SoC Control Technology to Reduce Separate Microcontrollers for DMS/HDL

Authors: *Akane Kimura*, Akira Mitamura*, Masahiko Kamiyama*, Hiromu Uemura**

**Mitsubishi Electric Mobility Corporation*

Abstract

Mitsubishi Electric Mobility Corporation contributes to creating a safe and secure mobility society by providing ADAS (Advanced Driver-Assistance Systems) products such as ADAS Electronic Control Units (ADAS-ECUs), Driver Monitoring Systems (DMS), and High-Definition Locators (HDL). DMS and HDL in particular require both computationally intensive tasks and real-time tasks, and in conventional Mitsubishi Electric Mobility Corporation systems, functions were divided between a system on chip (SoC) and a microcontroller to meet system requirements. However, recent advances in SoC capabilities have enabled a single SoC to execute both real-time and non-real-time tasks by selectively utilizing multiple cores. Accordingly, we integrated the microcontroller functions of the conventional system into the SoC and evaluated functionality, performance and cost.

1. Introduction

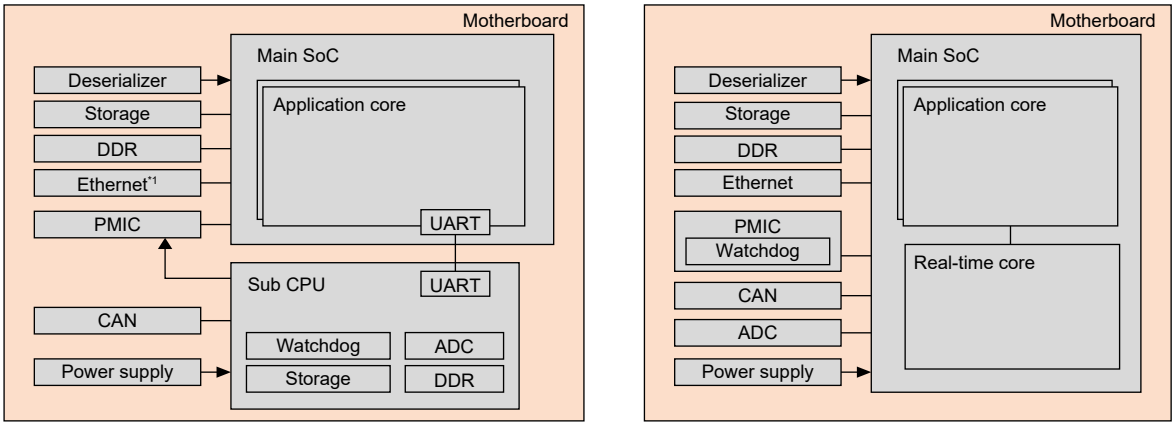
Mitsubishi Electric Mobility Corporation is working toward achieving an accident-free society in which people can move with peace of mind and safety, and is developing ADAS to support safe driving for drivers, such as ADAS-ECUs that apply information processing technologies for inside and outside the vehicle, DMS that applies image recognition technologies, HDL that applies high-precision positioning technologies, and Vehicle-to-Everything (V2X) based on external communication technologies.

While strict real-time performance is required for control-related ECUs such as electric power steering, ADAS-related ECUs such as DMS and HDL require both computationally intensive tasks such as image processing and real-time tasks.

In Mitsubishi Electric Mobility Corporation's conventional ADAS-related ECUs, a dual-processor architecture consisting of an SoC and a separate microcontroller was used to satisfy various performance requirements of in-vehicle devices. The SoC handled applications that define the system, such as image recognition and high-precision positioning, while the microcontroller handled functions that require real-time performance, such as in-vehicle network communication control. However, this dual-processor architecture increased the number of on-board components and board area, making cost reduction challenging and also complicating hardware design.

To address these issues, we developed a system (hereinafter referred to as the "developed system") using a heterogeneous-core SoC with a real-time core, thereby enabling both real-time and computationally intensive processing on a single SoC. As shown in Fig. 1, we integrated the microcontroller functions of the conventional system into the SoC to evaluate functionality, performance, and cost. This enabled unified storage, resource sharing, inter-core communication, and simplified system control, achieving a 30% reduction in component cost while maintaining the existing functionality and improving performance.

In this paper, we outline the features of the developed system in Chapter 2, present the evaluation results in Chapter 3, and conclude with Chapter 4.



(a) Conventional system (b) Developed system

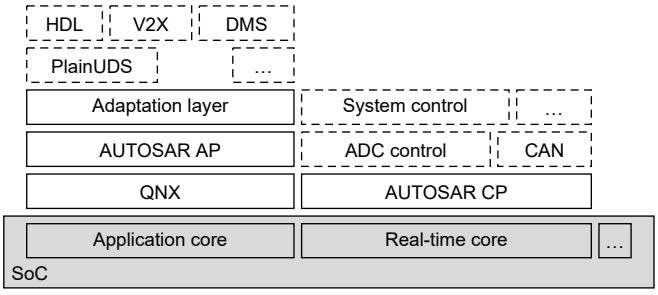
DDR: Double Data Rate, PMIC: Power Management IC, CAN: Controller Area Network, ADC: Analog-to-Digital Converter, UART: Universal Asynchronous Receiver Transmitter

*1 Ethernet is a registered trademark of FUJIFILM Business Innovation Corp.

Fig. 1 System block diagrams: conventional vs. developed

2. Features of the Developed System

As shown in Fig. 2, the developed system runs a real-time OS (QNX*2) and AUTOSAR*3 Adaptive Platform (AP) for computationally intensive tasks on the SoC application cores, while AUTOSAR Classic Platform (CP) for power control and in-vehicle communication runs on the real-time core, thereby achieving microcontroller integration.



UDS: Unified Diagnostic Services

Fig. 2 Software block diagram of the developed system

2.1 Configuration of the developed system

Details of the developed system configuration described above are provided below.

2.1.1 AUTOSAR CP

AUTOSAR CP is a platform for embedded systems defined by AUTOSAR, the automotive standardization organization, and it emphasizes real-time performance and stability. When deployed on the SoC's real-time core, it abstracts hardware details and improves software reusability and interchangeability.

2.1.2 AUTOSAR AP

AUTOSAR AP is a platform defined by AUTOSAR for advanced and computationally intensive applications. AUTOSAR AP is deployed on the SoC's application cores to enhance software reusability and interchangeability, as well as development efficiency and quality. As a defining feature of AUTOSAR AP, service-oriented communication between Adaptive Applications enables applications to be developed independently of specific communication counterparts, while facilitating switching between configurations with and without AUTOSAR AP.

*2 QNX is a registered trademark of BlackBerry Limited.

*3 AUTOSAR is a registered trademark of AUTOSAR GbR.

2.1.3 QNX

QNX is a real-time operating system (RTOS) developed by QNX Software Systems, a Canadian company. It employs a microkernel architecture and thus achieves high stability and real-time performance. QNX, which ensures high reliability and responsiveness while readily incorporating new technologies, is deployed on the application core. Because QNX can be deployed across various SoCs and hardware platforms without being tied to specific hardware, it is also well-suited for the development of various types of ECUs.

2.2 Single storage system

Automotive ECUs are equipped with storage, either within or outside the microcontroller, for storing and updating data such as control programs, configuration information, and logs. In the conventional system, dedicated storage was provided to ensure that both the SoC and the microcontroller could independently access their respective storage. The system was first booted from the bootloader stored in the microcontroller, and then the bootloader stored in the SoC was executed to complete the boot process.

In the developed system, all data, including the bootloader, is stored in a single storage device, enabling the system to boot from a single bootloader. This reduced the number of components and the board area required for the microcontroller and its peripheral circuits compared to the conventional system (Table 1). However, to implement the power-supply voltage and temperature monitoring functions previously performed by the microcontroller in the conventional system, an external ADC IC and a thermistor were added.

Table 1 Component configurations: conventional vs. developed

	Conventional System	Developed System
Storage	Two components	One component
Memory	Two components	One component
Separate microcontroller	✓	
X’tal, LDO, peripheral components	✓	
ADC IC		✓
Thermistor		✓

X’tal: Crystal Oscillator; LDO: Low Drop Out

2.3 Inter-function resource sharing (or single-memory system)

In the developed system, in addition to the storage described in Section 3.1, memory, timers, and sensors were implemented with higher performance by sharing the SoC’s high-performance hardware resources among the cores, while maintaining the same functionality as the conventional system.

As shown in Table 1, in the conventional system, each microcontroller had dedicated memory, enabling its respective functions to be achieved without interference from other microcontrollers. However, when multiple cores access the same memory, as in the developed system, memory access violations can become a critical issue. Therefore, memory isolation is achieved for the shared memory by utilizing both the SoC’s and the OS’s memory access protection capabilities. This enables a reduction in the number of components while ensuring safe resource sharing.

2.4 Inter-core communication function

In the conventional system, the SoC and the microcontroller were connected via physical lines such as UART for control, whereas in the developed system, inter-core communication using shared memory and an interrupt register was implemented. In addition, UART communication in the conventional system used a half-duplex protocol, which limited communication efficiency; in the developed system, this protocol was changed to a full-duplex protocol, thereby enabling high-speed communication. Furthermore, in the conventional system, strict synchronization of operating times was difficult, and in the event of a failure, analysis required cross-referencing the operation logs of each microcontroller. In the developed system, time synchronization is facilitated by having each core refer to the same clock. Furthermore, since information

on the real-time core can be obtained from the application core via shared memory, it is also straightforward to consolidate the log output mechanism on the application core.

2.5 System control function

In the conventional system, the microcontroller that first received vehicle information via communications such as CAN was responsible for startup and shutdown control and power management. For system control, multiple GPIOs (General Purpose Input/Output) were used to control the microcontrollers and the multi-function power-supply ICs before the inter-microcontroller communication described in Section 2.4 was established and after it had ended.

In the developed system, the real-time core performs system control. The number of GPIOs managed is reduced, and the startup/shutdown sequence is simplified by following the SoC boot sequence. In addition, reducing error sources and system state transitions is expected to improve system reliability and reduce development and verification costs.

Regarding software updates, complex state management and longer update times were also issues in the conventional system because software on multiple CPUs was updated sequentially. Furthermore, because the microcontroller had limited storage capacity, it was often difficult to support dual-bank redundancy. In the developed system, software updates can be simplified through a dual-bank configuration.

In the conventional system, system startup/shutdown and control such as system health monitoring were achieved not only via inter-microcontroller communication over UART but also with ON/OFF control (GPIO control) using hard wiring. In the developed system, these processes were replaced with shared memory and CPU register access, thereby reducing GPIO-based control and simplifying system control. This helps prevent increased system complexity caused by a shortage of GPIO pins, which was a critical issue in the conventional system.

The developed system also simplifies the software update function. In the conventional system, the following issues existed:

(1) Sequential processing was required: data in the SoC's storage must be updated first, followed by data in the microcontroller's storage.
(2) If an update failed on either the SoC or the microcontroller, the other node had to be rolled back to maintain system-wide consistency.
(3) Because the microcontroller had limited storage capacity, dual-image redundancy was not feasible, which necessitates complex update control.

In the developed system, consolidating storage into a single large-capacity SoC storage device addressed the issues of the conventional system, thereby simplifying the software update function and reducing the update time by 45% (equivalent to a 180% increase in update throughput).

3. Evaluation

This section evaluates the changes in the system resulting from the removal of the microcontroller described in Chapter 2.

3.1 Startup performance

With the removal of the microcontroller, boot time becomes a key challenge in realizing a large-scale system that supports multiple operating systems across multiple cores within a single SoC. As the system scale increases, the software image size also increases, lengthening the time required to load the image from non-volatile memory and start execution. Fast booting is essential for automotive ECUs; for instance, ADAS-related ECUs are required to respond on the CAN bus within approximately 200 ms after power-on.

In the conventional system, the microcontroller loaded only the minimum necessary data from internal flash memory in order to achieve rapid startup response. In the developed system, programs for the real-time core and the application core are managed separately, and rapid startup response is achieved by first loading only the programs that require fast response at startup. As shown in Fig. 3, we achieved startup response performance equivalent to that of the conventional system by first starting the minimum required programs for communication response and then starting the remaining programs in parallel. Furthermore, by separating the timing and regions of storage access, it became possible to save error logs through single-core processing when a system abnormality occurs, as in the conventional system.

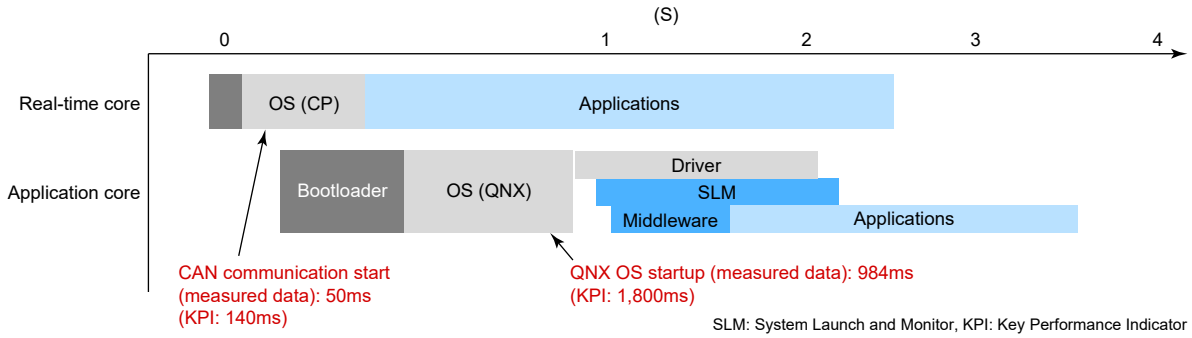


Fig. 3 Startup timing chart of the developed system

3.2 Inter-core communication performance

In the conventional system, because communication was performed over physical lines, it was relatively susceptible to noise and interference, and it was difficult to increase the amount of data transferred per communication. In contrast, in the developed system, inter-core communication using shared memory instead of physical lines became possible, and communication speeds several tens of times higher were achieved as the transfer size increased, as shown in Fig. 4.

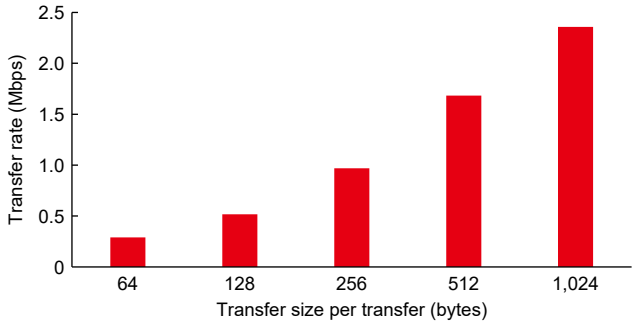


Fig. 4 Inter-core communication throughput of the developed system

4. Conclusion

Conventional systems with a dual-processor architecture faced challenges in reducing costs due to increases in both the number of on-board components and board area, which also complicated hardware design.

Therefore, we developed a single-SoC system using an SoC with a heterogeneous-core architecture that includes a real-time core capable of real-time processing. In this system, a real-time OS (QNX) and AUTOSAR AP for computationally intensive tasks run on the SoC application cores, while AUTOSAR CP for power control and in-vehicle communication runs on the real-time core, thereby integrating the microcontroller functions of the conventional system. We then evaluated its functionality, performance, and cost. The developed system enabled unified storage, resource sharing, inter-core communication and simplified system control, achieving a 30% reduction in component cost while maintaining the existing functionality and improving performance.