CAN Controller Sharing Mechanism for Mixed Critical Systems

Masataka Ogawa[1,a)] Yusuke Kato[1] Shinya Honda[1]

Abstract: Recently, high reliability is required for the development of embedded systems. Dealing with reliability requirements causes an increase in the complexly, development costs and verification costs of embedded systems. In order to reduce processings that must be ensured high reliability, an embedded software is divided into trusted/nontrusted partitions according to reliability required for each processing. A non-trusted partition accesses to a device may interfere with a trusted partition access because of sharing the device between trusted and non-trusted partitions. In the conventional device sharing method, only a trusted partition can access devices. Furthermore, if the non-trusted partition needs to access devices, it must request a trusted partition to deal with processings for device access. However, this method has a disadvantage that requests for device access may generate a high overhead. In this paper, we propose a device sharing mechanism Protection Wrapper that it allows to access devices without interference by the non-trusted partition and to share them with a low overhead. Applying our proposing mechanism to a CAN controller confirmed that the mechanism enabled to reduce a overhead for device sharing in comparison with the conventional method.

Keywords: embedded systems, mixed critical systems, automotive control systems, CAN, functional safety

1. Introduction

Recently, high reliability is required for the development of embedded systems. Dealing with these reliability requirements causes in an increase in the complexly, development costs, and verification costs of embedded systems. The reason is all functions don't need high reliability but functions only required low reliability must also be designed on the basis of high reliability requirements. This system which includes components required different reliability is called a mixed critical system. In order to reduce functions that must be ensured high reliability, Crespo et al. [1] suggested an approach that an embedded software is divided into trusted/non-trusted partitions according to reliability for each processing. This approach aims to increase independence between functions by applying a spatial and temporal protection and to enable to design each function with the required reliability.

There are automotive control systems as embedded systems that require high reliability. Recently, a number of ECUs (Electronic Control Units) per vehicle are quickly increasing because automotive control systems need higher performance and functionality. This causes problems with increasing hardware costs and insufficient space for putting ECUs in a vehicle. In order to reduce a number of ECUs in a vehicle, there is an approach that combines functions in different ECUs into one software and then executes the software on one ECU. However, the system may be mixed critical systems since high reliability functions (e.g. a brake control) and low reliability functions (e.g. a body control) are executed on one ECU.

Fig.1 shows models of mixed critical automotive control systems. There are two partitioning approaches. One is that different reliability functions are executed on one processor by using a kernel having spatial and temporal protection mechanisms (Fig.1a). The other is that functions required the same reliability level is located in one processor and the functions are executed on each processor (Fig.1b). An ECU includes devices accessed by processors and devices must be shared between different partitions. Recently, adapting multicore architecture to automotive control systems is progressing. For example, Herber et al. [2][3][4] proposed that a virtualized CAN controller was shared between different processors. Even if a malfunction occurs in one processor, functions in the other processors are operated correctly by adapting the spatial and temporal protection mechanisms to the virtualized CAN controller.

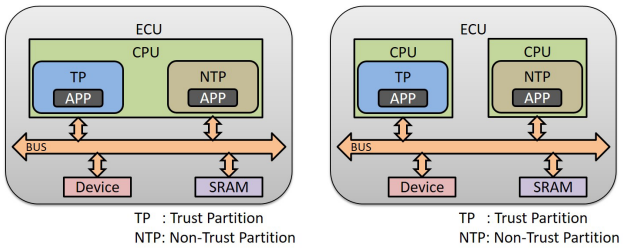
In this paper, we propose a device sharing mechanism Protection Wrapper that it allows to access devices without interference by a non-trusted partition and to share them with a low overhead. This mechanism allows to restrict device access from a non-trusted partition and to ensure behavior of a trusted partition.

2. Factors Threatening Reliability

Mixed critical systems include devices with small monetary/hardware area costs (e.g. timer modules, counter modules) and devices that should be shared between different partitions because of high costs (e.g. network interfaces). In this paper, we consider a sharing mechanism for a CAN controller which is one of the latter.

Generally, most of devices (including a CAN controller) are designed on the assumption that they are not shared between dif-

[1] Graduate School of Informatics, Nagoya University, Nagoya, 464-8603 Japan
[a)] masa-bach@ertl.jp



(a) Single processor (b) Multi processors

Fig. 1 Mixed critical systems

ferent partitions. Therefore, sharing a CAN controller may cause following factors threatening reliability.

Factor 1: A non-trusted partition overwrites incorrect settings to the CAN controller.

Factor 2: A non-trusted partition halts the CAN controller while a trusted partition initializes it.

Factor 3: A non-trusted partition sends messages frequently.

Factor 4: A non-trusted partition sends messages with a high priority message ID.

Factor 5: A non-trusted partition uses sending message boxes gratuitously.

Factor 6: A non-trusted partition reads messages to a trusted partition gratuitously.

A trust partition must be protected spatially and temporally. In regard to the spatial protection, systems should be designed so that a trusted partition doesn't cause a malfunction due to an unauthorized access from a non-trusted partition. In regard to the temporal protection, systems should be also designed so that receiving/sending messages from/to a trusted partition are not interfered by non-trusted messages. In addition, a non-trusted partition causes a following danger of network.

Factor 7: A non-trusted partition uses Message IDs allocated to a trusted partition.

In case that the CAN controller is shared, not only a trusted partition but the network must be protected. In addition, the network must not be occupied by the non-trusted partition due to unauthorized traffic or DOS attack.

In Section 3, we introduce three device sharing approaches for removing those factors.

3. Device Sharing Approaches

This section explains three approaches that enable to share devices while ensuring reliability.

3.1 PRC Approach

Fig.2 shows the PRC(Remote Procedure Call) approach. In this approach, only a trusted partition accesses target devices. In contrast, a non-trusted partition never access target devices and it must request a trusted partition to deal with processings for the device access if it needs to use the devices. However, this approach has following two disadvantages. One is that requests from a non-trusted partition may increase a load to a trusted partition. Hence, this approach is not appropriate for real time systems. The other is that independence between partitions is impaired because a trusted partition has a relationship with a non-trusted partition. Consequently, a trusted partition may be under DOS attacks such as the case that a non-trusted partition generates interrupts to a trusted partition repeatedly.

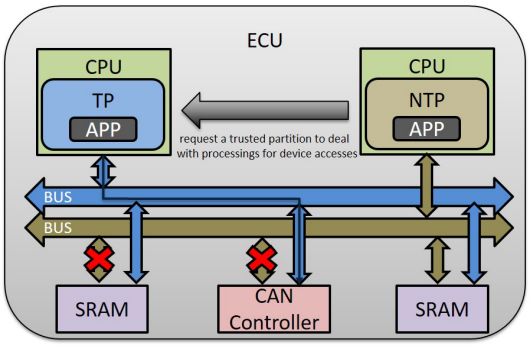


Fig. 2 RPC approach

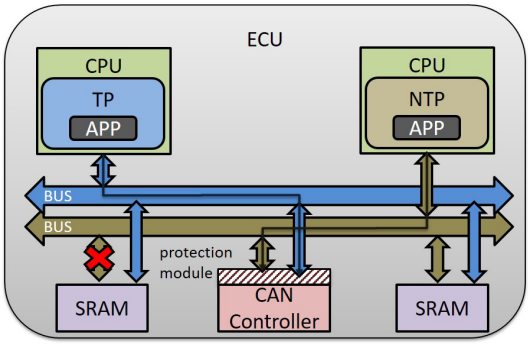


Fig. 3 Direct access approach

3.2 Full Virtualization Approach

In this approach, a shared device is modified so that several processors can access the device by adapting virtualization. This approach is difficult to develop and introduce because existing devices need to be modified drastically and processors need to support virtualization technology. In addition, this approach has low versatility since different places are modified for each device.

3.3 Direct Access Approach

Fig.3 shows the direct access approach. In this approach, a non-trusted partition is restricted access to a device. This approach allows to minimize the modification of devices and to share devices between partitions by adapting a wrapper that protects existing devices. Details of the wrapper are explained in Section 4.

4. Protection Wrapper

In this section, we explain our proposing device sharing mechanism Protection Wrapper. By applying the wrapper to a device, the direct access approach is realized.

4.1 Required Functions

This section shows functions required for dealing with above factors threatening reliability. Protection Wrapper includes following functions:

(a) **Access class monitoring function** identifies access classes (e.g., read request and write request).

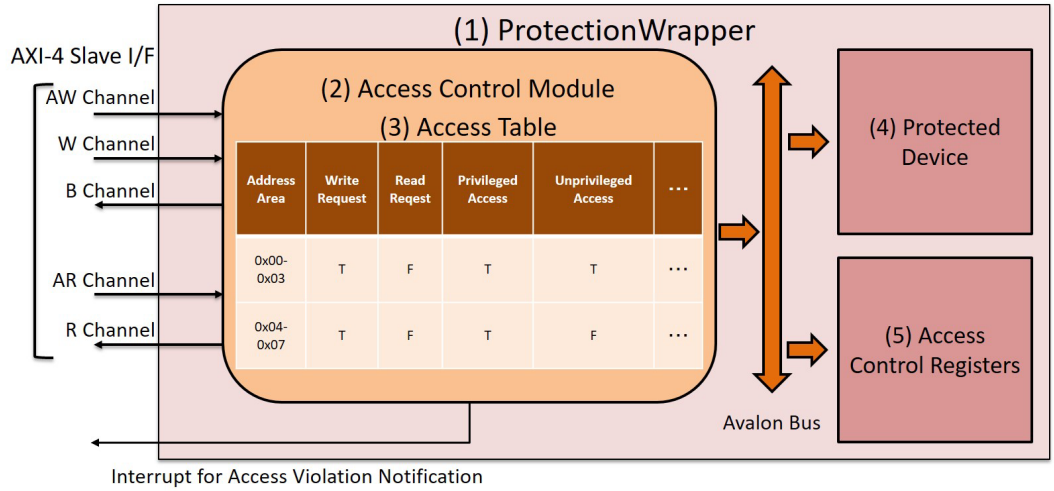


Fig. 4 Protection Wrapper

(b) **Access source partition monitoring function** identifies which partitions access to a device.
(c) **Writable data monitoring function** confirms whether the data written to the register are within the allowable values.
(d) **Access period monitoring function** confirms whether the device is accessed within the preset period.
(e) **Access violation monitoring function** notifies a trusted partition of the access violation if a non-trusted partition accesses devices illegally.

Those functions are able to eliminate the factors in Section 2 as follows:

Factor 1: Restrict that a non-trusted partition writes an illegal data to a device register with the function (b) and (c).
Factor 2: Restrict registers that are allowed to be written by a non-trusted partition with the function (a) and (b).
Factor 3: Restrict period of message sending by a non-trusted partition with the function (b) and (d).
Factor 4: Restrict message IDs used by a non-trusted partition with the function (b) and (c).
Factor 5: Restrict message boxes that are allowed to be written by a non-trusted partition with the function (a) and (b).
Factor 6: Restrict message boxes that are allowed to be read by a non-trusted partition with the function (a) and (b).
Factor 7: Restrict message IDs used by a non-trusted partition with the function (b) and (c).

4.2 Module Architecture

Fig.4 shows module architecture of Protection Wrapper. A target device must be accessed via the access control module and if the conditions of access tables are not satisfied, the access is denied. Details of access tables are explained in Section 4.3. If access to the device is denied, a trusted partition is notified of an access violation by the interrupt from the access control module with the function (e). Therefore, the trusted partition received the access violation enables to deal with the access safely even if a failure occurs in a non-trusted partition and an access violation is detected.

4.3 Access Table

Protection Wrapper controls an access to devices according to information of access tables. One access table consists of an address area and access attributes. If one or several the conditions of access tables are satisfied, the access is permitted.

4.3.1 Address Areas

The range of a register to provide protection is determined by address areas. Address areas can also designate an address range including several registers. In addition, the number of access tables is allowed to be changed.

4.3.2 Access Attributes

Protection Wrapper determines conditions for access control by referring access attributes. Protection Wrapper has following access attributes:

- **read request**: It defines whether an access request from a read channel is permitted or prohibited.
- **write request**: It defines whether an access request from a write channel is permitted or prohibited.
- **privileged / unprivileged access mode**: It defines whether a privileged/unprivileged access request is permitted or prohibited.
- **secure / non-secure access mode**: It defines whether a secure / non-secure access request is permitted or prohibited.
- **data / instruction access mode**: It defines whether a data / instruction access request is permitted or prohibited.
- **source partition**: It defines whether an access request from each partition is permitted or prohibited.
- **access period**: If a device is accessed at a frequency exceeding the access period, the access request is prohibited.
- **writable data**: If a data written to the device register differs values specified the writable data, the access request is prohibited.

4.4 Access Control Registers

Addess areas and access attributes are stored in access control registers. Not only protected devices but access control registers itself is also allowed to be protected by the access control module. In addition, values of registers can be changed dynamically.

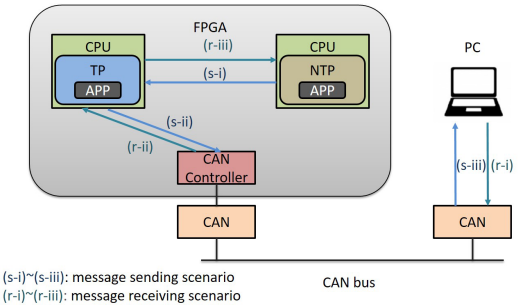


Fig. 5 Sending / receiving scenarios in RPC approach

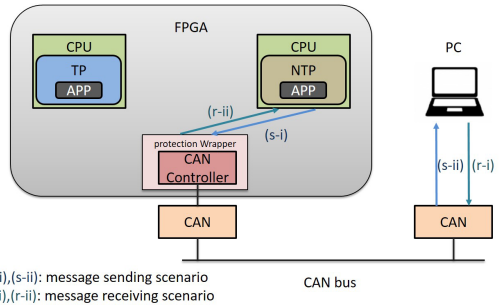


Fig. 6 Sending / receiving scenarios in the direct access approach

Table 1 Latency generated by sending / receiving messages

	Sending messages	Receiving messages
RPC approach	1,557 cycle	1,760 cycle
Direct Access approach	194 cycle	455 cycle

Table 2 The number of LUTs for each communication module configuration

communication module	the number of LUTs
CAN controller * 1	45,886
CAN controller * 2	55,228
CAN controller + Protection Wrapper with 5 access tables	50,516
CAN controller + Protection Wrapper with 10 access tables	54,239
CAN controller + Protection Wrapper with 20 access tables	61,665

5. Evaluation

In this section, we explain evaluations about RPC approach and a direct access approach. Since a full virtualization approach is difficult to realize, we did not evaluate it in this paper.

5.1 Outline of Experiments

We implemented the evaluation system that shares the CAN controller between two processors and communicates with external ECUs or PCs. The system is constructed on FPGA (Field Programmable Gate Array) and includes two processors with clock frequency of 50 MHz. Applications in a trusted partition are executed on one processor and applications in a non-trusted partition is executed in the other. We evaluated latency and hardware on the system in the RPC approach / the direct access approach.

5.2 Latency Evaluation

We evaluated latency due to CAN communications between the evaluation system and a host PC. Fig.5 shows sending / receiving scenarios in the RPC approach. In the sending scenario, firstly, a non-trusted partition requests message sending to a trusted partition. Secondly, the trusted partition accesses a CAN controller and sends messages instead. Finally, the host PC receives messages via the CAN bus. The receiving scenario proceeds in the reverse order of the sending scenario.

Fig.6 shows sending / receiving scenarios in a direct access approach. In the sending scenario, a non-trusted partition directly accesses a CAN controller through Protection Wrapper and send messages to the host PC via the CAN bus. The receiving scenario proceeds in the reverse order of the sending scenario as well as the RPC approach.

Table 1 shows latency generated by sending / receiving scenarios. In the sending scenario, applying the direct access approach was able to reduce the latency by about 85% compared to the RPC approach.

The reason is that an overhead generated when a non-trusted partition requests device access to a trusted partition is not necessary in the direct access approach. In the receiving scenario, we confirmed the same latency reduction as the sending scenario.

5.3 Hardware Area Evaluation

Table 2 shows hardware areas of the evaluation system. If one CAN controller is not shared between partitions, it is necessary to prepare two CAN controllers and to allocate the different CAN controller to the each partition. . In the direct access approach, a communication module consists of one CAN controller and Protection Wrapper with several access tables. The result of the evaluation indicates that the direct access approach can reduce hardware areas compared to the case of preparing two CAN controllers by setting the number of access tables to 10 or less.

6. Conclusion

In this paper, we proposed Protection Wrapper, which was a device sharing mechanism for mixed critical systems. Protection Wrapper allows to protect a trusted partition and the network and to share devices between partitions with a low overhead. In evaluations, we measured the latency and the hardware area when this mechanism was applied to the systems included a CAN controller. As a result of evaluations, the direct access approach which is applying Protection Wrapper to a CAN controller allowed to reduce the access latency by about 85% compared to the RPC approach. In addition, we confirmed that applying Protection Wrapper needed smaller hardware areas than preparing a CAN controller for each partition.

References

[1] Crespo, Alfons, et al. "Mixed criticality in control systems." IFAC Proceedings Volumes 47.3 (2014): 12261-12271.
[2] Herber, Christian, et al. "Spatial and temporal isolation of virtual can controllers." ACM SIGBED Review 11.2 (2014): 19-26.
[3] Herber, Christian, et al. "Self-virtualized CAN Controller for Multicore Processors in Real-Time Applications." ARCS. 2013.
[4] Herber, Christian, et al. "HW/SW trade-offs in I/O virtualization for Controller Area Network." Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE. IEEE, 2015.