

A Comparison of Dual-Core Approaches for Safety-Critical Automotive Applications

Eva Beckschulze, Falk Salewski and Stefan Kowalewski

Embedded Software Laboratory, RWTH Aachen University, Germany

Copyright © 2009 SAE International

ABSTRACT

Safety is a requirement concerning an increasing number of automotive applications. Recent safety standards set requirements for designing safety-critical systems. Among others, these specifications include a comprehensive detection and handling of hardware faults. Currently emerging dual-core microcontrollers provide a cost-effective opportunity to fulfill these requirements. In this paper we analyze a safety-critical application example and discuss two different approaches, an application-specific approach and a generic approach for implementing functional safety requirements on a dual-core microcontroller. An investigation of the associated concepts called *function monitoring architectures* and *generic architectures* reveals their differences and at the same time advantages and disadvantages. Besides effects on safety, effects on reliability, modifiability and costs are evaluated and presented graphically.

INTRODUCTION

Today more and more automotive electronic systems are implemented to take over complex tasks. In many cases a failure of a system is not admissible as potentially arising hazardous situations can lead to major accidents. Therefore, even for simple applications extensive actions have to be taken to reduce the residual risk to an acceptable limit. For future automotive systems, the safety standard ISO 26262 will be the

guideline used to determine the extent of the necessary safety functions. As the development of the standard ISO 26262 is still in progress, in this paper information is based on the working draft [1].

There are many ideas addressing the problem of fault handling in programmable devices, particularly for microcontrollers. In safety-critical automotive applications there are two major challenges which have to be dealt with. The first challenge is the cost pressure the automotive industry is forced to bear resulting in the need for low cost hardware. However, the general concept to achieve functional safety is redundancy. In case of hardware faults hardware redundancy is required to a certain degree. Multi-core microcontrollers, in particular dual-core microcontrollers, which are introduced in embedded systems in order to meet the increasing demand for performance, are also a promising option for safety-critical applications. Therefore, we focus our study on dual-core microcontrollers in this paper.

The second challenge is the compliance with real-time requirements. Those are not only given by the functional specification but also by safety requirements. This means that a safety measure will only be effective if it is executed once in the defined period. Such requirements prove difficult for single point failures. Under the terms of ISO WD 26262, a single point failure results from a single fault and can without a further fault directly lead to a hazard. There are different solutions to this problem.

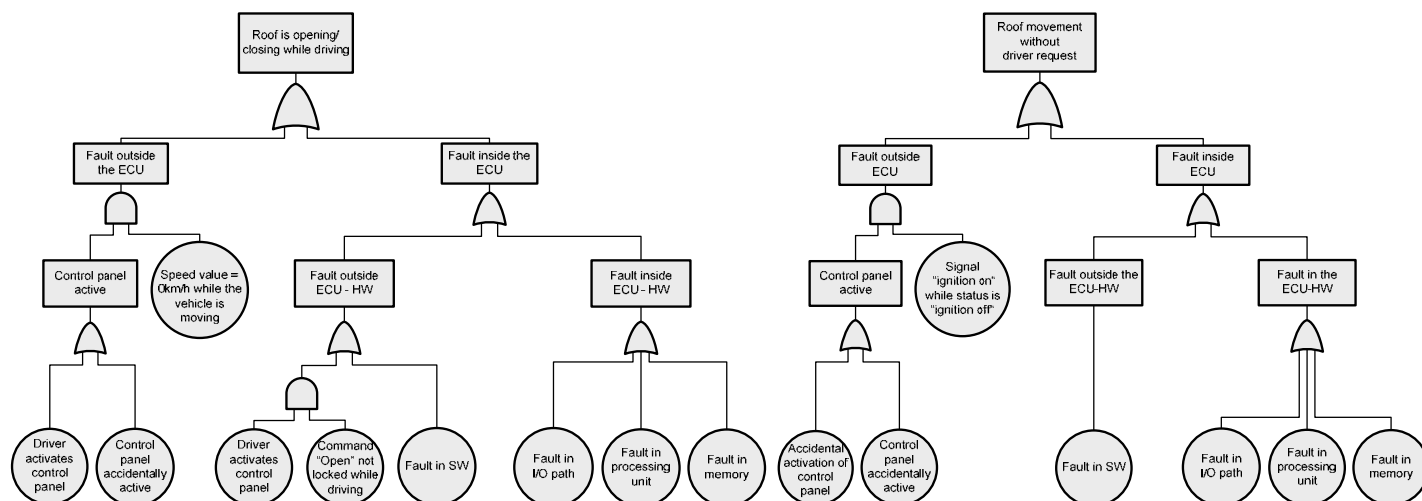


Figure 1: Fault tree analysis for safety goal 1 (left) and 2 (right)

In [2] we presented concepts for exploiting dual-core architectures for fault-handling in automotive control units. In this paper, we extend this work by detailing these concepts and evaluating the resulting methodology to an elaborated application example. By adopting the procedure provided in ISO WD 26262 we specify functional safety requirements for the application example. Subsequently, two different technical implementations on a dual-core microcontroller are proposed. In the second part we compare the two concepts to each other and point out their advantages and disadvantages with regard to different aspects.

APPLICATION EXAMPLE

In this section, we analyze a controller for a convertible hard top to exemplify a safety-critical automotive system. The considered system interacts with the user through a dedicated control panel triggering a hydraulic pump for starting the movement of the convertible top. The processes "open" and "close" compose the main function of the controller including the control of a number of sensors and actuators as well as the interaction with other control units.

On the one hand, potential hazards initiated by the system include the convertible top breaking away while driving at high speed and on the other hand, injuries induced by the movement of the convertible top.

According to ISO WD 26262 the first goal towards a functional safe system consists in the development of a *functional safety concept*. Based on the functional safety concept the *technical safety concept* is derived specifying the technical implementation of the functional safety requirements. In the following sections these concepts for the convertible top are presented.

FUNCTIONAL SAFETY CONCEPT - As provided in the ISO WD 26262, we start with the hazard analysis and risk assessment. Hazard analysis requires a comprehensive understanding of the considered system, in order to detect possible malfunctions. For the convertible top the following hazards have been identified:

1. While the vehicle is driving, the partly opened convertible top breaks away resulting in an accident.
2. The movement of the convertible top causes an injury (bodily parts are clamped in mechanics).

The next step concerns risk assessment of the identified hazards. This is achieved by classifying the hazards by means of three parameters, estimating the potential severity, the probability of exposure in the driving and operating situations and the controllability. It is expected, that hazards in many automotive systems are rated ASIL C¹ which involves demanding safety requirements. Therefore we assume the class ASIL C for both identified hazards without further risk assessment.

For both hazards, safety goals are formulated defining top-level safety requirements:

1. The convertible top will not move if the vehicle is driving.
2. The convertible top will not move if this is not requested by the user.

¹ The correctness of this assumption depends on the final version of ISO 26262. Furthermore, the actual ASIL classification has to be considered carefully for each individual system.

Deriving functional safety requirements from these safety goals can be achieved via a fault tree analysis (FTA) for each safety goal using the undesired state as top event. Figure 1 shows the FTAs for the two safety goals. They take into account faults outside the electronic control unit (ECU) as well as faults in the Hardware (HW) and Software (SW) of the ECU. Considering the basic events of the fault trees, the following functional safety requirements are extracted:

- The fault *speed value erroneously 0 km/h* must be detected and handled.
- A fault in the delivered ignition status (*ignition on although ignition off*) must be detected and handled.
- The faults *control panel accidentally active* and the misuse (driver requests roof movement while driving) must be detected and handled.
- A fault in the I/O path, in the processing unit and in the memories must be detected and handled.
- Software faults causing a violation of a safety goal must be detected and handled.

The handling of these safety-critical faults is closely connected with the safe states of the system. For the convertible top, maintaining or switching over to the safe state *deactivated roof actuation* is sufficient. In addition to the reaction to a fault, a so-called *fault tolerant time span* has to be defined for each safety-goal. It determines the time that may pass before a fault causes a hazardous state. Because of the inertia of the mechanics, the fault tolerant time span for safety goal 1 is set to 1s. For safety goal 2 a faster reaction of 100ms is required.

Based on the functional safety requirements, the safety architecture concept is designed as shown in Figure 2. Primarily, it comprises the two functional units roof control function and safety function. The latter forms a redundant unit, which is responsible for fulfilling the safety requirements. Therefore, redundant input paths are used for all safety-related input parameters. An additional out-put path is designed to unlock the roof actuation if all safety requirements are fulfilled. Furthermore, both units are provided with the necessary hardware components. As the independence of the functional units is required in order to guarantee the compliance with the safety goals, further measures have to be taken if both units rely on common resources.

TECHNICAL SAFETY CONCEPT - When implementing the functional safety concept on a dual-core microcontroller, the partitioning into the roof control function and the safety function as well as their required independence advises to map the control function to one processor and the safety function to the other. Still there are two different approaches left.

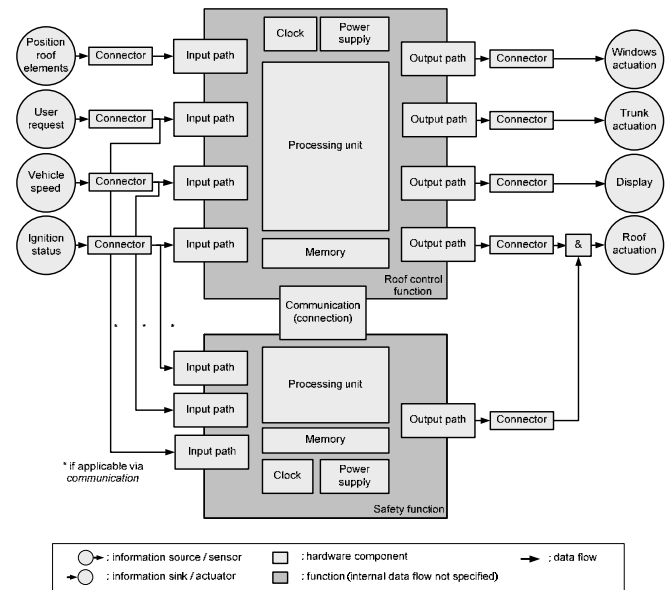


Figure 2: Safety architecture concept

Application-specific Approach - This approach uses the knowledge of the application to define an appropriate safety function. As in the application example only the roof actuation is safety-related, the safety function will have to decide redundantly if the activation is allowed. Evaluating the FTAs and the functional safety requirements, the safety goal can be fulfilled by implementing the following relation:

An activation of the hydraulic pump will be allowed if the current speed is 0 km/h, the ignition has been turned on and the control panel is activated.

Thus, the only task performed by the safety function is the continuous checking of three simple conditions and the appropriate actuation of the provided output which locks or unlocks the roof actuation. While the compliance with the fault tolerant time spans is without difficulty (in practice one check took about 3ms), the existence of common cause failures due to the common use of hardware resources has to be analyzed carefully.

The roof control function and the safety function both use their own processor. Hence, processor faults will not lead to a single point of failure (except for design faults). Further on, we assume that the dual-core microcontroller possesses two physically independent volatile memories. As the dual-core microcontroller chosen for implementation stores both programs in a common flash memory, the program of the safety function is loaded into the private volatile memory of its core after the flash memory was verified by a checksum test during initialization. Further commonly used resources are the microcontroller bus and the clock. The easiest way to avoid such common cause failures is an external time-windowed watchdog, which has to be triggered once in the provided time-window. If the triggering fails, a reset