Fault Tolerant Distributed Architectures for in-Vehicular Networks

Syed Misbahuddin
King Fahd University of Petroleum and Minerals Saudi Arabia

Nizar Al-Holou
University of Detroit Mercy

ABSTRACT

The increasing trend of automotive electronics mandates the introduction of multiple processors in automotive electronics. The automotive electronic systems have to operate in harsh environments having a high temperature range, high humidity, unpredictable vibrations and rapid voltage variation. In such environment, the automotive electronic systems become vulnerable to intermittent and transient failures. Depending upon the importance of the tasks performed by the processor, a processor's failure inside automotive electronic system may lead to serious consequences. Fault tolerant computing techniques are used to keep the computer systems running in spite of one or more processors' failures. The concept of fault tolerant is well known in many applications such as airplanes, industry, and military. However, the question of fault tolerant design has drawn little attention in automotive electronics. In this paper, various fault tolerant architectures for automotive applications have been proposed. In these schemes, fault tolerant is achieved by assigning the task of a failing processor to another processor in the system. In this way, the automotive electronic system may continue to function with multiple processors' failure.

I.INTRODUCTION

Electronics has been introduced into the automobiles to provide efficient implementation of all automotive functions. In the initial electronics implementation, a single processor called electronic control unit provides complete vehicular control. In order to improve the existing features and as well as to introduce new features, more processors are being introduced in the automotive electronic systems. When multiple processors are used in an automobile system, their failures may lead to the unavailability of the feature associated with the failing processor. Depending upon the importance of the failing processor, the automotive system may come to a complete halt. In the current state of the art, there is no fault tolerant architecture available in automobiles [18]. Because of cost and space restrictions associated with automotive electronics, special fault tolerant methods should be developed. In this paper, various fault tolerant distributed processing architectures for automotive applications have been proposed. In these proposed schemes, software-hardware based approaches have been proposed to address the fault tolerant issues in automobiles. In section II, single bus based architecture is out-lined along with fault tolerant algorithm for detecting faults in processors. Section III extends the idea of single bus based scheme to hierarchical distributed architecture. In section IV a multi-network scheme is presented. Finally, conclusion is discussed in section V.

II. SINGLE BUS BASED FAULT TOLERANT DISTRIBUTED PROCESSING ARCHITECTURE

Single vehicle wide networks provide many advantages such as economical multiplexing, flexibility in adding or removing control nodes and single communication protocols [1]. In a single bus-multiplexing network, all intelligent nodes are connected to the system bus through interfaces. In this section, a single bus based fault tolerant distributed processing architecture is proposed. This architecture allows a vehicular system to function in spite of multiple processor's failures. The fault tolerant is achieved by assigning the tasks of the failed processor to a functioning processor, which will continue its original tasks in addition to assigned tasks of the failed processor

ARCHITECTURAL FEATURES OF THE SINGLE BUS BASED ARCHITECTURE

The proposed system consists of n processing nodes (P_1, P_2, ... P_n ,) m sensor groups (SG_1, SG_2...SG_m) and k

smart sensor and actuators [2]. All processors are connected to the bus via network interface logic, which is subjected to the random errors [3]. The processing power of a processor becomes unavailable when processor and/or interface logic becomes faulty. This scheme uses a central controller unit (CCU). The central control monitors the performance of all processors connected to the automotive multiplexing bus. When detecting of a processing node's failure, the CCU executes a fault tolerant algorithm that assign the tasks of the failed processor to the another processor.

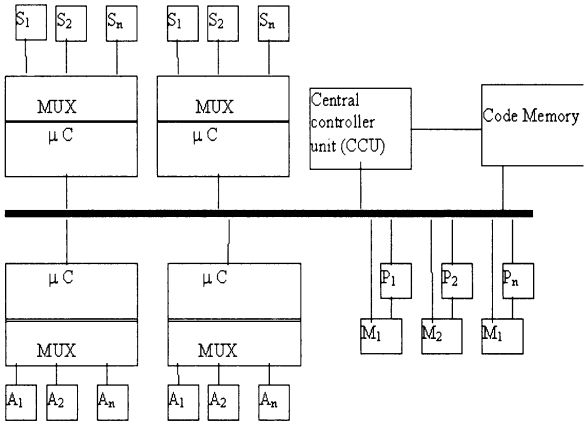


Figure 1: Block diagram of single bus based fault tolerant distributed processing architecture

In this proposed architecture, each processor is interfaced with two port memory modules. One port of each memory module is connected to its own processor and second port is to the main bus. Dual port memory allows any processor in the system to access any other processor's memory via the multiplexing bus without involving the processor. This feature of dual memory is conducive in implementing the proposed fault tolerant scheme discussed later. In this system, the code memory module shown in Figure 1 holds the segment of significant program codes for all processors.

FAULT DIAGNOSTIC ALGORITHM FOR THE PROPOSED SINGLE BUS BASED ARCHITECTURE

The proposed single bus based scheme will allow an automotive system to function in case of one or more processors' failures. In order to implement this scheme, the central controller unit (CCU) performs a supervisory action. During its supervisory action, the CCU identifies the faulty processor in the system and takes the appropriate actions to keep the system running. The CCU uses a variable called processor index (*PINDX)*, which points to an *ith* processor at a given instant of time. The CCU sends a diagnostic message periodically to all processors in the system indicated by *PINDX*. If a processor is not faulty, it will respond to the CCU's acknowledgment message within a predefined interval of time then, it will mark that processor as faulty processor and will assign the tasks of the faulty processor to another processor performing frivolous tasks in the system. The CCU can transfer the critical program code of the faulty processor to the assigned processor's memory by accessing the faulty processor's memory via multiplexing bus. Alternatively, the assigned processor may directly execute the critical code of the faulty processor by reading it from the code memory of the faulty processor. For the first option, the assigned processor's memory will be partitioned into two parts in such a way that half of it will hold the critical program code of the faulty processor and the other half will hold the program code of the assigned processor. The assigned processor will continue performing its original tasks in addition to this new assignment on a time sharing basis. The assigned processor can access the sensors and actuators related to the faulty processor via the serial bus. The fault diagnostic algorithm executed by the central controller is summarized in the flow chart shown in Figure 2.

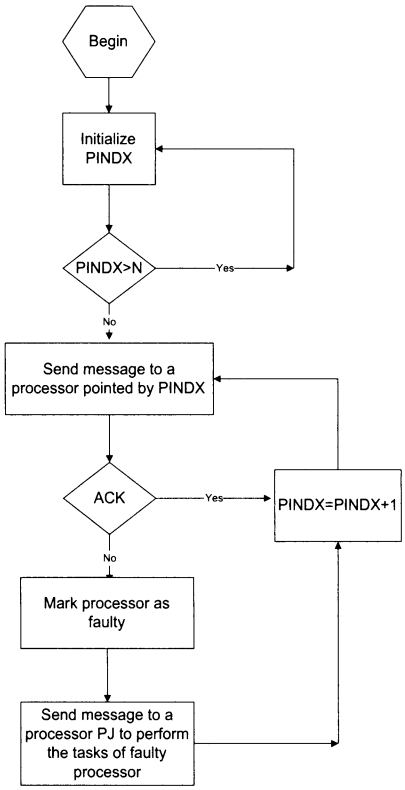


Figure 2 Fault diagnostic algorithm performed by the central controller unit

In the proposed scheme, it is assumed that each processor has the potential of executing the task performed by other processor. Therefore, no redundant processors are needed to implement the fault tolerant

approach is cost effective in the sense that the fault tolerant capability is achieved by software and a limited hardware. The failure of the CCU will be catastrophic for the operation of the whole system because in this situation failure in any processor will not be detected. In order to avoid this problem, a single line called the central controller's active line (CCA) can be used as proposed in [4]. As long as the CCU is not faulty, an active high signal will be available on the CCA. Whenever this active high signal becomes low, a watchdog timer will become active. If the high logic level does not appear on the CCA within a defined time period, the watchdog timer will interrupt any of the processors in the system to takeover the responsibilities of the CCU. The assigned processor will continue its original assignment on a time-sharing basis

III. DEVELOPMENT OF A HIERARCHICAL DISTRIBUTED PROCESSING SYSTEM (HDPS)

In single bus based fault tolerant distributed system discussed in section II, all processors in the system are connected to one global bus. In this situation, the global bus becomes congested when the data traffic increases. Global bus then becomes the bottleneck for the whole system and its failure will bring the whole system to a halt. To overcome this problem, a hierarchical distributed processing system is proposed in this section in which the concepts of global and local buses have been used. This scheme is based upon the classification of automotive system into functional subsystems. The automotive electronic systems can be divided into functional subsystems according to their physical locations and functions as follows [5]:

1. Vehicle Drive Control Group (VDCG): This group may include the engine control, transmission control, cruise control, suspension control, steering control, throttle control, traction control, four wheel steering control and knock control.
2. Intelligent and Security Group (ISG): This group may include the air bag control, automatic collision avoidance and notifier system, ABS, engine immobilizer control and lojack system.
3. Intelligent Transportation System Group (ITSG): This group provides support for the intelligent transportation System (ITS). The ITSG may include the navigation computer and ITS support control [6].
4. Body Control Group (BCG): this group may include the instrument cluster control, trip computer, climate controller, tachometer and fuel gauge control.

THE ARCHITECTURAL FEATURES OF HDPS

Based on the classification of the automotive electronic system, a hierarchical distributed processing system has controller unit (CCU), a code memory module and *N* functional subgroups (G_1-G_N) as shown in Figure 3. Each subgroup also contains one special purpose processor called coordinator processor (*CP*). The purpose of the coordinator processor is to provide communication facility between processors located in different subgroups. Also, the coordinator processor in each group can provide the performance history of all processors inside the corresponding subgroup.

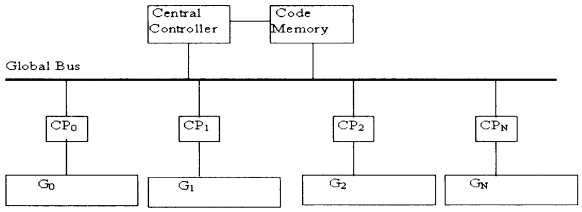
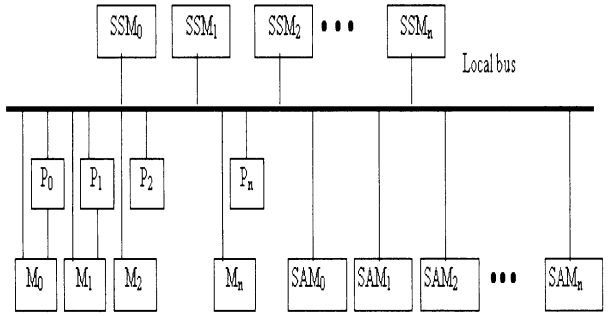


Figure 3: Architecture of proposed hierarchical distributed processing architecture

A subgroup G consists of a smart sensor group, a smart actuator group, a number of homogenous processing elements and a local bus as shown in Figure 4. All processors within a subgroup communicate with each other through the local bus. The processors in different subgroups can also communicate with each other through the global bus.



SSM = Smart sensor module
SAM = Smart actuator module
P = Processor

Figure 4: Architecture of a typical subgroup in the proposed hierarchical distributed architecture

FAULT DIAGNOSTIC ALGORITHM IN THE HDPS

In automobiles, if a processor in a significant subsystem fails then the whole automotive system may fail. In order to avoid complete failure of the system, there should be a fault diagnostic and fault tolerant algorithm for the system. In this section, a fault diagnostic algorithm is presented for the proposed HDPS. The central controller unit (CCU) implements this algorithm shown in Figure 5. In order to implement the fault diagnostic algorithm, the CCU sends diagnostic messages periodically to all

PINDX points to a processor in a group pointed by *GINDX*. At the beginning of the algorithm, the variable *GINDX* is initialized to point to the first group in the system. The algorithm tests whether *GINDX* is greater than *NG*, total number of groups in the system. If *GINDX* is found greater or equal to *NG*, the *GINDX* is initialized to point to the first subgroup. On the other hand, if *GINDX* is not equal to *NG*, then the variable *PINDX* is initialized to "0." In this case, *PINDX* points out to the very first processor in the subgroup pointed by *GINDX*. A diagnostic message is sent to the processor pointed out by *PINDX* in a subgroup indicated by *GINDX.* Before sending the diagnostic message, the CCU checks whether *PINDX* has become greater than *NP*, which is the number of processor in a group pointed by *GINDX.* If so, the CCU will reset *PINDX* to "0" and the pointer *GINDX* is incremented and control is transferred to another group. If CCU has not sent messages to all processor in a subgroup, then it will send the diagnostic message to the processor pointed by *PINDX* in the subgroup. CCU will anticipate an acknowledgment message from the processor within a specified interval of time. If an acknowledgment is not received from a processor within certain interval of time, the CCU assumes that the processor is faulty. In case of a processor's failure, the CCU assigns the tasks of the faulty processors to another processor in the same subgroup. The assigned processor continues its original assignment on a time sharing basis.

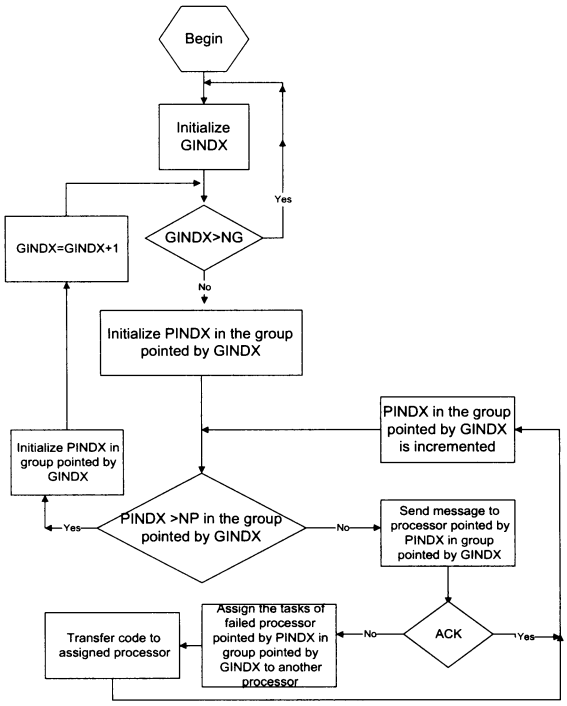


Figure 5: Fault diagnostic algorithm performed by the control unit in HDPS

FOR AUTOMOTIVE APPLICATIONS

The global bus in the hierarchical distributed system discussed in section III may pose the same limitation as indicated in single bus based architecture. That is, a failure in the global bus may lead to system's malfunctioning. To avoid this situation, a multiple network distributed processing system (MNDPS) is proposed in this section. Different groups of an automobile system may need different bus speeds. In order to accommodate this need, multiple buses can be introduced in automobiles. These buses are connected with each other through bridges. Figure 6 shows the proposed multiple network scheme for automotive applications. Each sub network consists of necessary processors, smart sensors and smart actuators. A coordinator processor (CP) is included in each sub network. Processors within a sub network communicate with each through local bus. For the communication among various sub-networks, the processors can use coordinator processors and bridges. Because of the multiple bus characteristics, individual sub networks can contain individual protocols. In this scheme, no global bus has been used. This feature eliminates the bottleneck of global bus failure in the proposed HDPS. In the proposed MNDPS a central controller unit (CCU) is attached to one of the sub networks. This sub network is called as supervisory sub network (SSN). The SSN contains a code memory which holds the critical program codes of all processors. The CCU sends diagnostic messages to all processors in the whole system. If a processor is found malfunctioning in a sub network, the CCU can assign the task of the faulty processor to any other processor in the same sub network. The assigned processor continues its original assignment on a time sharing basis. The critical program code of the faulty processor is transferred to the assigned processor. The fault diagnostic algorithm proposed for HDPS can be applied for MNDPS without major changes.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.