IMPROVED SYSTEM ARCHITECTURE FOR SAFETY-RELEVANT SYSTEMS USING DYNAMIC DISTRIBUTION AND STATE BUFFERING

Philipp Nenninger * **Oliver Rooks** * **Uwe Kiencke** *

* *Institute of Industrial Information Technology, Universität Karlsruhe (TH), Germany*

Abstract: Drive-by-wire systems are gaining ground in the automotive industry and approaching maturity. In this paper the state of the art is presented, which is a static Duo Duplex system and focus on two crucial components, the input-management and the output-management. Additionally a novel architecture based on fail silent units which uses dynamic distribution of functions within the system is proposed. In order to eliminate the transitional phase in controller functions, a fault-tolerant *State Server* is introduced, which stores the states of all safety-relevant functions.

Keywords: Distributed Systems, Fault Tolerant Systems, Drive-by-Wire, State Server

1. INTRODUCTION

The integration of drive-by-wire technology into the field of vehicle construction provides the opportunity to improve the interior space of the vehicles as well as to implement innovative driver-assistance systems in order to increase road safety (Spiegelberg, 2002). In the last ten years by-wire systems in combination with mechanical backup (throttle-by-wire, Sensotronic Brake Control, Active Steering) were developed and integrated in the large-scale production, whereas complete electrical systems are still the focus of research. These systems generate electrical commands from the driver's inputs and transfer them to computer controlled actuators. Usually no safe state exists in the case a failure occurs. For these systems it is necessary to apply fault-tolerant components, where as today's requirements ask for the compensation of at least one safety relevant fault (Isermann *et al.*, 2002).

In the automotive field the application of the so-called *fail-silent strategy* seems to be appropriate. If an error occurs in an electrical or mechatronic part of the drive-by-wire system, this component will stop its communication. Thus a mapping of all faults to a unique observable symptom takes place, which proves to be a great advantage.

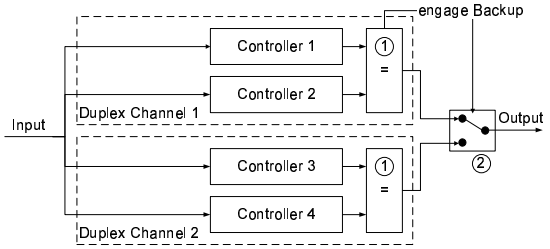

Fig. 1. Duo Duplex Structure

Applying this strategy, different authors have chosen to implement a drive-by-wire computer system following the so-called duo-duplex structure (Rooks *et al.*, 2003) (Armbruster *et al.*, 2004). Its basic implementation is shown in figure 1. It con-

sists of four ECUs, divided into two duplex channels which behave fail-silent (Hammett, 2002). Each ECU processes the same input data simultaneously. Two comparison units (see (1) in figure 1) observe the accordance of the outputs within the channels. In a faultless situation the first channel is active and communicates with its environment. If an error occurs inside the first duplex system the switching unit (see (2) in figure 1) will activate the previously passive channel.

Considering a complete drive-by-wire system several of these computer nodes build up a distributed system. This leads to the situation that the overhead in utilized microcontrollers reaches 300 % of a non redundant simplex system. On the way to series production reducing redundancy will prove to be a major challenge. In this paper the authors outline, that dynamic distribution of vehicle functions can reduce the level of redundancy under certain circumstances.

This article is organized as follows: Section 2 provides an overview of a state of the art duo duplex system. It describes a software based approach applying COTS ECUs as well as CAN[1] communication. The basic software structure used to handle the input data as well as the monitoring of the output data is addressed.

In section 3 the focus is on the advantages using dynamic distribution of vehicle functions to reduce the degree of hardware redundancy. It is shown, that the main challenge is not the distribution of the functionality but the safe storage and exchange of state information.

Finally section 4 summarizes the paper and provides an outlook to future work.

2. STATE OF THE ART

2.1 Overview

The following description of a software based Duo Duplex System is mainly taken from (Rooks, 2004). Firstly the basic hardware structure is introduced. Additional the problem of safe communication interruption in case of a component failure is addressed. Finally several parts of the redundancy managements are discussed, which operate the Duo-Duplex System. The objective is the description of the mechanisms necessary to build up a fail-silent unit.

In figure 2 the software based Duo Duplex system is shown. It consists of two fail-silent units, each comprising two electronic control units (ECU). Furthermore each fail-silent unit includes an additional hardware component called BUSPWR

[1] Controller Area Network

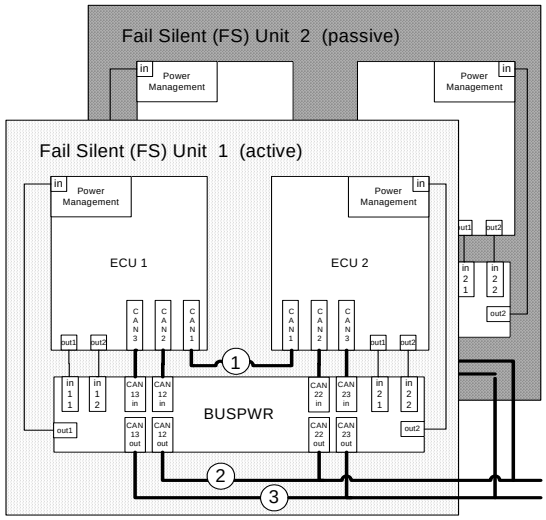

Fig. 2. Structure of software based Duo Duplex System

block. The communication among the ECUs as well as between the ECUs and their environment (sensors, actuators) is based on CAN communication.

As shown in figure 2 the ECUs of one fail-silent unit communicate via a private bus (marker (1) in picture 2) , which helps to reduce the communication load on the external CAN buses (marker (2) and (3) in picture 2), which are used to exchange information with the environment of the Duo Duplex System. To avoid loss of function due to the failure of one external CAN bus the external communication has to be redundant.

In case of a failure of one of the ECUs the communication ports of the faulty component have to be interrupted. It cannot be guaranteed that it fails silent in every failure situation. For this reason an additional hardware component, the BUSPWR block, is introduced. It is designed following the so-called *Switching Signal Path Structure* (Rooks, 2004) allowing for the detection of latent component failures. For this reason every single failure transfers the BUSPWR block into the safe state *communication interrupted*.

It is obvious that a mechanism is required, which generates one valid output data out of four processing results. To succeed in this the ECUs are ordered by priority represented by *controller numbers* shown in table 1. At first it is necessary to distinguish between an active and a passive fail-silent unit. The active fail-silent unit is entitled to send output data to the environment. Computation results of the passive fail-silent unit do not leave the Duo Duplex system. Within one fail-silent unit there is also a difference in the behavior of the computer nodes. Passive nodes only generate data for inner channel comparison purposes. As shown in table 1, only the active computer node in the active fail-silent unit (controller number 1) is

Table 1. Controller Numbers

Fault Free	ECU	Fail-silent Unit	Final Output
1	active	active	Yes
2	passive	active	No
3	active	passive	No
4	passive	passive	No

allowed to send information to the environment of the drive-by-wire system. Additionally the active node of the passive channel (controller number 3) transmits an alive message via channel two and three. In case a computer node detects its malfunction the controller number is replaced by the value 5.

After this introduction of the redundant hardware structure, the focus is on the software operating the computer system. It is implemented on every ECU and comprises three parts introduced in the next sections.

2.2 Input-Management

It is the task of the input-management to provide synchronized input data to the control functions of the drive-by-wire computer system. This is an important issue especially if the monitoring of the ECUs' output is realized by bitwise comparison. This way to supervise the correct processing of the ECUs can be evaluated according to the attributes listed in table 2.

Table 2. Comparison of bitwise Monitoring and Monitoring with Tolerance

Attribute	Monitoring with Tolerance	Monitoring bitwise
Fault Containment	not guaranteed	yes
Fault Detection	delayed	immediate
Boolean Output	no	yes
Coding/Compression	under certain conditions	yes
Synchronized Input	not necessary	necessary
Diversity in Hardware	yes	under certain conditions
Diversity in Software	yes	no

The main advantages of bitwise monitoring are the guaranteed fault containment and the immediate fault detection. Additionally the coding and compression of the processing results of each ECU is possible as well as the handling of boolean outputs, where no tolerances can be defined. The major disadvantage, compared to monitoring with tolerances, is the renunciation of diversity in software.

When using bitwise monitoring it is necessary to guarantee synchronized inputs for each ECU of one fail-silent unit. Under these circumstances the output values of the ECUs are identical in the fault-free situation. To achieve synchronized inputs a combined approach was chosen based on the clock of one of the ECUs generating time intervals and a counter implemented in both ECUs counting the number of CAN messages of the present time interval. Therefore a master-slave configuration of the fail-silent units' ECUs has to be established. After an initial synchronization resetting the counters and defining the first input message the master determines the limits of the time intervals. At the end of each time interval it calculates the number of received input messages of the CAN buses. This information is sent to the neighboring ECU, which generates an identical data set from the received number of CAN messages. At this point robustness against lost messages is a key issue of this mechanism.

2.3 Output-Management

The output management concentrates on the failure detection comparing the processing results of the ECUs of each fail-silent unit. Figure 3 shows the algorithm applied. Some operations are linked to a specific position of the ECU in the redundant controller. In the following the this process is explained in detail. The numbers in brackets correspond to the flowchart in figure 3.

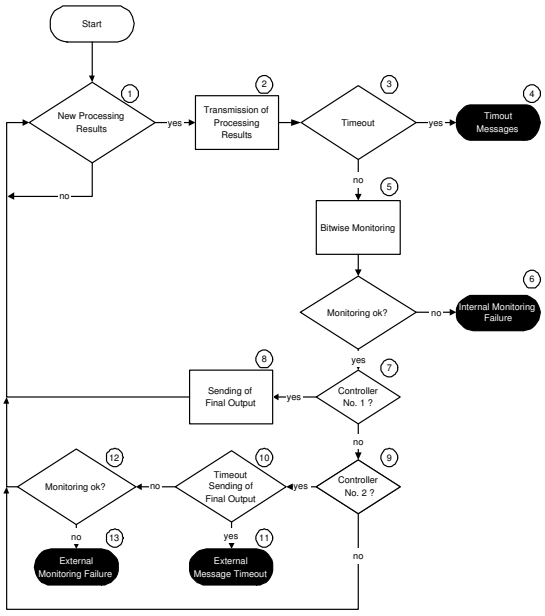

Fig. 3. Monitoring Process

As soon as a new processing result of the control functions is available (1), it is transmitted via CAN1 to the neighboring ECU (2). Afterwards the ECU observes CAN1 waiting for the corresponding data coming from the neighbor (3). After its reception both computer nodes are aware of the both results. If the corresponding data of

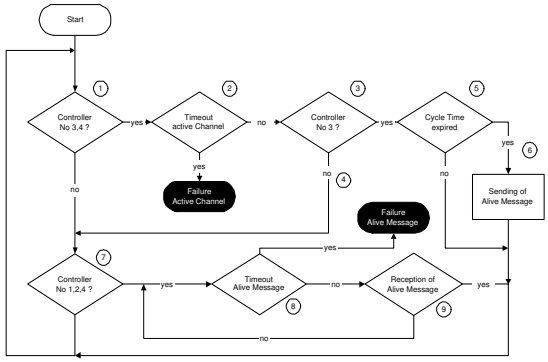

Fig. 4. Alive Process

the neighbor is not received within a predefined period of time, a timeout will occur which is called *Timeout Message* (4). In case the processing result of the neighbor is received in time the process will monitor the processing results (5). If these results do not match the *internal Monitoring Failure* will emerge (6). The process will continue according to the controller number if the comparison is successful:

ECU 1 (7): The ECU with the controller number 1 transmits its verified output data to the environment via the buses 2 and 3 (8). After this operation the computer node awaits new processing results of the corresponding control function (1).

ECU 2 (9): After a successful inner channel monitoring this ECU observes the reception of the first ECU's final output on the CAN buses 2 and 3 (10). If the data is not received within a predefined period of time, *External Message Timeout* will occur (11). Otherwise the data is compared to the result of ECU1 known from the inner channel monitoring (5). In the case of discrepancy the process terminates in the error state *External Monitoring Failure* (13). If the values match the process will wait for new processing results of the corresponding control function (1).

ECU 3 and 4 (9): After their inner channel comparison (5) the monitoring process terminates and both ECUs wait for new processing results (1).

Besides the inner channel monitoring of the processing results each fail-silent unit has to be aware of the health state of the neighboring fail-silent unit. Depending on this information the passive fail-silent unit is activated respectively information about the failure of the passive fail-silent unit is passed to the environment of the duo duplex system. This so-called *alive process* is shown in figure 4.

To be aware of the current situation of the duo duplex system it is necessary to identify the loss of the passive backup fail-silent unit. The alive process is implemented on each of the ECUs but its functionality depends on the current controller number. The main goal is to send a special CAN message - the so called *alive message* - from the passive fail-silent unit to the active one. As long as no error inside the passive fail-silent unit is detected, the ECU with the controller number 3 transmits the alive message via the communication channels 2 and 3 (6). The ECUs of the active fail-silent unit receive these messages (9). In the case they are not received within a predefined period of time (8), the ECUs with the controller number 1 and 2 assume the failure of the passive fail-silent unit. This situation has to be reported to the environment of the duo duplex system, because the fault-tolerant properties of the redundant drive-by-wire controller are lost. Besides the monitoring of the state of the passive fail-silent unit the health state of the active fail-silent unit is supervised. This is the task of the ECU with the controller number 3 and 4. If there were no final messages on the CAN buses 2 and 3 a failure of the active fail-silent unit is assumed (2). In the following the passive fail-silent unit activates itself by changing the controller numbers of the ECUs.

3. ADVANTAGES OF DYNAMICALLY DISTRIBUTED SYSTEMS

The static redundancy presented in the previous section provides a fail tolerant architecture which is fulfills the requirement of being able to tolerate one safety critical fault. However this comes at the price of having a large overhead. In the following section, a new approach which allows for a considerable reduction of this overhead will be presented. Each fail safe unit comprises two independent microcontrollers (μC) which need not be distributed to different ECUs as presented in section 2.1.

Dynamically distributed systems offer a substantial advantage over statically distributed systems: In case of an error of hardware or software, a fault free module can, under certain conditions, take over the operation of the faulty one. These conditions are the following:

- The function[2] which is to be distributed does not rely on sensor information or access to actuators which is exclusive to certain μCs or fail silent units. With intelligent networked sensors and actuators becoming more prevalent, this is an assumption which does not limit the overall functionality of the system or largely increases its cost. This is also a prerequisite for the original fail silent units presented before.

[2] A *function* is defined as the sum of all the independent sub-functions originally executed by one fail silent unit.

- The fault free fail silent units have enough resources to run the function under consideration. This might be achieved by designing the system with these additional resources or by dropping less important functions and accepting degradation.
- A function can be divided into several small independent sub-functions which can be distributed to different fail silent units without a loss of functionality. This is not a hard prerequisite but limits the need for extra resources.
- Because execution speed rather that code size limits the number of sub-functions on a μC, the code for every sub-function in the network can already be present on a sufficient number of μCs. This constraint also helps to keep the bus load low which, in case of the CAN bus, is necessary to keep the bus almost deterministic (Ellims *et al.*, 2002).

With the above requirements fulfilled, dynamical distribution yields a large improvement in cost. Considering a safety-relevant system as shown in fig. 5. It consists of networked sensors and actuators and 4 homogeneous fail silent units. It is not a prerequisite for the fail silent units to be homogeneous, but for clarity's sake this assumption will be made during the derivation of the proposed architecture because it allow to introduce the term *normalized speed*. A fail silent unit has the normalized speed of 1 if it can execute exactly one function, assuming that all the functions within the system require the same amount of computational power.

In case of an single error, a fail silent unit will fail

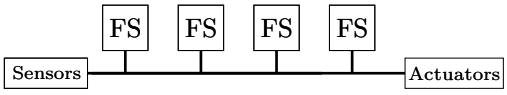

Fig. 5. Safety-Relevant System with Dynamical Distribution

silent and the other units will replace it by executing the necessary functions. Assuming that the faulty fail silent unit executed several independent functions and that the cost of hardware increases roughly proportional with its capabilities, only 4 instead of 8 fail silent units are needed. Because each of these units has to execute one third of the functions of the faulty one and therefore has to have the normalized speed of 1.33, dynamical distribution reduces the cost of the system by a factor of 0.67. The savings become more relevant if more fail silent units are involved.

The reason behind this is that a system with dynamical distribution only provides one backup unit for the entire system which is hidden in the unused capacities (normalized speed >1) of the fail silent units while a system consisting of several Duo Duplex systems has a dedicated backup fail silent unit for each operational fail silent unit. Because automotive drive-by-wire only has to be fail-operational after one safety-relevant failure and common-mode faults which lead to simultaneous errors in more than one fail silent unit can be considered to be rare when using the necessary precaution (Collinson, 1999), a dynamically distributed system as described above fulfills the requirements for simple input-output systems like a Medvedev FSM. An automotive example for this would be a function which allows for easier parking by multiplying the steering wheel angle and a velocity dependent factor. A sudden loss of this function could lead to a jerk in the wheel turn angle. On the other hand, the function can be restarted on another μC without knowledge of the previous values.

3.1 Dynamic Distribution using State Buffering

For most control applications however, the state contains crucial information. Starting a backup unit from cold redundancy as proposed above would therefore result in a transient phase which for most drive-by-wire applications is not acceptable. Hence a mechanism to store the system states outside of the fail silent units is needed (see fig. 6).

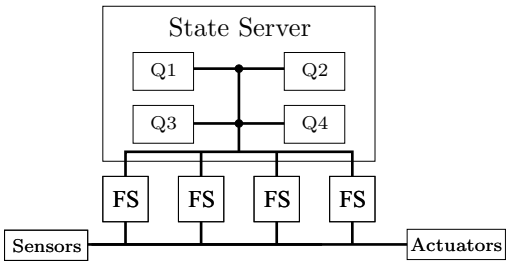

Fig. 6. Safety-Relevant System with Dynamical Distribution

The proposed system architecture not only comprises the 4 fail silent units needed to build the dynamic system in fig. 5 but also a *State Server*. This State Server mirrors the states of all the functions running on the fail silent units. In order to ensure the integrity of the data, an internal state matching is conducted. Because of the crucial importance of the State Server, a quadruplex structure is suggested, so the State Server can tolerate 2 faults.

The units within the State Server are dedicated systems since their main function is to record and verify the state information which is sent over the CAN buses and they do not have the computing power to execute sub-functions. Because of the high volume of data created by state information,

connecting the State Server to the fail silent units via an internal CAN buses will help to minimize the load on the main CAN.

Because the system only has to tolerate one safety-relevant fault, the μC capability needed for each architecture can be calculated by:

$$n_{DuoDuplex}(x) = 4x$$
$$n_{Dynamic}(x) = 2x\frac{x}{x-1} + 4$$

Table 3. Requirements for the Different Architectures

	μCs per Lane	Normalized Speed per μC	μCs in State Server
Duo Duplex	4	1	0
Dynamic Redundancy	2	$\frac{x}{x-1}$	4

with x being the number of functions. The two functions are plotted in fig. 7. The figure shows that the break-even for the two architectures is between 3 and 4 functions.

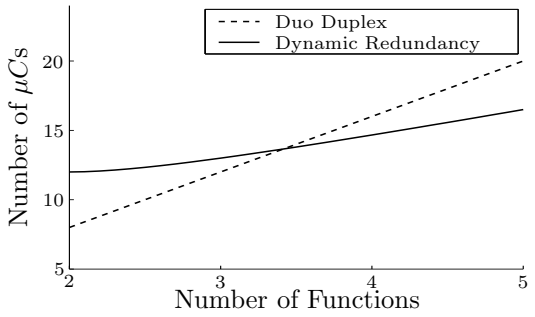

Fig. 7. Number of μCs with the normalized speed 1 needed if one safety-relevant error must be tolerated

3.2 Comparing Duo Duplex and Dynamic Distribution using State Buffering

While a fault causes a system made up of Duo Duplex systems to go into a state in which one additional fault affecting the backup channel of the Duo Duplex system already affected by a fault can lead to a catastrophic failure, the dynamic architecture allows for graceful degradation by stopping not safety-relevant functions in order to free capacity and using previously unused capacity to run safety-related functions. In the case of a car, functions which may be stopped include automatic shifting and ESP. Dropping these functions will lead to less comfortable driving and will limit the functionality of the car, but combined with an alert for the driver will allow him to reach a safe state, for example by stopping on the curb.

4. CONCLUSION

The integration of drive-by-wire systems in future cars require electronic systems which are able to be operational after one safety-relevant fault and fail silent after the second safety relevant fault. A static software based Duo Duplex system which represents the state of the art can provide this functionality at the price of a high level of hot redundancy.
Because the automotive sector is especially cost sensitive, the approach of using dynamic redundancy in combination with a unit which eliminates the transient phase of cold standby by buffering the state information of the implemented control functions while relying mainly on COTS components is presented. This not only decreases cost if more than 3 fail silent units are used, but also allows degradation of system functionality by stopping non safety relevant functions and therefore freeing resources for safety relevant ones.

REFERENCES

Armbruster, M., K. Bäuerle, R. Reichel, A. Maisch and G. Spiegelberg (2004). X-by-wire system of the next generation. In: *7th International Symposium on Advanced Vehicle Control (AVEC 04)*. Arnhem, The Netherlands. pp. 135–140.

Collinson, R. P. G. (1999). Fly-by-wire flight control. *Computing and Control Engineering Journal* pp. 141–152.

Ellims, M., S. Parker and J. Zurlo (2002). Design and analysis of a robust real-time engine control network. *IEEE Micro* **22**, 20–27.

Hammett, R. (2002). Design by extrapolation: an evaluation of fault tolerant avionics. *IEEE Aerospace and Electronics Systems Magazine* **17**(Apr 2002), 17–25.

Isermann, R., R. Schwarz and S. Stölzel (2002). Fault-tolerant drive-by-wire systems. *IEEE Control Systems Magazine* **22**, 64–81.

Rooks, O., M. Armbruster, S. Büchli, A. Sulzmann, G. Spiegelberg and U.Kiencke (2003). Redundancy management for drive-by-wire computer systems. In: *22nd International Conference, Safecomp 2003*. Edinburgh, UK. pp. 249–262.

Rooks, Oliver (2004). Softwarebasierte Sicherheitsfunktionen in Drive-by-Wire Fahrzeugrechnern. Dissertation. Universität Karlsruhe (TH).

Spiegelberg, G. (2002). Ein Beitrag zur Erhöhung der Verkehrssicherheit und Funktionalität von Fahrzeugen unter Einbindung des Antriebstrangmoduls $MOT_{ion}X - ACT$. Dissertation. Göttingen.