus network architecture that will allow all the surviving nodes in the bus network to remain connected in the presence of node failures, without requiring redundant nodes. The fact that a CST and its mirror image do not have branch nodes in common implies that losing a branch node in one tree will not partition its mirror image. Accordingly, a dual bus scheme comprising a CST and its mirror image, referred to as the *CST dual scheme* (denoted as CST_d), as shown in Figure 4(a), will be effective in tolerating single or multiple node failures given that

1. The failed nodes are of the same type (all branch or all leaf) with respect to one of the complete stack trees (see Figure 4(b)), or
2. The failed nodes involve both branch and leaf nodes but they form a cluster at either end (or both ends) of a CST, which will not affect the connectivity of the remainder of the tree (see Figure 4(c)).

We use *terminal clustered branch-leaf failures* to refer to the second failure pattern described above. Thus, for the cases which involve only the above failure patterns, all the surviving nodes will remain connected (no network partitioning). On the other hand, if a branch node and a leaf node in a CST_d based network fail in a form other than terminal clustered branch-leaf failure (see Figure 4(d)), both the primary and mirror-image networks will be partitioned.

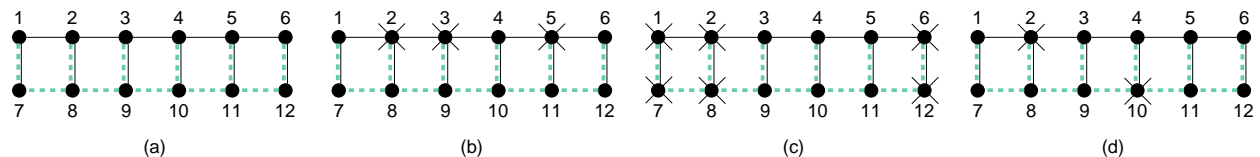


Figure 4: CST -Based Dual Bus Network (CST_d)

3.2.2 The CST_r Scheme

With the motivation of building a robust bus network architecture capable of tolerating more node failures (in terms of number and pattern), we exploit a unique feature of IEEE 1394, namely, the *port-disable* capability (a port is the physical interface between a node and a link) [12]. This feature enables the physical connections between the physical layer of a

node and the serial bus cable to become “invisible” from the view point of the reminder of the bus network. Another feature of IEEE 1394 we utilize is the *keep-alive* of the physical layer with cable power. This feature allows the link-layer hardware and the host processor to be powered off without affecting the capability of the physical layer to pass-on messages, enabling a failed node to be isolated and bypassed. The “port-disable” and “keep-alive” features of IEEE 1394 collectively imply the following:

- 1) By using disabled ports, backup connections between nodes can be added without forming loops (recall that loops are prohibited by IEEE 1394). By a “backup connection,” we mean a serial bus cable that connects (via disabled ports) two nodes which are not expected to have a direct connection in the original network configuration (differing from connection replication); and
- 2) Upon fault detection, by disabling physical ports, a failed node will be allowed to be isolated from the rest of the bus network, and necessary backup link(s) can be activated (by enabling the corresponding ports) to repair the partitioned network such that messages can be routed in a reconfigured network, bypassing the failed node.

Consider a bus network based on the CST_s topology with n branch nodes (size n), as shown in Figure 5(a). If we add a backup link between any two leaf nodes labeled i and j which satisfy the relation $|i - j| = 1$, and also add backup links to connect branch nodes 1 and n , leaf nodes $(n + 1)$ and $2n$, then we get a topology as shown in Figure 5(b) (an instance of the topology in which $n = 6$). Because the added connections (dashed edges) are of inactive nature, the bus network remains free of loop and thus complies with the IEEE 1394 tree topology criterion. Figure 5(c) illustrates the bus network from a 3-dimensional perspective, which enables us to visualize the network as a ring. Accordingly, we denote this bus network configuration as CST_r . To aid the description of failure mechanisms of a CST_r based bus network, we introduce the following terminology:

Definition 4 *A failed branch node i and a failed leaf node j in a CST_r based network of size n will form a cut-type failure if $|(j \bmod n) - (i \bmod n)| \leq 1$.*

Figure 6 illustrates the concepts of cut-type and non-cut-type failures. Specifically, the failure comprised by nodes 2 and 9 in Figure 6(a), that by nodes 5 and 11 in Figure 6(b), and that by nodes 1 and 12 in Figure 6(c), are cut-type failures. On the other hand, the node failures shown in Figures 6(d) and 6(e) are non-cut-type failures. Further, we use the term *clustered failure* to refer to the failure of a group of nodes which are adjacent to each other. Figures 7(a) and 7(b) illustrate the scenarios of clustered and non-clustered multiple cut-type failures, respectively. Clearly, while the non-clustered cut-type failures shown in 7(b)

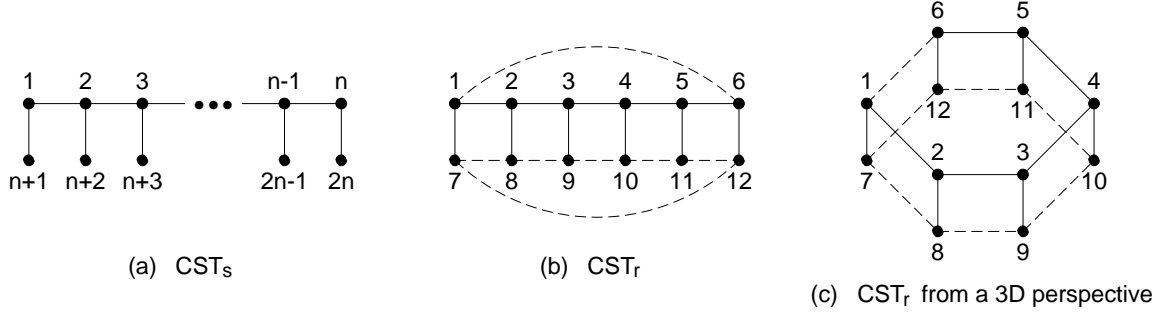


Figure 5: Applying Disabled Backup Links

leads to bus network partitioning (i.e., the traffic across either of the “cuts” are disabled), the clustered cut-type failures shown in 7(a) does not even if node 6 also fails (i.e., the traffic across the clustered “cuts” can be re-routed through the enabled backup link $\{7,12\}$), although both scenarios involve multiple cut-type failures. The above discussion shows the necessary and sufficient condition for partitioning a CST_r based bus network:

Remark: A bus network based on the CST_r topology will be partitioned if and only if there exist multiple cut-type failures which do not constitute a single cluster.

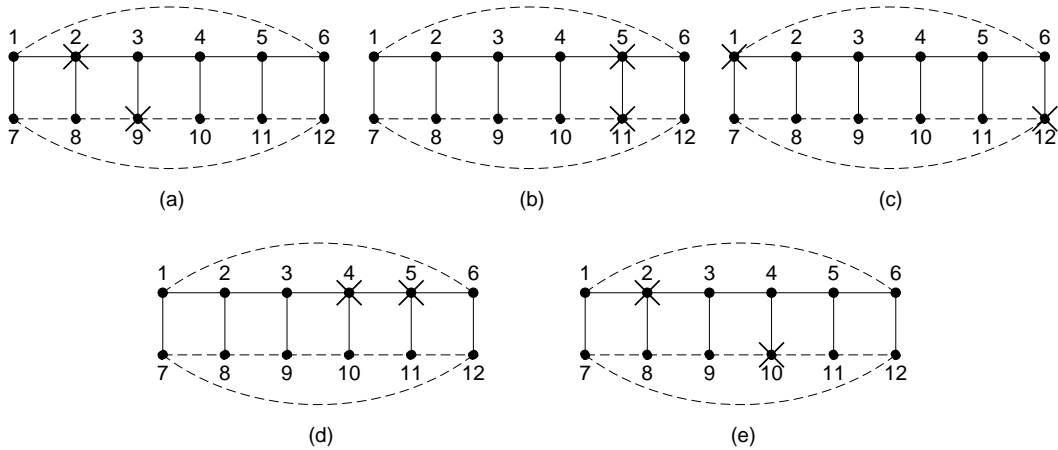


Figure 6: Cut-Type and Non-Cut-Type Failure

Figure 8 illustrates how partitioning happens in a network that originally has a CST_r structure: The first cut-type failure (single or clustered) will break the ring structure (Figure 8(a)) so that the remainder of the network becomes a CST_s based structure with some backup links (Figure 8(b)); whereas the second cut-type failure (single or clustered) will break the CST_s based structure, resulting in network partitioning (communication between any two nodes separated by the “cut” becomes impossible, see Figure 8(c)).

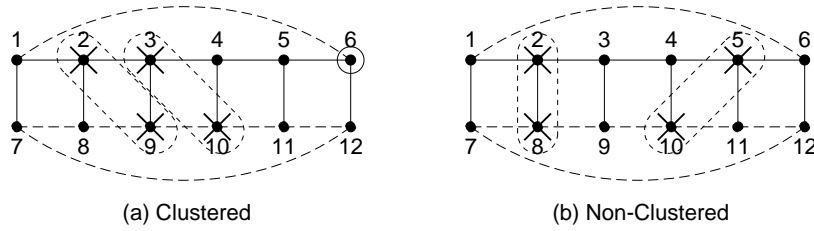


Figure 7: Clustered vs. Non-Clustered Multiple Cut-Type Failures

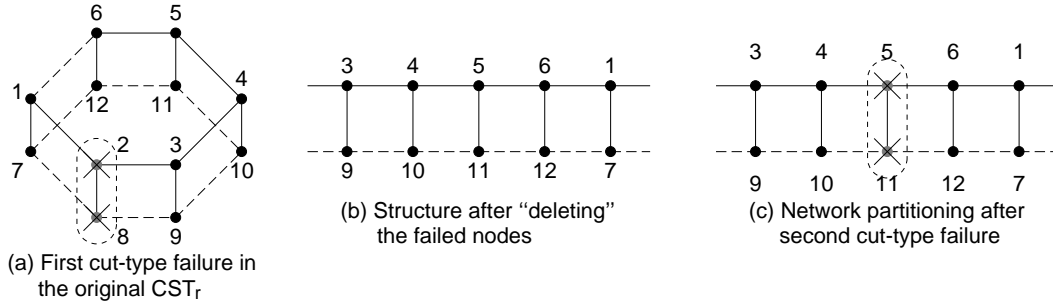


Figure 8: CST_r -Based Bus Network Partitioning

Figure 9 depicts the simplified X2000 architecture in which the CST_r based bus network described above is applied. In the figure, the solid and dashed thick lines marked “1394 Bus” represent the active and backup links, respectively. During normal operation, the active connections are driven by enabled ports while the ports of backup connections are disabled to avoid forming loops. The thin lines marked “I2C Bus” correspond to the interface for fault detection, isolation and reconfiguration. The I2C bus is a very simple low-speed multi-drop bus and used mainly for protecting the 1394 bus. Hence this engineering bus has low utilization and low power consumption.

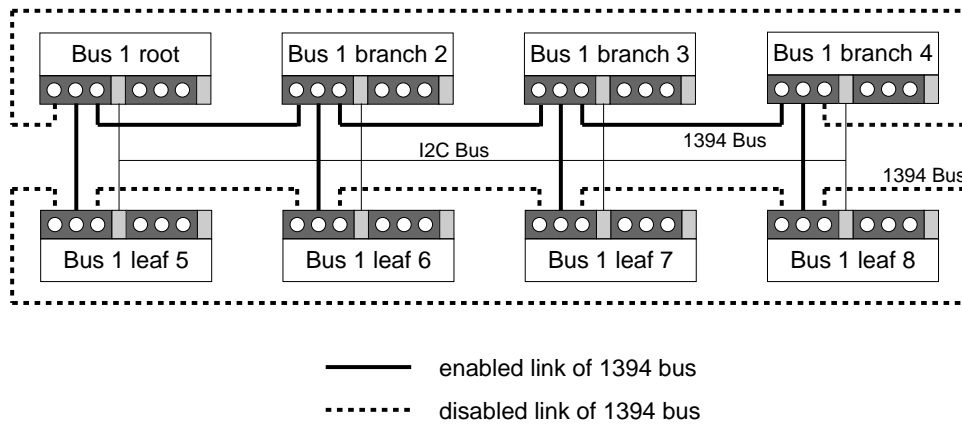


Figure 9: CST_r -Based Fault-Tolerant Bus Network

4 Bus Network Reliability Evaluation

4.1 Definition and Notation

In accordance with the objective of the fault-tolerant bus architecture design stated in Section 3.2, we define bus network reliability as follows:

Bus network reliability is the probability that, through a mission duration t , the network remains in a state in which all the surviving nodes are connected.

Indeed the causes of a node failure encompass physical layer failure, link layer failure and CPU failure. Moreover, while redundant links (serial bus cables) are permitted in the X2000 architecture, to duplicate and cross-strap nodes for bus network fault tolerance purpose cannot be justified due to the power, mass, and volume constraints. As a result, the likelihood of node failure is significantly greater than that of link failure. Therefore, in the reliability assessment that follows, we concern only node failure. On the other hand, when a node fails, we assume that there is a possibility that the faulty node may go undetected or the corresponding network reconfiguration process (including port disabling/enabling, etc.) may unexpectedly crash the system. We call the complement of the probability of such an event “coverage.” Before proceeding to derive solutions of reliability measures, we define the following notation:

R_s Reliability of a CST_s based bus network.

R_d Reliability of a CST_d based bus network.

R_r Reliability of a CST_r based bus network.

λ Poisson failure rate of a node.

t Mission duration.

Q_t Probability that a node fails during a mission.

c Conditional probability (coverage) that a failed node is detected and the corresponding reconfiguration process succeeds given that such a node failure occurs.

4.2 Solutions for R_s and R_d

We begin with analyzing the CST_s and CST_d based bus network schemes. As explained in Section 3.2, terminal clustered branch-leaf failures in a CST_s or CST_d will not affect the connectivity of the remainder of the tree. Thus, we can retrieve a “remainder” from the

original CST_s or CST_d , by eliminating the portion(s) comprised by the terminal clustered branch-leaf failures. Consequently, according to the definition of bus network reliability defined in Section 4.1, the reliability calculation for a remainder leads to the solutions of R_s and R_d . Specifically, we evaluate the reliability of a remainder by conditioning it on its size k (the number of its branch nodes), or equivalently speaking, by conditioning it on the event that the terminal clustered branch-leaf failures involve $(n - k)$ branch-leaf node pairs. Note that a remainder of size k in the original CST_s or CST_d of size n has $(n - k + 1)$ possible positions (which in turn, determines the positions of the terminal clustered branch-leaf failures). Note also that the reliability of a remainder of size k for a CST_s based network is the probability that all k branch nodes are failure-free and all faulty leaf nodes are detected and reconfigured successfully (if any). Letting this probability be denoted as $U(k)$, we have,

$$U(k) = (1 - Q_t)^k \sum_{j=0}^k \binom{k}{j} (1 - Q_t)^{k-j} (c Q_t)^j.$$

Then, the theorem of total probability leads to the following solution for R_s :

$$R_s(t) = \sum_{k=1}^n (n - k + 1) U(k) (c Q_t)^{2(n-k)}, \quad (1)$$

where $Q_t = 1 - e^{-\lambda t}$ is the probability that a node fails during mission time t , and $(c Q_t)^{2(n-k)}$ is the probability that $2(n-k)$ nodes are involved in the terminal clustered branch-leaf failures and fault detection and reconfiguration for each of the failed nodes are successful. Note that although we assume that the time to a node failure is exponentially distributed, the models developed in this paper can accommodate other assumptions such as Weibull distribution by formulating Q_t accordingly.

Likewise, letting the reliability of a size- k remainder in a CST_d based network be denoted as $V(k)$. According to the failure scenario analysis in Section 3.2.1, we have

$$V(k) = 2(1 - Q_t)^k \sum_{j=1}^k \binom{k}{j} (1 - Q_t)^{k-j} (c Q_t)^j + (1 - Q_t)^{2k}.$$

Thus, the measure R_d can be expressed as

$$R_d(t) = \sum_{k=1}^n (n - k + 1) V(k) (c Q_t)^{2(n-k)}. \quad (2)$$

4.3 A Recursive Model for R_r

The solution for R_r would be impossible to obtain based on the straightforward use of combinatorics methods because

- 1) The bus network becomes capable to tolerate more node failure patterns, which makes the representation of the conditions under which the system can survive more complex, and
- 2) The ring-like structure makes it difficult to ensure that the scenarios considered in the model are exhaustive and mutually exclusive.

As explained in Section 3.2.2, a bus network architecture with a CST_r topology will be partitioned if and only if there exist multiple cut-type failures which do not constitute a single cluster. In other words, the surviving nodes in a CST_r based bus network will remain connected if there exists *at most one* cut-type failure cluster. In the model construction method described below, we view a single cut-type failure as a special case of cut-type failure cluster (where the size of the cluster is one) and treat a size- n network that is free of cut-type failure as a special case of remainder (where the sizes of the cluster and remainder equal to 0 and n , respectively).

Specifically, we first condition network reliability on the size of a cut-type failure cluster (the number of branch nodes involved in the cluster), then we evaluate the probability that the remainder, which is the portion of the CST_r based network structure excluding the cluster, is free of cut-type failure. The key step toward the evaluation of this probability is the derivation of a set of recursive functions that enumerate the combinations and permutations of failed and surviving nodes in the remainder where cut-type failure is absent. By successively expanding and reducing the sizes of the failure cluster and the remainder, respectively, and employing the recursive functions, we exhaustively enumerate the probabilities that a remainder is cut-type failure free. Accordingly, the measure we seek to evaluate can be expressed as

$$R_r(t) = \sum_{m=1}^{n-1} n(cQ_t)^{2m} F(n-m) + F(n) + (n-1)(G(n) + H(n)), \quad (3)$$

where the index m represents the size of a cut-type failure cluster, $(cQ_t)^{2m}$ is the probability that such a cluster exists and each of the individual failed nodes comprising the cluster is detected and undergoes reconfiguration successfully, the coefficient n is the number of possible positions of the cluster in the CST_r based network, and $F(n-m)$ evaluates the probability that the remainder is cut-type failure free given that the size of the failure

cluster is m . This probability is solved by a set of recursive functions which “walk through” the remainder backward, ensuring that 1) the distinct scenarios characterized by the number and positions of failed nodes are exhaustively enumerated, and 2) the remainder is cut-type failure free. More succinctly,

$$F(k) = F_0(k) + F_1(k) + F_2(k), \quad (4)$$

in which k is the size of the remainder ($k = n - m$). Then, in each iteration of the recursion, F_0 , F_1 and F_2 individually traverse from the end of a size- i “sub-remainder” where

- 1) both the nodes i and $(i + n)$ are surviving nodes,
- 2) i is a surviving node and $(i + n)$ is a failed node, and
- 3) i is a failed node and $(i + n)$ is a surviving node,

respectively. For each step of the traversal, the recursive functions are derived in a way such that no cut-type failure would be formed within the resulting node-pair sequence. More precisely,

$$\begin{aligned} F_0(i) &= [F_0(i-1) + F_1(i-1) + F_2(i-1)] (1 - Q_t)^2 \\ F_0(1) &= (1 - Q_t)^2 \end{aligned} \quad (5)$$

$$\begin{aligned} F_1(i) &= [F_0(i-1) + F_1(i-1)] (1 - Q_t) c Q_t \\ F_1(1) &= (1 - Q_t) c Q_t \end{aligned} \quad (6)$$

$$\begin{aligned} F_2(i) &= [F_0(i-1) + F_2(i-1)] c Q_t (1 - Q_t) \\ F_2(1) &= c Q_t (1 - Q_t) \end{aligned} \quad (7)$$

For example, if nodes i and $(i + n)$ in the remainder are a failed and a surviving node, respectively, then the status of the nodes $(i - 1)$ and $(i + n - 1)$ must be 1) both surviving, or 2) failed and surviving, respectively (otherwise a cut-type failure will be formed). This particular sequencing rule is implemented by the recursive function F_2 (Equation (7)).

To aid further explanation of the model, we introduce the term *cut-type failure node pair*, abbreviated as *CFP*, which refers to the node pair that forms a cut-type failure. Per

Definition 4 in Section 3.2.2, a branch node i in a CST_r based network of size n potentially could be involved in three differing CFPs, as shown in Figure 10. We call the CFPs $\{i, i+n-1\}$, $\{i, i+n\}$ and $\{i, i+n+1\}$ forward CFP, vertical CFP and backward CFP, respectively.

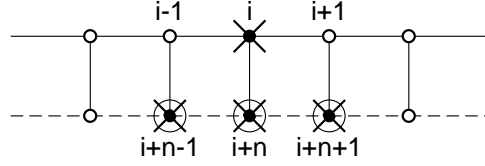


Figure 10: Cut-Type Failure Node Pair

It is worth to note that, due to different patterns of the junctions between the cut-type failure cluster and the remainder, the first term of Equation (3) takes into account not only for the cut-type failure clusters that are constituted by vertical CFPs but also the cut-type failure clusters that can be viewed as the clusters formed by forward and backward CFPs, as illustrated in Figures 11(a) and 11(b), respectively.

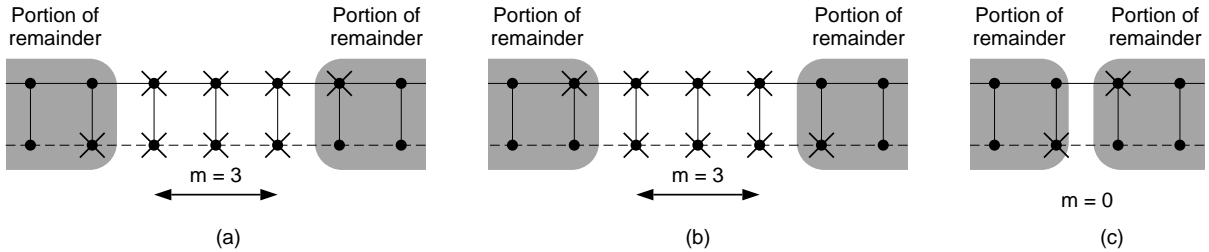


Figure 11: Failure Cluster and Remainder

For the special case in which $m = 0$, the cut-type failure cluster becomes degenerate while the remainder spans the entire CST_r based bus network. The corresponding scenarios are enumerated by the rest terms in Equation (3), namely, $F(n)$ and $(n-1)(G(n) + H(n))$. Although no coefficient is attached to $F(n)$, different starting positions of this special case “remainder” in the CST_r based bus network are implicitly taken into account by the recursive functions. In other words, as Equations (5), (6) and (7) together exhaustively enumerate the combinations and permutations of failed and surviving nodes such that no cut-type failure will be formed within the remainder, different positions of a “remainder” in the CST_r structure (where $m = 0$) are indeed “inherently” considered by $F(n)$. For example, the “remainder” in Figure 12(b) can be viewed as a result of shifting the starting position of that in Figure 12(a) toward right by one node position — both cases (where $m = 0$) are enumerated by $F(n)$.

The only exception is the scenario in which $m = 0$ and a single cut-type failure (a forward or backward CFP) is formed at the “junction” where the two ends of the “remainder” merge,

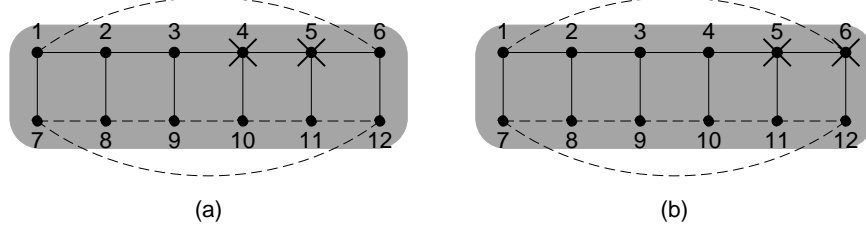


Figure 12: “Positions” of a Remainder of Size n

as illustrated in Figure 11(c) (recall that a single CFP by itself will not partition the network). Since the recursive functions are formulated in a way such that the cases where the remainder has an internal CFP are excluded, what missed in the term $F(n)$ but compensated by the term $(n-1)(G(n) + H(n))$ in Equation (3) are the $(n-1)$ different positions of this particular CFP.

The derivations of $G(n)$ and $H(n)$ are based on the recursive functions which are formulated in a manner such that only the scenarios where the “merge” of the two ends of the “remainder” results in a forward CFP and a backward CFP are considered, respectively (while CFP does not exist elsewhere):

$$G(n) = [G_0(n-1) + G_1(n-1)](1 - Q_t)c Q_t \quad (8)$$

$$G_0(i) = [G_0(i-1) + G_1(i-1) + G_2(i-1)](1 - Q_t)^2 \quad (9)$$

$$G_0(1) = 0$$

$$G_1(i) = [G_0(i-1) + G_1(i-1)](1 - Q_t)c Q_t \quad (10)$$

$$G_1(1) = 0$$

$$G_2(i) = [G_0(i-1) + G_2(i-1)]c Q_t(1 - Q_t) \quad (11)$$

$$G_2(1) = c Q_t(1 - Q_t)$$

$$H(n) = [H_0(n-1) + H_2(n-1)]c Q_t(1 - Q_t) \quad (12)$$

$$H_0(i) = [H_0(i-1) + H_1(i-1) + H_2(i-1)](1 - Q_t)^2 \quad (13)$$

$$H_0(1) = 0$$

$$H_1(i) = [H_0(i-1) + H_1(i-1)](1 - Q_t)cQ_t \quad (14)$$

$$H_1(1) = (1 - Q_t)cQ_t$$

$$H_2(i) = [H_0(i-1) + H_2(i-1)]cQ_t(1 - Q_t) \quad (15)$$

$$H_2(1) = 0$$

4.4 Evaluation Results

Applying the models developed in the previous subsections and using *Mathematica*TM, reliability measures for the bus networks based on CST_s , CST_d , and CST_r are evaluated with respect to the node failure rate λ , size of bus network n and mission duration t (in hours).

Figure 13 depicts R_s , R_d and R_r as functions of component node failure rate λ . In this evaluation, the size of a CST-based bus network n is set to 16 (a 32-node network), the fault detection and reconfiguration coverage c is set to 0.9999 (which is conservative as the coverage is defined on a single node basis), and mission duration t is set to 90,000 hours (which implies an *over 10-year* long-life mission). It can be observed from the figure that, while CST_d results in an appreciable amount of reliability improvement over CST_s , CST_r leads to significantly more reliability gain. The quantitative results show that R_r will be greater than 0.999997 if node failure rate λ is 10^{-8} or lower. Furthermore, after λ reaches 10^{-7} , both R_s and R_d rapidly drop and become unacceptable but R_r remains relatively steady.

Figure 14 shows the results of the evaluation for which λ is set to 10^{-7} , t and c remain 90,000 hours and 0.9999, respectively, while n becomes a variable parameter. It is interesting to note that R_d equals to R_r when $n = 2$. This is a reasonable result because for a 4-node network, the node failure patterns that will partition a CST_d based network coincide with the failure patterns that will partition a CST_r based network. It can also be observed that the reliability improvement resulting from applying the CST_r network architecture becomes more significant as the size of the network increases. This is because more routing alternatives that are comprised by active and backup links are available in a larger CST_r based network.

Figure 15 illustrates the evaluation results of a study for which λ and n are set to 10^{-7}

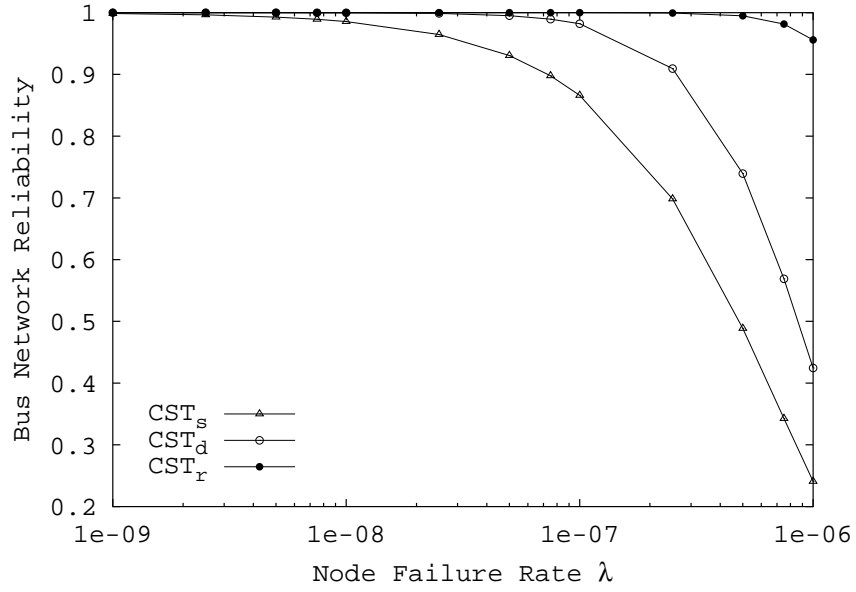


Figure 13: Bus Network Reliability as a Function of Node Failure Rate ($n = 16$)

and 16, respectively, and c remains 0.9999, while mission duration t becomes a variable parameter. Apparently, both R_s and R_d become unacceptable for long-life missions. On the other hand, R_r remains very reasonable (i.e., 0.999929) even when $t = 100,000$ (a mission duration about 11.5 years).

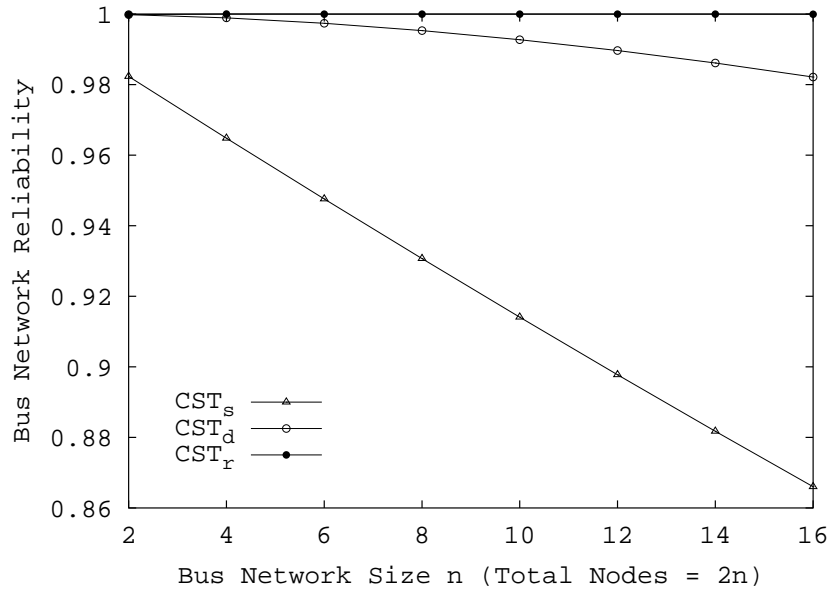


Figure 14: Bus Network Reliability as a Function of Network Size ($\lambda = 10^{-7}$)

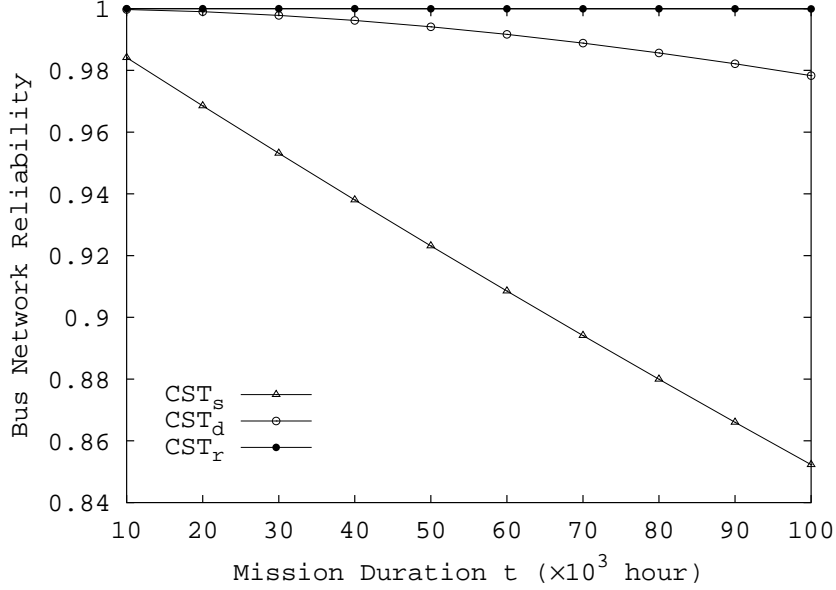


Figure 15: Bus Network Reliability as a Function of Mission Duration ($\lambda = 10^{-7}$)

5 Discussion

The application of the CST_r based bus network for the X2000 avionics system is enabled by a set of algorithms for error detection, diagnosis, and reconfiguration, which are built on the top of the standard features of IEEE 1394 [9]. In order to facilitate efficient network reconfiguration, the bus set that consists of IEEE 1394 and I2C is duplicated at the system level such that one bus set is the mirror image of the other, as depicted in Figure 16. Accordingly, when a node failure is detected, it is first examined whether the failed node is a branch node in the active bus and a leaf node in the backup bus. If this is the case, recovery can be accomplished by switching to the backup bus and then reconfiguring the faulty bus in background. However, if the failed node is a branch node for both buses, then recovery should begin with reconfiguring the faulty active bus by port-enabling/disabling. In this manner, the bus architecture takes the advantages of both the CST_d and CST_r topologies (see Sections 3.2.1 and 3.2.2).

It is worthy noting that the dual CST_r scheme is justified by 1) its ability in enhancing error recovery efficiency, as described above, and 2) its affordability, because unlike node replication, the effects of link replication on weight, mass, and power consumption are acceptable. Nonetheless, the dual CST_r scheme does not further improve the ability of the bus network in tolerating node failure. Accordingly, the qualitative failure-mechanism analysis presented in Section 3.2.2 and the quantitative reliability analysis presented in Sections 4.3 and 4.4 for the CST_r based bus network, are also applicable to the dual CST_r scheme.

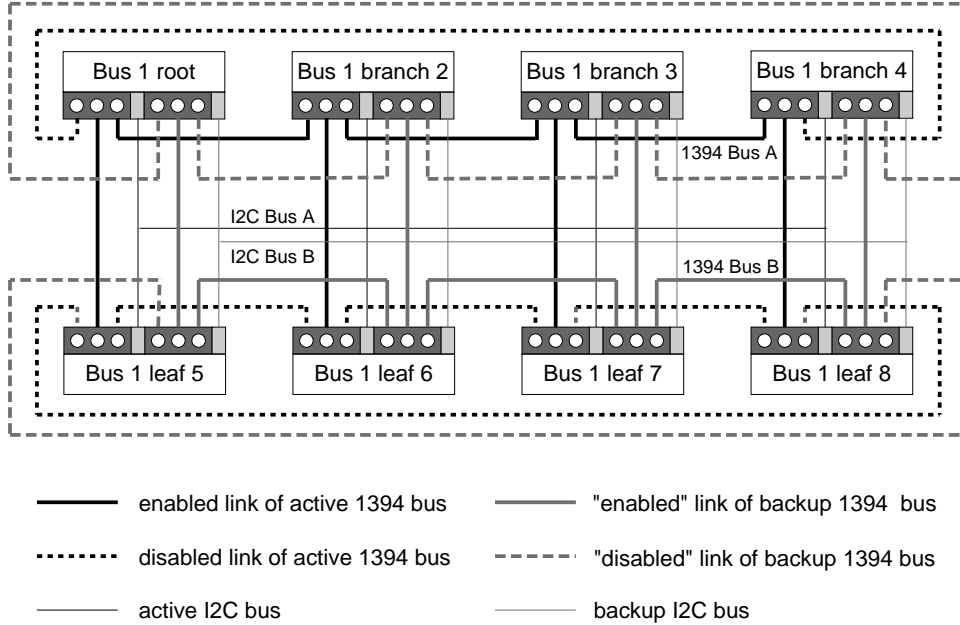


Figure 16: Dual CST_r Scheme

The fault-tolerant COTS bus architecture is implemented by a system I/O assembly [17], referred to as the System Input-Output (SIO) slice, which is depicted in Figure 17. The SIO slice is the interface gateway between the flight computers/microcontrollers and the IEEE 1394 and I2C buses. In particular, the main function of the SIO slice is to interface the IEEE 1394 and I2C buses with the internal PCI bus in a flight computer or microcontroller. The SIO slice consists of two redundant bus-interface units, each of which has a 3-port IEEE 1394 bus interface and two I2C bus interfaces. The bus-interface unit is composed of two ASICs: The Digital I/O (DIO) ASIC and the Mixed Signal I/O (MSIO) ASIC. The DIO ASIC implements the link layer of the IEEE 1394 bus, the two I2C bus controllers, and the fault tolerance enhancement control logic for the buses (fault detection/reconfiguration). The two I2C controllers are symmetrical so that either one of them can act as the role of a system bus or a subsystem bus. In addition, the DIO ASIC includes the functions such as the UART, to support software development and discrete input and output signals for miscellaneous control and monitoring. It is worth to mention that only a minimal percentage of the circuitry of the ASICs is dedicated to custom-designed fault tolerance logic. As indicated in Figure 17, in the DIO ASIC, only the portion encapsulated by the dashed oval is of custom-logic nature. The MSIO ASIC implements the physical layer of IEEE 1394 and the bus drivers of I2C. The MSIO ASIC is electrically isolated from the DIO and the rest of the flight computer or microcontroller. The purpose of the electrical isolation is to prevent the propagation of electrical faults between the bus and the nodes, and to mitigate grounding problems.

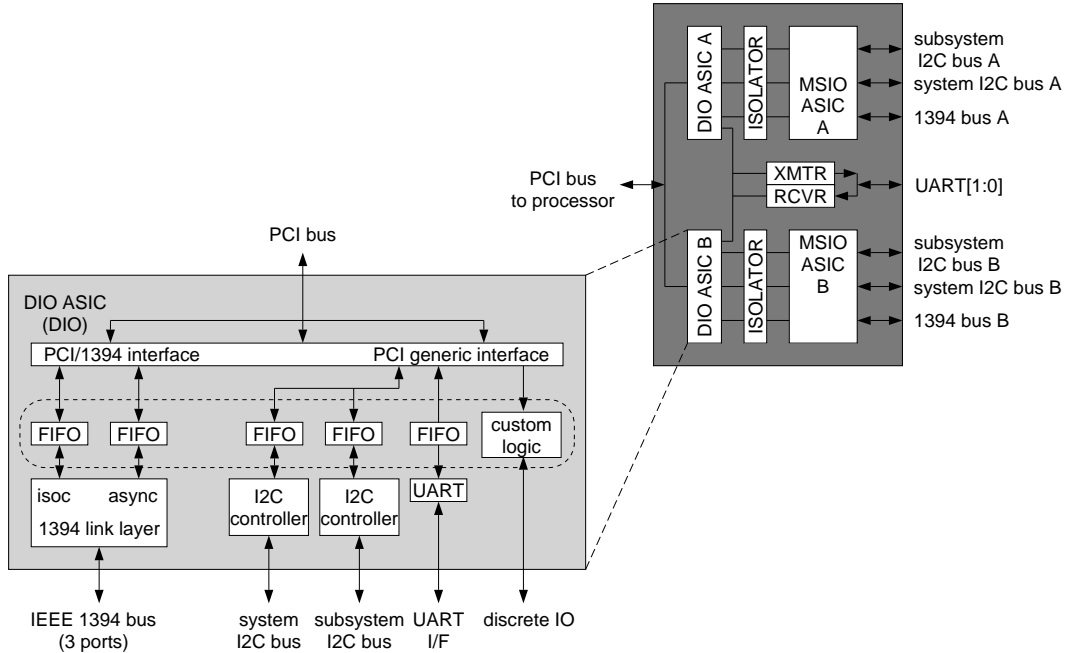


Figure 17: Hardware Implementation

The design and development of the ASICs described above have been completed by the X2000 team and ready for fabrication. The ASICs have a total gate-count (a complexity metric for ASICs) of 400 Kgates dedicated to logic circuitry. By maximizing the use of COTS IPs, the design and development of the logic circuits (400 Kgates) of the ASICs take only 11 person-years. On the contrary, the ASIC design for Cassini, the first NASA/JPL project that employed the ASIC approach, was conducted strictly in-house. The set of Cassini ASICs had an approximate total of 100 Kgates for logic circuitry and took 15 person-years to complete. Contrasting X2000 to Cassini¹, we can notice an approximate “per 100-Kgates” cost ratio of 2.75 to 15. Figure 18 contrasts the two projects in the above terms. Apparently, using COTS IPs for ASIC design and development enables us to develop avionics systems with higher complexity at a lower cost, relative to the approaches relying on custom design and development. This indeed demonstrates the importance of one of our bus standard selection criteria, that is, the availability of COTS IPs for implementing the bus (see Section 2). Without the available COTS IPs, knowledge acquisition and team training by themselves (for the detailed bus interface design) would have been prohibitively expensive. More importantly, our efforts in using COTS buses to develop mission-critical systems and exploiting COTS IPs for ASIC design indicate the following: The risks from using COTS technologies for development cost reduction can be managed, if they cannot be

¹Cassini and X2000 are the first and second NASA/JPL projects that employ the ASIC approach, respectively.

avoided. In particular, as exemplified by the qualitative and quantitative analyses presented in the previous sections, COTS-based fault tolerance is a viable and necessary means for minimizing and maximizing, respectively, the risks and the benefits from extensively using COTS technologies in mission-critical systems.

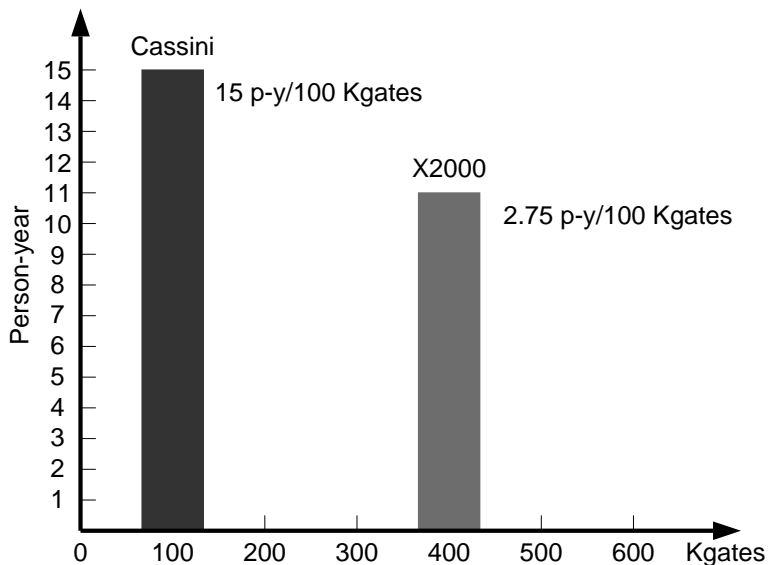


Figure 18: Cost Reduction from Using COTS Technologies

It is clear that our methodology for designing and implementing COTS-based fault tolerance is indeed a two-tiered approach, as illustrated in Figure 19:

1. We directly utilize the “built-in” fault detection mechanisms of the employed COTS products, wherever appropriate, and
2. We strive for *COTS-implemented fault tolerance*, which means that we build fault tolerance capabilities for the COTS-based system on the top of i) the standard features of the employed COTS products, and/or ii) the COTS products themselves (e.g., subsystem redundancy based on the COTS components which have diversified and complimentary architectural features and thus are unlikely to have common mode failure).

This approach enables us to minimize the custom design in a COTS-based fault-tolerant system and to maximize the benefits from using COTS [18]. With respect to the IEEE 1394 bus standard, its “built-in” fault detection mechanisms include error codes (i.e., data and header CRCs), acknowledgment packet, and timeout. Accordingly, we fully utilize those “built-in” mechanisms to implement the algorithms for node failure detection [9]. Moreover,

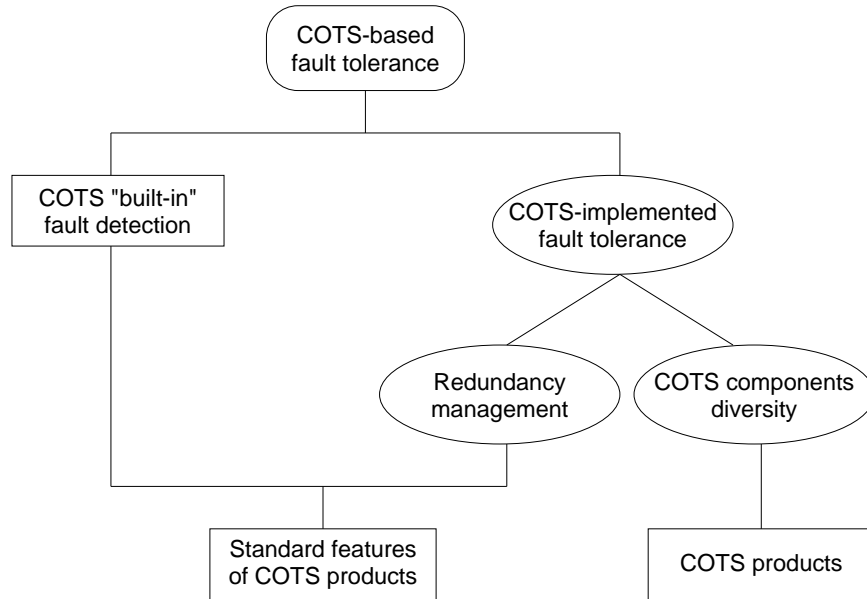


Figure 19: A Two-Tiered Approach to COTS-Based Fault Tolerance

rather than relying on custom-logic circuitry to compensate COTS products' lack of fault tolerance capabilities, we foster “COTS-implemented fault tolerance” in the bus architecture design. For example, the IEEE 1394 bus standard accommodates two kinds of transactions: isochronous transaction and asynchronous transaction. An isochronous transaction has guaranteed on-time delivery but does not require acknowledgment; an asynchronous transaction, on the other hand, requires acknowledgment but is not guaranteed to have on-time delivery. Since the cycle master (the root) of the 1394 bus normally sends an isochronous cycle-start message every $125 \mu\text{sec}$, our fault detection algorithms treat the cycle-start message as the heartbeat of the root node to monitor its health condition. And, since an asynchronous transaction will normally result in an acknowledgment from the destination node, we periodically issue asynchronous transactions and use them as polling messages to detect node failure (which will result in an acknowledgment timeout). As to COTS component diversity, our selection of I2C, which has topological characteristics differing from and complimentary to IEEE 1394, for the purpose of fault diagnosis and network reconfiguration during 1394 bus network partitioning, is an example of the “off-the-shelf diversity.” Finally, it is worth recalling that the port-disable feature of IEEE 1394 plays a key role in enabling the CST_r topology to meet the “no-loop” requirement from the COTS bus standard and to become an effective solution for fault tolerance.

6 Summary

We have presented qualitative and quantitative analyses of a COTS-based fault-tolerant bus network architecture. To implement COTS-based fault tolerance is becoming a major challenge for us today when cost concern has led to increased use of COTS products for critical applications. On the other hand, vendors remain reluctant to incorporate fault tolerance features into COTS products because doing so is likely to increase development and production costs and thus weaken the market competitiveness of their products. Therefore, to cope with the current state of COTS is crucial for us. Our analyses demonstrate that rigorous assessment and innovative utilization of pertinent standard features of a COTS product could enable us to circumvent its shortcomings and facilitate us to implement COTS-based fault tolerance for critical applications. Further, our effort reported in this paper promotes a two-tiered approach to the design and development of COTS-based fault tolerance. This approach enables us to minimize and maximize, respectively, the risks and the benefits from using COTS technologies in highly-reliable systems.

From analytic method point of view, the recursive function based model described in this paper enables us to obtain the exact solution of the reliability measure R_r that is otherwise very difficult to solve. Indeed, the recursive function based model can be adapted for the evaluation of the bus architectures which have particular constraints on reconfiguration and performance degradation. Aimed at further reliability and performance improvements, we are currently developing extended models for a comprehensive study of the CST_r based bus architecture, using more sophisticated measures, including performability.

Acknowledgment

The authors are thankful to Mr. Huy H. Luong and Mr. William Charlan of the Center for Integrated Space Microsystems at JPL who supplied useful data on the X2000 ASIC design and to the anonymous reviewers of the conference version of this paper for their helpful comments. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

- [1] L. Alkalai and A. T. Tai, "Long-life deep-space applications," *IEEE Computer*, vol. 31, pp. 37–38, Apr. 1998.

- [2] L. Alkalai and S. N. Chau, "Description of X2000 avionics program," in *Proceedings of the 3rd DARPA Fault-Tolerant Computing Workshop*, (Pasadena, CA), June 1998.
- [3] A. T. Tai, L. Alkalai, and S. N. Chau, "On-board preventive maintenance: A design-oriented analytic study for long-life applications," *Performance Evaluation*, vol. 35, pp. 215–232, June 1999.
- [4] L. Alkalai, "A roadmap for space microelectronics technology into the new millennium," in *Proceedings of the 35th Space Congress*, (Cocoa Beach, FL), Apr. 1998.
- [5] T. Shanley and D. Anderson, *PCI System Architecture*. Addison Wesley, 1995.
- [6] IEEE 1394, *Standard for a High Performance Serial Bus*. Institute of Electrical and Electronic Engineers, Jan. 1995.
- [7] D. Anderson, *FireWire System Architecture, IEEE 1394*. PC System Architecture Series, MA: Addison Wesley, 1998.
- [8] D. Paret and C. Fenger, *The I2C Bus: From Theory to Practice*. John Wiley, 1997.
- [9] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, "Design of a fault-tolerant COTS-based bus architecture," *IEEE Trans. Reliability*, vol. 48, pp. 351–359, Dec. 1999.
- [10] J. R. Marshall, "Building standards based COTS multiprocessor computer systems for space around a high speed serial bus network," in *Proceedings of the 17th Digital Avionics Systems Conference*, (Bellevue, Washington), Nov. 1998.
- [11] L. Alkalai, S. N. Chau, and A. T. Tai, "Networks based on stack-tree topology and a 1394 bus," *NASA Tech Briefs Journal*, vol. 24, p. 64, June 2000.
- [12] IEEE P1394A, *Standard for a High Performance Serial Bus (Supplement), Draft 2.0*. Institute of Electrical and Electronic Engineers, Mar. 1998.
- [13] J. Donaldson, "Cassini Orbiter Functional Requirements Book: Command and Data Subsystem," JPL Internal Document CAS-4-2006, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, June 1994.
- [14] C. S. Raghavendra, A. Avizienis, and M. D. Ercegovac, "Fault tolerance in binary tree architecture," *IEEE Trans. Computers*, vol. C-33, pp. 568–572, June 1984.
- [15] Y.-R. Leu and S.-Y. Kuo, "A fault-tolerant tree communication scheme for hypercube systems," *IEEE Trans. Computers*, vol. C-45, pp. 641–650, June 1996.

- [16] K. Sasidhar, L. Alkalai, and A. Chatterjee, "Testing NASA's 3D-stack MCM space flight computer," *IEEE Design & Test of Computers*, vol. 15, pp. 44–55, July-September 1998.
- [17] S. N. Chau *et al.*, "The implementation of a COTS based fault tolerant avionics bus architecture," in *Proceedings of IEEE Aerospace Conference*, (Big Sky, MT), Mar. 2000.
- [18] S. N. Chau, "Experience of using COTS components for deep space missions," in *Proceedings of the 4th IEEE International Symposium on High Assurance Systems Engineering*, (Washington, DC), p. 116, Nov. 1999.