AUTOSAR-based Communication Coprocessor for Automotive ECUs

Ahmed Hamed
Mentor Graphics Corporation
Cairo, Egypt
Email: Ahmed_Hamed@mentor.com

Mona Safar, and M. Watheq El-Kharashi
Ain Shams University,
Department of Computer and Systems Engineering
Cairo, Egypt
Email: mona.safar@eng.asu.edu.eg,
and watheq.elkharashi@eng.asu.edu.eg

Ashraf Salem
Mentor Graphics Corporation
Cairo, Egypt
Email: ashraf_salem@mentor.com

***Abstract*—In this paper, we present a novel approach to enhance the performance of the AUTOSAR-based Electronic Control Units. The operations done by the AUTOSAR communication module are the most Electronic Control Unit time-consuming operations so our approach modifies the design model of the AUTOSAR Layered Software Architecture by adding the communication coprocessor component. This model-based hardware/software codesign expedites the AUTOSAR communication operations while keeping the interfaces with the upper and lower layers unchanged. The coprocessor covers two communication-based operations. It consists of six building blocks. It communicates with the original Electronic Control Unit through the External Peripheral Interface module, which is a high speed parallel bus for external peripherals. The implemented coprocessor achieves up to 140x speedup over the software communication module solution. This gives a room to extend the automotive applications and increase the amount of the exchanged information by these applications without affecting the performance.**

I. Introduction

In this paper, we explore the usage of coprocessors in the AUTOSAR Layered Software Architecture. We modified the design model of the AUTOSAR Layered Software Architecture by adding the AUTOSAR COM Coprocessor component, as shown in Fig. 1. The building blocks of our coprocessor and the interactions between these building blocks are shown in Fig. 2.

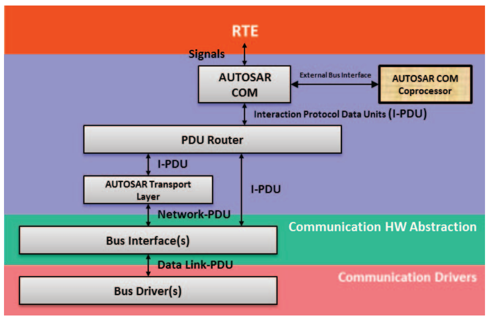

Fig. 1. Modified Layered Software Architecture of AUTOSAR.

The implemented coprocessor achieves up to 140x speedup over the software (SW) COM solution. This gives a room for the Original Equipment Manufacturers (OEMs) and Tier1 suppliers to extend their automotive applications and increase the amount of the exchanged information by these applications without affecting the performance.

II. Design of the AUTOSAR COM Coprocessor

The AUTOSAR COM Coprocessor, integrated within any ECU, as shown in Fig. 2, consists of six building blocks, which are described in TABLE I.

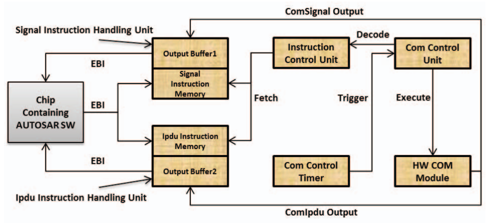

Fig. 2. Building blocks of the AUTOSAR COM Coprocessor.

The Signal Instruction Memory has two variants: Single shot mode and batch mode. The size of Signal Instruction Memory is 1 word in the single shot mode. In the batch mode, the size of Signal Instruction Memory is 64 words.

The HW COM Module building block, as shown in Fig. 3, consists of two main memory components.

- The rom_n component, which is a Read Only Memory (ROM) that contains the properties of the ComSignals that are processed by the coprocessor. The rom_n component is a 32-bit word ROM, where each ComSignal requires four ROM words for saving its related information, as shown in TABLE II.
- The Ipdu component, which is a Random Access Memory (RAM) that contains the run-time values of the processed ComIpdus. The Ipdu component is a 64-bit word RAM, where each ComIpdu requires one RAM word for saving its current run-time value.

The designed coprocessor is a multi-cycle coprocessor. The number of the execution cycles needed for the SendSignal and ReceiveSignal instructions are 12 cycles. It consists of four phases, as shown in TABLE III.

TABLE I. BUILDING BLOCKS OF THE COPROCESSOR.

Blocks and their Functionality
Chip Containing AUTOSAR SW
the original ECU that contains the AUTOSAR SW, which contains the SW part of the COM module. The COM requests from the application or the lower layer (e.g., Com_SendSignal, Com_RxIndication, etc.) are received by a wrapper code inside the SW part of the COM module. The interfaces of this wrapper code match the prototype of the SW COM related APIs. The wrapper code loads Signal Instruction Memory and Ipdu Instruction Memory with the instructions corresponding to these requests. Then, it gets the return values and values of the output parameters from Output Buffer1 and Output Buffer2.
Signal Instruction Handling Unit
contains two components, which are: Signal Instruction Memory and Output Buffer1. Signal Instruction Memory contains the loaded instructions that are associated with the ComSignal related APIs (e.g., Com_ReceiveSignal, Com_SendSignal, etc.). Output Buffer1 contains the return values and values of the output parameters from these APIs (e.g., SignalDataPtr parameter of the Com_ReceiveSignal API).
Ipdu Instruction Handling Unit
has the same functionality of Signal Instruction Handling Unit but for the ComIPdu related APIs (e.g., Com_RxIndication, Com_TxConfirmation, etc.).
Instruction Control Unit
fetches the instructions from Signal Instruction Memory and Ipdu Instruction Memory.
COM Control Unit
decodes the instructions fetched by the Instruction Control Unit and also is responsible for processing the events from the COM Control Timer. It uses the resources of the HW COM Module to execute the COM instructions.
COM Control Timer
triggers the COM Control Unit for executing the main/periodic functions of the COM module (Com_MainFunctionTx, Com_MainFunctionRx, and Com_MainFunctionRouteSignals).
HW COM Module
contains the needed HW resources (e.g., buffers, registers, memories, arithmetic units, etc.) that are used for executing the COM instructions.

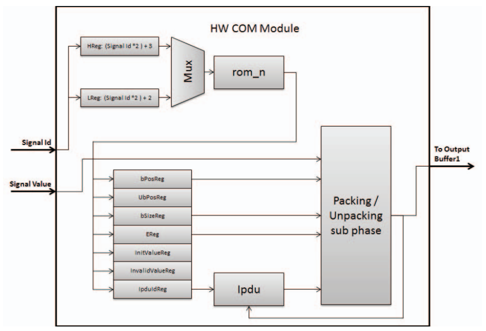

Fig. 3. The components of the HW COM Module building block.

TABLE II. ENCODING OF THE INFORMATION OF A GIVEN COMSIGNAL INSIDE THE ROM_N COMPONENT.

	B_{31-26}	B_{25-20}	B_{19-14}	B_{13-12}	B_{11-0}
Word 0	Init Value				
Word 1	Invalid Value				
Word 2	Ipdu Id				
Word 3	bPos	UbPos	bSize	E	Not used

TABLE III. PHASES OF THE COPROCESSOR.

Phase	**Functionality**
Fetching	The Instruction Control Unit fetches the instructions from Signal Instruction Memory.
Decoding	The COM Control Unit decodes the fetched instructions.
Execution	The COM Control Unit distributes the control signals to control the HW resources in the HW COM module building block. This phase consists of two sub phases that will be done to complete the execution of the decoded instructions. 1) The Extraction sub phase is responsible for preparing the needed information for the packing/unpacking process. It is responsible for extracting the needed properties of the current processed ComIpdu (ComIPduType, IPduSignalProcessing, etc.). It is also responsible for extracting the needed properties of the current processed ComSignal (ComSignalEndianness, ComSignalType, etc.). It is also responsible for extracting the run-time value of the current processed ComIpdu. 2) The Packing/Unpacking sub phase is responsible for packing the ComSignal into its corresponding ComIpdu based on the information extracted in the previous sub phase. Then, it is responsible for writing back the new value of the ComIpdu and then preparing the ComIpdu to be transmitted in case of the SendSignal instruction. It is also responsible for unpacking the ComSignal from its corresponding ComIpdu and then preparing it to be received in case of the ReceiveSignal instruction.
Delivery	The Output Buffer1 is loaded with the received value of the ComSignal after unpacking it from its corresponding ComIPdu in case of the ReceiveSignal instruction. The Output Buffer1 is loaded with the value of the ComIPdu to be transmitted in case of the SendSignal instruction.

III. EXPERIMENTAL RESULTS

The designed coprocessor has been synthesized using four different variants. Variant 1: The coprocessor contains 2000 ComSignals and 200 ComIpdus and it works in the single shot mode. Variant 2: The same as variant 1 but the coprocessor works in the batch mode. Variant 3: The coprocessor works in the single shot mode but the number of the ComSignals and ComIpdus have been doubled (4000 ComSignals and 400 ComIpdus). Variant 4: The same as variant 3 but the coprocessor works in the batch mode.

The performance of the designed coprocessor has been compared to the SW COM solution that is used in the Central Electronic Module ECU. The performance comparison is shown in TABLE IV.

Instead of adding the raw value of the execution time of the Com_SendSignal/Com_ReceiveSignal API that is used in the SW COM solution, the variable t is added to the execution time of the internal function Com_PackSignal/Com_UnPackSignal. This variable is used to represent the time needed by the Com_SendSignal/Com_ReceiveSignal API to complete the internal main functionality of these APIs that is equivalent to the phases/sub phases of the coprocessor.

TABLE IV. PERFORMANCE COMPARISON BETWEEN THE SW COM SOLUTION AND THE DESIGNED COPROCESSOR.

Comparison Points	**SW COM solution (120 MHz)**	**Coprocessor Variant**		**Speedup Variant**	
		1&2	**3&4**	**1&2**	**3&4**
Packing sub phase	2.42 us	16.59 ns	16.81 ns	**145.87**	**143.96**
Unpacking sub phase	2.16 us	16.59 ns	16.81 ns	**130.20**	**128.49**
Com_SendSignal	2.42 us + t	66.36 ns	67.26 ns	**>36.47**	**>35.98**
Com_ReceiveSignal	2. 16 us + t	66.36 ns	67.26 ns	**>32.55**	**>32.11**

IV. CONCLUSIONS AND FUTURE WORK

In this paper, a novel approach is presented to enhance the performance of AUTOSAR-based ECUs. Our approach is based on modifying the design model of the AUTOSAR Layered Software Architecture by adding the AUTOSAR COM Coprocessor component. The implemented coprocessor achieves up to 140x speedup over the SW COM solution. This gives a room to extend the automotive applications and increase the amount of the exchanged information by these applications without affecting the performance.

However, this cannot be considered a complete and standalone solution that can replace the SW COM solution in today's automotive software applications unless the remaining APIs (e.g., Com_RxIndication, Com_SendSignalGroup, Com_ReceiveSignalGroup, Com_MainFunctionRx, etc.) have been covered.