Improvement of Automotive Sensors by Migrating AUTOSAR End-to-End Communication Protection Library into Hardware

Horia V. Caprita[1, 2, *], Dan Selisteanu[2]
[1]*Continental Automotive Systems,*
8 Salzburg St., 550018 Sibiu, Romania
[2]*Department of Automatic Control and Electronics, University of Craiova,*
13 A.I. Cuza St., 200585 Craiova, Romania
horia.caprita@continental.com

***Abstract*—This paper explores new methods to increase the level of safety of data transfer between sensors and electronic control units (ECUs) in automotive communication. A new model of basic sensors to be used in automotive electronics is proposed. This model contains hardware modules that implement the end-to-end communication protection (E2E) mechanism, as defined by the Automotive Open System Architecture (AUTOSAR) standard. By adding this feature inside the sensors, it is possible that, in addition to increasing the safety level, these sensors can be directly connected to the network ECUs via standard communication buses (e.g., Local Interconnect Network (LIN), Controller Area Network (CAN), Flexray, etc.). This paper describes the model, design, and mapping (in a Field Programmable Gate Array device (FPGA)) of the hardware E2E module capable of generating the Cyclic Redundancy Code (CRC) and counter signal for a customized message. This message represents the output of the new sensor E2E module used in a safety communication as requested by the automotive E2E standard. The model is validated also by comparing the data output of the E2E hardware with the data output of the AUTOSAR software E2E library. Finally, future needs and directions are suggested in this area.**

***Index Terms*—Automotive electronics; Vehicle safety; Data buses; Cyclic redundancy check codes.**

I. INTRODUCTION

To control the complex functionality of the nowadays cars, there is a need to integrate not only electronic devices based on microprocessors or microcontrollers, but also peripheral devices such sensors and actuators. Currently, car manufacturers design the architecture of the cars by using tens or hundreds of Electronic Control Units (ECUs), all of them sending or receiving data by using communication buses specific to the automotive domain (Local Interconnect Network (LIN), Controller Area Network (CAN), or Flexray, etc.). These systems have a close interaction with basic or smart sensors and actuators, can perform signal processing or run machine learning or other advanced control algorithms [1]. Independent of the complexity of the software that controls one process or another inside the car, there is a great need to ensure the safety of the passengers. The functional safety of road vehicles is defined and standardized in ISO 26262 which is an international standard for the functional safety of electrical and/or electronic systems that are installed in road vehicles [2]. This paper proposes an improvement of the model for the basic sensors that measure temperature, height level, speed, vibrations, etc. to be connected directly to the car buses while also respecting the safety level of communication.

The novelty of the proposed approach consists of the migration of several software mechanisms in hardware modules, with certain advantages related to safety communication and optimal use of available resources. To our knowledge, this approach has not yet been developed or implemented in basic sensors for automotive applications.

This paper begins with related works on automotive sensors and communication. Section III presents a brief overview of the car electronics architecture and basic notions about the safety levels in road vehicles. The following subsections describe the Automotive Open System Architecture (AUTOSAR) software architecture and the End-to-End (E2E) communication protection library in software [3]. Section IV describes the subject of the research: a hardware model of the E2E protection mechanism (designed for the sender node) proposed by the authors to be implemented in basic sensors. The final sections highlight the results and the conclusions of migrating this software module to hardware using a Xilinx Spartan 7 FPGA electronic device [4].

II. RELATED WORKS

Currently, the hardware/software implementations used in automotive industry are complex. These implementations are constantly evolving due to the need to bring new features that will lead to the development of more advanced cars in terms of traffic safety. The introduction of the

Manuscript received 12 February, 2022; accepted 17 June, 2022.
This research was supported by the Continental Automotive Systems Sibiu under Grant No. 3314/21.10.2020.

AUTOSAR architecture allowed the standardization of the software by describing the operating mechanisms, communication, diagnosis, application modules, etc. This standardization has also led to the development of sensors/actuators that are controlled by Electronic Control Units (ECUs). Current scientific work focuses on new methods to streamline these hardware/software mechanisms. Luckinger and Sauter [5] explore a new way of clock synchronization over the CAN bus. This fully AUTOSAR-compliant software method (with a direct impact on the CAN hardware) achieves a synchronization accuracy of around 400 microseconds, which leads to increased communication security between the car's ECUs. Another method to optimize communication on the CAN bus is presented in [6]. The authors propose an innovative method for packing frames on the Controller Area Network with Flexible Data Rate (CAN-FD) bus to optimize message traffic in communication networks. The optimization consists in packing in a single frame the signals that are transmitted cyclically in the same period, which leads to the reduction of the number of messages on the CAN-FD bus. Another direction of research is to optimize the sensors used in automotive. In [7], the authors present a multi-criteria optimization method based on Markov chains to obtain an efficient multi-sensor system that can be used in Advanced Driver Assistant System (ADAS) applications. Another approach in the development of electronic modules used in communication is to migrate software functionality to hardware. The purpose of migration is to increase the efficiency of the transmission/reception mechanisms to ensure the integrity and security of the data on the communication bus. In [8], the authors propose the migration to Field Programmable Gate Array (FPGA) of the transmission software for the standard AFDX messages used in satellite module communication, while in [9], they propose the migration to FPGA of the 32-bit Cyclic Redundancy Code (CRC-32) calculation algorithm.

III. ARCHITECTURE OF A CAR: HARDWARE/SOFTWARE OVERVIEW

A. Overview of the Car's Architecture

Each functionality of today's cars is controlled using one or several electronic control units based on microcontrollers or microprocessors. There are dedicated ECUs inside the car for controlling functionalities like engine control, braking system control, interior control or driving system (autonomous or not), etc. (Fig. 1). All these ECUs are interconnected through communication buses directly or via gateways. The logical data are transported over these buses and organized into messages (each message may contain one or more signals). There are several standardized communication protocols for automotive (but not limited to these): LIN, CAN/CAN-FD, or Flexray.

The Local Interconnect Network (LIN) protocol specifies a master-slave communication in which a master node can address one of the up to 16 slaves to ask for data. LIN is used in applications that require slow data rates (up to 20 kb/s). The Controller Area Network (CAN) protocol is a serial protocol with a data rate up to 1 Mb/s. CAN uses a message-oriented protocol. No addresses are defined for node addressing; an ECU can start to transmit if there are no messages on the bus. An arbitration policy is applied when several ECUs try to emit data simultaneously on the bus.

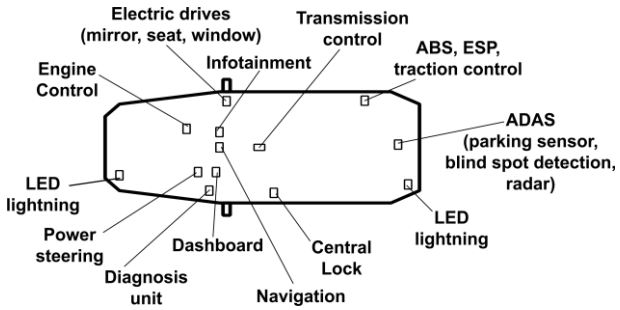

Fig. 1. Various automotive functionalities inside cars.

The ECU, which will try to send the message with the lowest identifier, will win the bus. The other ECUs will become receivers for providing the acknowledge and eventually storing the data [10]. Flexray is a time-driven protocol (serial data) used in safety critical systems due to the channel redundancy (there are two channels defined). It is based on internal timers and synchronization between nodes; the access to the bus is periodic, leading to a deterministic and predictable behaviour. The communication structure is organized into 64 cycles, each cycle having 5 milliseconds with a data rate of 10 Mb/s [11].

B. Safety in Automotive

Communication within the vehicle is an important part of the functional safety; Data transfer between ECUs and sensors/actuators must meet the ISO 26262 requirements. ISO 26262 specifies several safety levels (Automotive Safety Integrity Level (ASIL)) to be covered by the hardware/software/system defined for every ECU in the vehicle (Table I).

TABLE I. LEVELS OF SAFETY FACTORS IN AUTOMOTIVE.

Severity		**Exposure**		**Controllability**	
S0	No injuries	E0	Incredibly unlikely	C0	Controllable in general
S1	Light to moderate injuries	E1	Very low probability	C1	Simply controllable
S2	Severe to life-threatening injuries	E2	Low probability	C2	Normally controllable
S3	Life-threatening to fatal injuries	E3	Medium probability	C3	Difficult to control or uncontrollable
-	-	E4	High probability	-	-

Each ECU is ranked according to the determined ASIL level. This level is strictly related to the following factors (Table I): severity (damages in case of system failure), exposure (probability of failure occurrence), and controllability (the degree of control of the system made by the driver or external actions).

ISO 26262 defines four levels of ASIL, denoted from A to D. ASIL A is the minimum level of risk, and ASIL D is

the maximum. The standard also defines an additional level for not safety relevant applications - Quality Management (QM). The ASIL level of an automotive application (ECU) is determined by correlating all factors defined in Table I and using the matrix depicted in Fig. 2 [2].

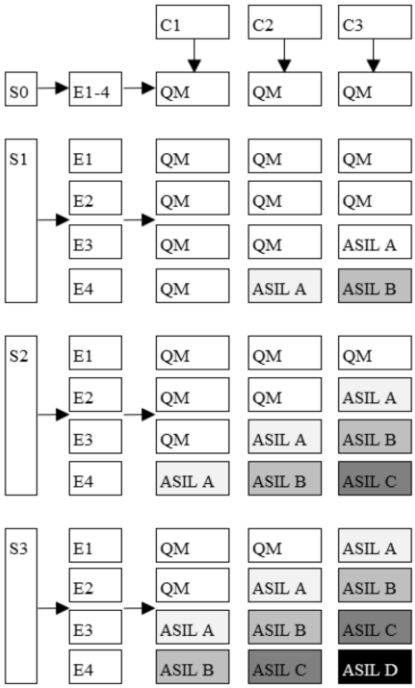

Fig. 2. Determining the ASIL level based on severity, exposure, and controllability [2].

C. Automotive Open System Architecture (AUTOSAR)

The complexity of automotive software requires hardware resources able to perform many tasks in parallel having limited computing power and memory resources. The current ECUs are based on multi-core processors integrated within microcontrollers. The management of an ECU is performed by a multi-core Real-Time Operating System (RTOS). The software that controls an ECU must be robust enough to meet the ASIL level required by its functionality. Robustness can be achieved if the software is standardized, which, in turn, will improve the performance and assure the required level of safety. This is the reason why automotive manufacturers defined a standardized software architecture to be used in automotive applications: AUTOSAR [12].

AUTOSAR is a layered software architecture with benefits in decreasing the dependence between hardware and software, decoupled software development, and software reusability. Currently, two platforms of AUTOSAR are defined: classic platform and adaptive platform. The classic platform is used for embedded systems with hard real-time and safety constraints, while the adaptive platform is used for high-performance computing ECUs to build fail-operational systems (e.g., autonomous driving) [13].

The layered software architecture defined in the classic platform is depicted in Fig. 3. There are three main software layers: Basic Software layer (BSW), Runtime Environment layer (RTE), and Application Software layer (ASW).

The basic software layer is used to control the hardware of the ECU and to provide services for communication, diagnostics, safety assurance, or access to different types of memories (Fig. 4).

The microcontroller abstraction layer contains software drivers with direct access to the microcontroller (MCU) and its peripherals. By using this layer, the higher software layers will be independent of the MCU. The ECU abstraction layer provides a set of Application Programming Interfaces (APIs) for accessing the peripherals/devices regardless of their location (MCU internal/external) and may contain drivers for external devices. Services layer provides basic services for BSW and ASW like OS functionality, network management and communication, memory management, diagnostics, state management, etc.

AUTOSAR RTE is a “glue” layer that allows communication between BSW modules and ASW modules called “Software Components” (SWCs), states transitions, tasks mapping, and more. Using this layer creates independence between basic and application software development. The AUTOSAR metamodel is a necessary input for both parties involved in the development of the mentioned software layers (BSW and ASW). This metamodel contains the interfaces, ports, runnable, and other structures used in BSW/ASW communication and is also used as input for the RTE software generation process. For this generation, there are several applications on the market [14] developed by automotive stakeholders such as Vector, Elektrobit, Continental, etc.

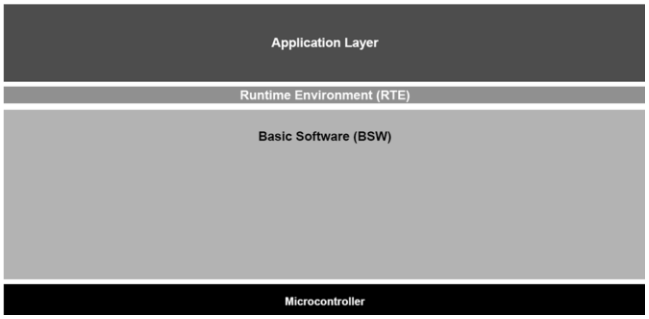

Fig. 3. AUTOSAR layered software architecture [13].

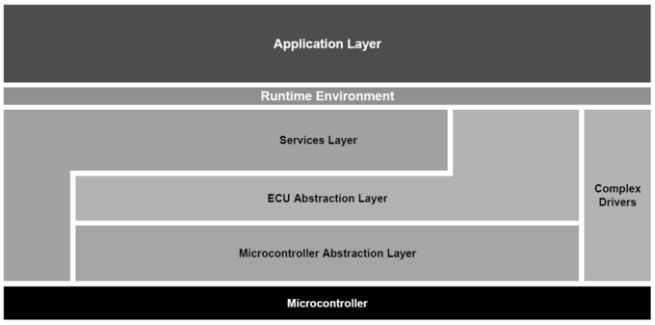

Fig. 4. A view inside the BSW layer in AUTOSAR [13].

D. Communication Stack in AUTOSAR

The communication (COM) stack is used to implement the logical communication between ECUs in a car. It consists of three different modules placed in BSW layers: communication drivers, communication hardware abstraction (interfaces), and communication services (Fig. 5).

Communication drivers represent software modules that have protocol-specific implementations and that control the

protocol-specific hardware (e.g., LIN, CAN, Flexray).

Communication hardware abstraction is a group of modules (protocol-specific) that provide an equal mechanism to access a bus channel regardless of its location (on-chip/on-board) [13]. Communication services consist of a set of software functions used to send/receive data, pack/unpack signals in/from a message, etc. These services are independent of the bus channel or the protocol used in communication.

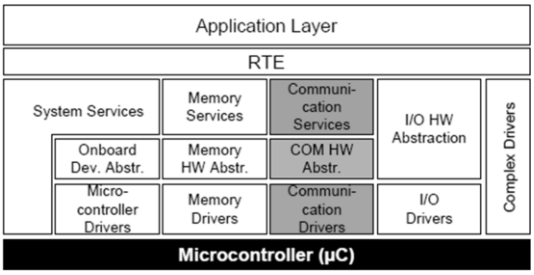

Fig. 5. BSW COM stack in AUTOSAR [8].

From a safety point of view, communication between ECUs must be protected at the two levels: hardware and software. Communication is protected on the hardware level by using a specific bus driver (ASIC). As an example, in case of a transmission ordered by the communication layer, this integrated circuit will pack the message (Data Field) into a frame to be sent on the bus as defined by the standard protocol. On the receiving side, error detection is performed in hardware based on the Cyclic Redundancy Code (CRC) sequence field.

The hardware verification mechanism is not enough from a safety perspective. As an example, a received message (Data field in frame) can be validated at the destination. As a result, it will be provided to the software modules in the communication stack, which, in turn, will provide the received data to ASW. In such a case, the data received and validated by the hardware can be altered along the transfer between software modules due to several factors: wrong configured identifiers, wrong configuration of the message inside AUTOSAR modules, wrong configured data paths, etc. To prevent such faults and to ensure the required safety level, AUTOSAR introduced a mechanism responsible for communication protection on the software level: end-to-end communication protection (E2E).

E. End-to-End (E2E) Communication Protection in AUTOSAR

The concept of end-to-end communication was first presented in 1973 by Branstadt [15]. The concept has been implemented on a large scale in banking application systems for high-level auditing procedures as a matter of policy and legal requirement [16]. The concept of E2E evolved over the years and the number of areas of applications has been increased. Therefore, E2E has been adopted by the industry as a standardized functions library. This library contains algorithms for data protection; the responsibility for the correct use of the library lies with the calling software module.

Using the AUTOSAR E2E library in software development guarantees a higher level of safety communication and allows for the runtime detection of hardware- or software-related errors (Fig. 6).

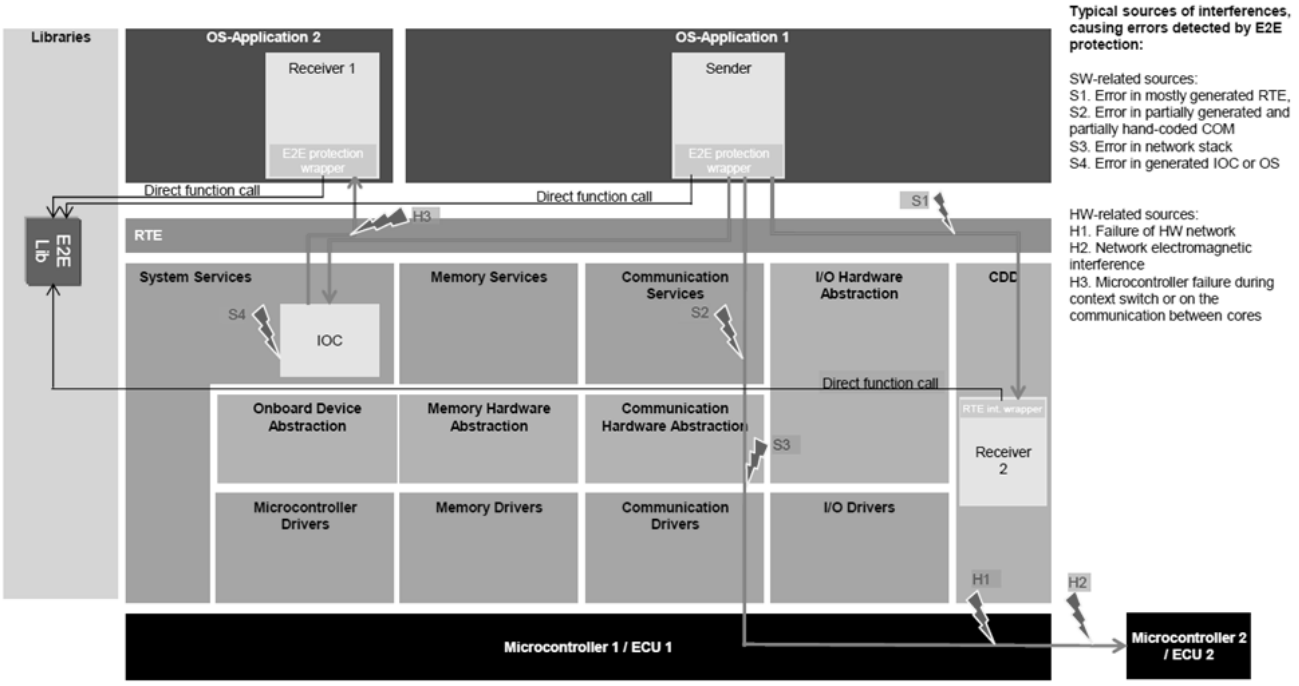

Fig. 6. Examples of faults mitigated by E2E protection in AUTOSAR applications [18].

The protection achieved by the E2E library consists in computing the CRC over the data to be sent/received in BSW [17], [18]. E2E data protection is based on the CRC calculation method defined in the CRC-8-SAE J1850 standard [19], [20] which specifies the computation procedures for different lengths of the CRC (8-bit, 16-bits, 24-bits, etc.). According to the E2E standard, the data must contain a counter which is incremented on each transmission. This counter will be used at the destination for timeout detection (incomplete received sequence of data) [21], [22]. Moreover, to calculate the CRC, a data identifier is needed (a fixed value defined for the application). This

Data ID can be implicit or explicit transmitted on the bus.

E2E defines several profiles to be used in AUTOSAR applications, such as:

– E2E Profile 1 (1A, 1B, and 1C variants) with 8 bits CRC, 16 bits of Data ID used for CRC computation (implicit or explicit transmitted on the bus), a 4 bits Counter (for timeout detection) and using the polynomial value 0x1D (Fig. 7 and Fig. 8);

– E2E Profile 2 - 8 bits CRC, a list of 8 bits Data ID addressed by the Counter, 4 bits Counter, polynomial value 0x2F;

– E2E Profile 4 - 32 bits CRC, 32 bits Data ID explicit transmitted on the bus, 16 bits Counter, polynomial value 0x1F4ACFB13;

– E2E Profile 5 - 16 bits CRC, 16 bits Data ID implicit transmitted on the bus, 8 bits Counter, polynomial value 0x1021;

– E2E Profile 6 - 16 bits CRC, 16 bits Data ID implicit transmitted on the bus, 8 bits Counter, polynomial value 0x1021, 16 bits Length field for variable length data.

The principle of using the E2E mechanism in AUTOSAR is depicted in Fig. 9.

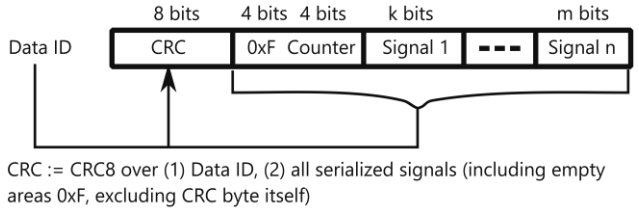

Fig. 7. Message content according to E2E Profile 1A (implicit transmission of Data ID) [17].

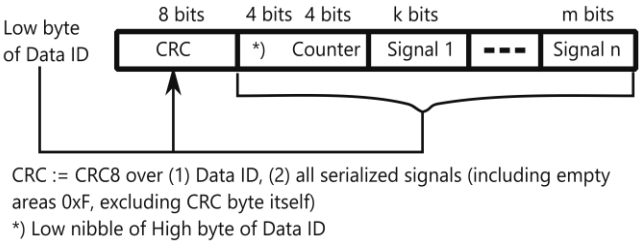

Fig. 8. Content of the message according to the E2E Profile 1C (explicit transmission of Data ID) [17].

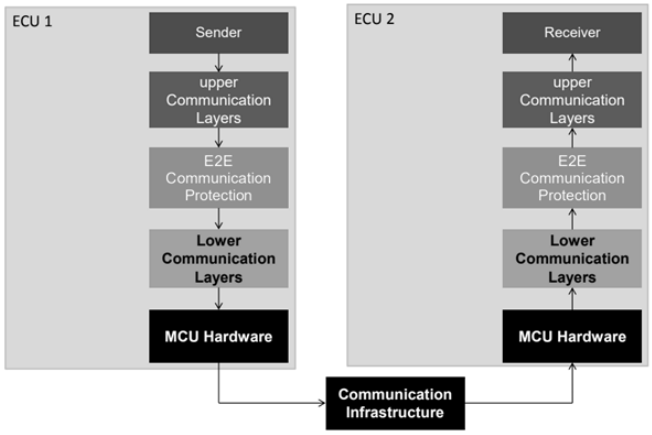

Fig. 9. E2E communication protection principle in AUTOSAR [17].

On the sender side, the data to be transmitted are protected by calling the E2E_Protect function that will increment the Counter and will calculate the CRC over the entire data, including Counter.

The message that will be sent on the bus will contain along the data produced by the application (BSW or ASW) also the calculated CRC and Counter. On the receiver side, a similar E2E mechanism is applied. The consumer will call the E2E_Check function, which will check the Counter to be in the right sequence (for timeout detection) and will calculate the CRC over the received data (excluding received CRC).

Finally, it compares the calculated CRC with the received CRC. If these values are equal, the data are safe to be used. Otherwise, they are not. The result is communicated to the consumer application that will use the data or not, according to the safety requirements.

IV. Migration of E2E Communication Protection Library to Hardware

A. Hardware E2E Module for Basic Sensors

At this time, the basic sensors in automotive are connected directly on an ECU through a standard interface like SPI, I2C, PSI5, ADC, etc. (Fig. 10) [23]. In this case, the physical data are read by the ECU1 which transforms and protects the data (using the E2E mechanism) to use it and to be sent on the configured bus channel. The other ECUs will read the data from the bus, will check the E2E status, and if it is valid, will use it inside the BSW or ASW according to the specification.

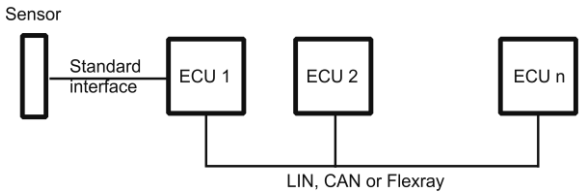

Fig. 10. Connection of a basic sensor inside the car.

The aim of this paper is to design and evaluate an E2E hardware module for basic sensors [24], [25] capable of sending protected E2E data on an automotive communication bus. Figure 11 depicts the integration of the proposed sensor into an automotive communication network.

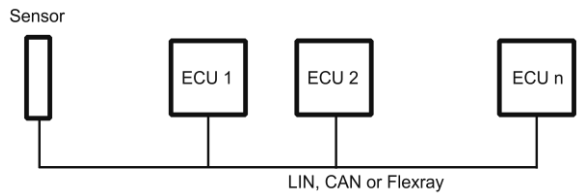

Fig. 11. Basic sensor with E2E protection communicating in an automotive network.

A customized message provided by the sensor has been defined. It includes CRC, Counter fields (according to the E2E Profile 1A), and Signal A, which represents the raw value of the physical value measured by the sensor. The format of the message on the bus is depicted in Fig. 12.

To realize the integration of the new sensor into an

automotive network, several changes are required:

– The sensor must have a converter able to transform the physical value into the raw value (logic value) for generating Signal A;

– Add logic able to handle the Counter value (rolling value);

– Add logic able to provide the Data ID (fixed value);

– To implement a fast CRC calculation algorithm, the sensor must have an internal ROM memory to store a Look-Up Table (LUT) with 256 elements [26]. LUT values will be used in XOR operations for generating the CRC;

– Adding a hardware bus driver/transceiver (LIN, CAN or Flexray) used in communication on the bus (this is not in the scope of this paper).

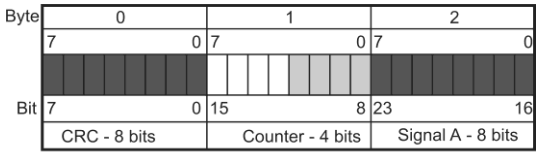

Fig. 12. Sensor message format according to E2E Profile 1A.

The CRC computing algorithm is performed by the sensor according to the E2E Profile 1A specifications. The logic diagram of the algorithm is presented in Fig. 13.

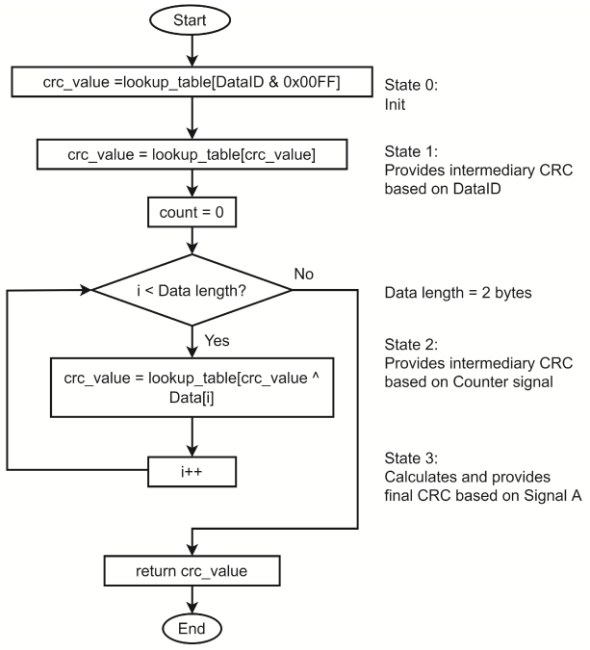

Fig. 13. Logic diagram of the CRC computing algorithm (E2E Profile 1A).

B. Detailed Design of Hardware E2E Module

Field Programmable Gate Array (FPGA) is an integrated circuit that can be programmed using a Hardware Description Language (HDL) to be used for specific custom applications. An FPGA contains an array of hierarchical Configurable Logic Blocks (CLBs) and reconfigurable interconnects that can be configured according to the desired application [27].

The Xilinx Vivado 2021.2 tool [28] has been used to design, synthesize, validate, and integrate the E2E module in an FPGA device. Vivado allows the user to design the hardware using basic block elements like basic registers, logic gates, ALU units, etc. Moreover, Vivado allows user to develop new logical blocks (user customized) by supporting VHDL and Verilog hardware description languages. After the design phase, the user has the possibility to simulate the design, analyse the results, synthesize it into FPGA, and finally run it directly on an FPGA device [29], [30]. The results presented in this paper were obtained using a Xilinx Spartan 7 FPGA device which is the highest density device in the Spartan-7 family [31].

The block diagram of the E2E module for basic sensors is presented in Fig. 14 and consists of input buffers (Data ID and Signal A), clock generator, a 4-state state machine, a CRC computing logic (including LUT), and output buffers for generated signals (CRC, Counter, and Signal A).

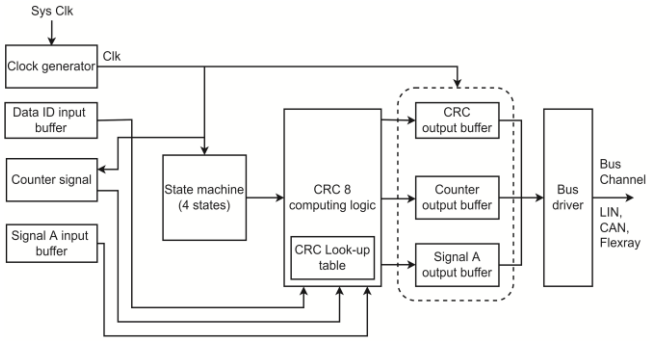

Fig. 14. Block diagram of the E2E module for basic sensors (Profile 1A).

The 4-state state machine controls the CRC computing logic. The meanings of the states mentioned in Fig. 13 are as follows:

– State 0: The initialization value is read from the LUT based on the Data ID and the start value 0xFF;

– State 1: The first intermediary CRC is calculated;

– State 2: The second intermediary CRC is calculated using Counter value;

– State 3: The final CRC value is calculated using Signal A and stored onto the output buffer.

Each state contains four-clock cycles along the value of the CRC is calculated (Fig. 15):

– Clock 0: The address of a location in the LUT is calculated (XOR operation);

– Clock 2: The data addressed in the LUT is read and provided in an internal buffer for the next step (state);

– Clock 3 on state 3: The final values of the CRC, Counter, and Signal A, are stored into output buffers. In this step, the values of the signals are ready to be processed by the bus driver to be sent on the communication bus.

Five steps that are necessary to be performed in the Vivado tool to implement the E2E hardware module in FPGA (Figs. 16–18):

1. Create a block design based on the block diagram (Fig. 14). The design contains predefined or customized logical blocks. Verilog hardware description language has been used in designing the E2E module to describe the run-time libraries (RTL) of customized inner blocks. The detailed design of the CRC 8 computing logic block (Fig. 14) is shown in Fig. 16.

2. Create a detailed design. In this phase, the block design is mapped on standard basic logical blocks (Fig. 17).

3. Synthesize the design. The next step consists of mapping the synthesized design onto the FPGA structure. Logical basic blocks are mapped onto the CLBs inside the FPGA configured in the project. The schematic is complex and hard to depict in one figure.

4. Implement the design. The synthesized design is placed into FPGA onto specific CLBs. In this phase, it is possible to make changes related to the placement of the modules inside the FPGA area. Figure 18 shows the design implemented inside the Xilinx Spartan 7 FPGA, where the CLBs are marked with black rectangles.

5. Generate the bitstream and program the FPGA. It is necessary to have the development kit connected to the computer and to "write" the generated bitstream inside the FPGA device.

The E2E block design is based on Profile 1A, in which the CRC contains Data ID. Other E2E profiles require Data ID to be explicitly sent to the bus, which leads to a design change (due to a change in the sensor message).

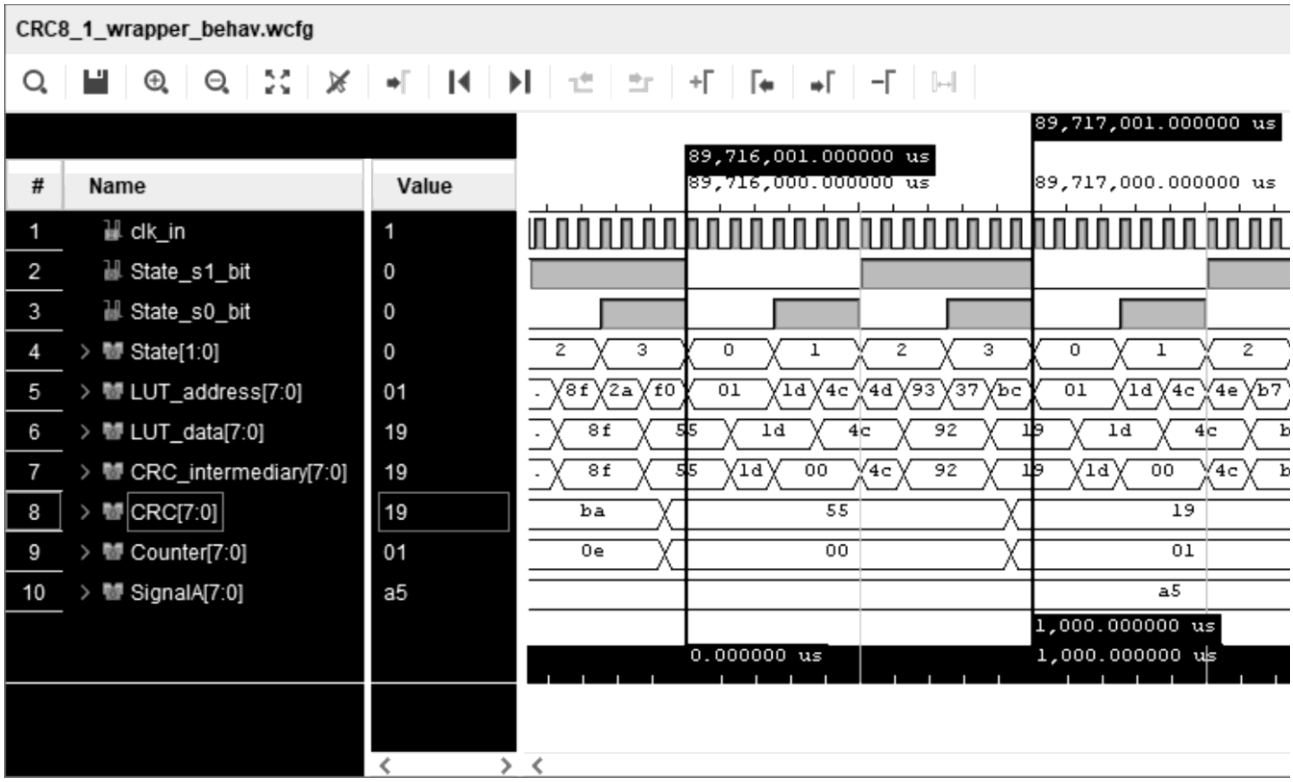

Fig. 15. Waveforms of the state cycles and CRC values (E2E Profile 1A).

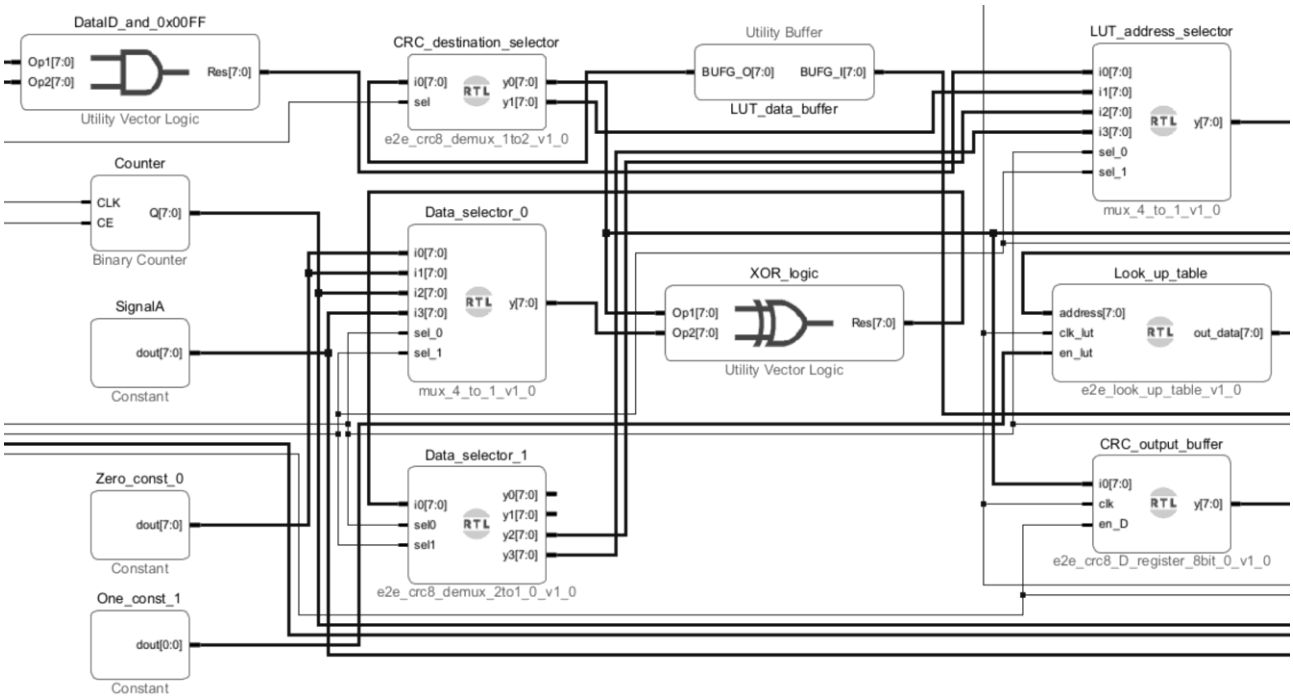

Fig. 16. Block design of the E2E module for the basic sensor. A view inside the CRC8 computing logic block.

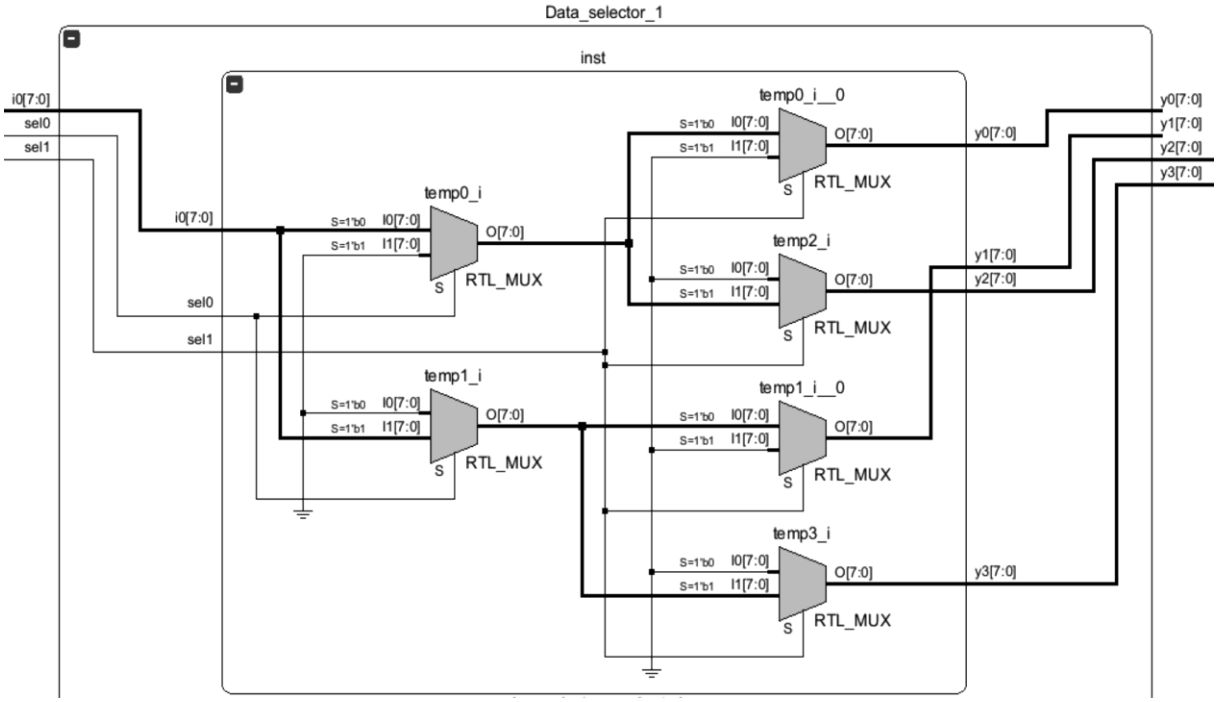

Fig. 17. Detailed design of the E2E module for the basic sensor. A view inside the Data_selector_1 block.

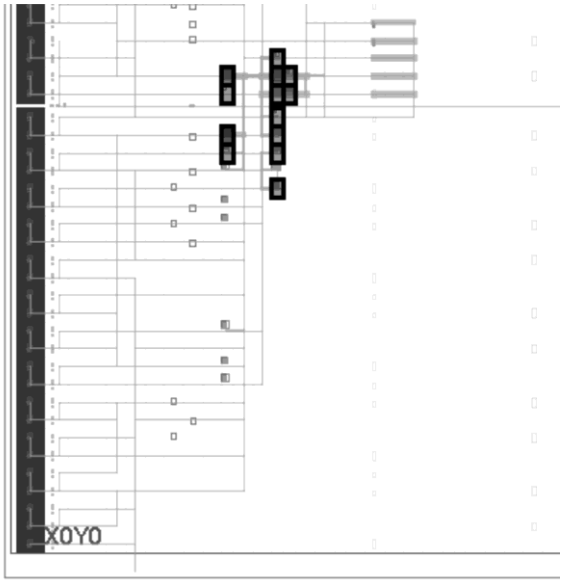

Fig. 18. Implemented design of the E2E module in the Xilinx Spartan 7 FPGA.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The simulation of the E2E module (Vivado tool) is based on a clock signal with a period of 62.5 microseconds. Using this timing, the sensor will send a message on the bus on every millisecond, which is a reasonable data rate for a CAN bus, for example. The results reflect the functionality of the E2E module. The generation of the Signal A and the implementation of the bus driver are not within the scope of this paper. The range of the Counter signal is 0x00…0x0E (0x0F value represents an invalid value). Signal A had a constant value (0xA5) during the experiments and is represented on 8 bits (see Fig. 12), while Data ID was set to the value 1. The output values of the hardware module E2E are presented in Table II and Fig. 19.

TABLE II. OUTPUT VALUES OF CRC, COUNTER, AND SIGNAL A.

Cycle	CRC	Counter	Signal A
0	0x55	0x00	0xA5
1	0x19	0x01	
2	0xCD	0x02	
3	0x81	0x03	
4	0x78	0x04	
5	0x34	0x05	
6	0xE0	0x06	
7	0xAC	0x07	
8	0x0F	0x08	
9	0x43	0x09	
10	0x97	0x0A	
11	0xDB	0x0B	
12	0x22	0x0C	
13	0x6E	0x0D	

These results were also validated using a different environment based on the Vector CANalyzer tool [32]. In this environment, the computer is the sender (via Vector VN 7610 hardware), while the receiver (and checker) is a real ECU. Communication is carried out using a real CAN network. The values provided by the E2E software library implemented on a computer using the CAPL scripting language [33] were captured from the real CAN bus. Figure 20 represents the waveforms of the signals captured on the CAN bus (the values of the signals are displayed in decimal), while Fig. 21 shows the data values of the signals (hexadecimal representation).

A comparison of the data shown in Table II and Fig. 21 shows that the values are similar, which validates the proposed hardware model designed in the Vivado tool. Furthermore, on the receiver side, the ECU reported no errors in checking the E2E status of each received message, which means that the values generated by the E2E module are correct.

The utilization rate of the FPGA's CLBs is presented in Fig. 22. It is observed that for the E2E hardware module, there is a need of 46 Look-Up Table (LUT) elements for logic blocks, 30 Flip-Flop registers, 1 block RAM, and 54 I/O pins.

Considering that one CLB inside Xilinx Spartan 7 FPGA contains 8 LUTs, 16 Flip-Flop registers and 256 bits of RAM [31], it results that there is a need of 13 CLBs (Fig. 18) to integrate E2E module in FPGA. To be mentioned that the implementation of an E2E module able to handle more than 2 bytes of data will require supplementary hardware resources.

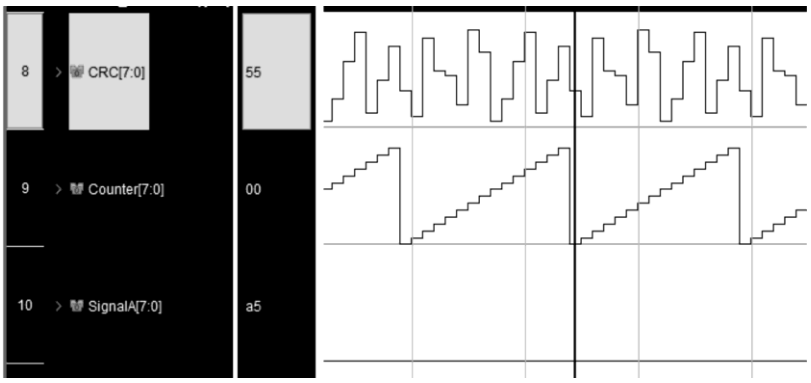

Fig. 19. Waveforms of output signals CRC, Counter, and Signal A obtained by simulating the E2E hardware model.

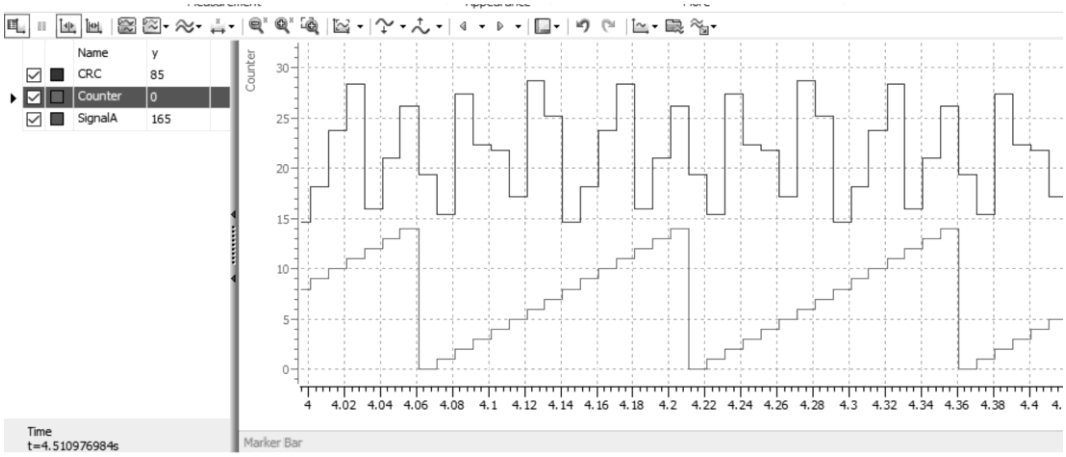

Fig. 20. Waveforms of the signals CRC, Counter, and Signal A captured on a real CAN bus.

2.110747 CAN 1	100	Sensor	CAN Frame	Tx	3	55 00 A5
2.120613 CAN 1	100	Sensor	CAN Frame	Tx	3	19 01 A5
2.130517 CAN 1	100	Sensor	CAN Frame	Tx	3	CD 02 A5
2.140601 CAN 1	100	Sensor	CAN Frame	Tx	3	81 03 A5
2.150555 CAN 1	100	Sensor	CAN Frame	Tx	3	78 04 A5
2.160705 CAN 1	100	Sensor	CAN Frame	Tx	3	34 05 A5
2.170589 CAN 1	100	Sensor	CAN Frame	Tx	3	E0 06 A5
2.180543 CAN 1	100	Sensor	CAN Frame	Tx	3	AC 07 A5
2.190541 CAN 1	100	Sensor	CAN Frame	Tx	3	0F 08 A5
2.200537 CAN 1	100	Sensor	CAN Frame	Tx	3	43 09 A5
2.210487 CAN 1	100	Sensor	CAN Frame	Tx	3	97 0A A5
2.220509 CAN 1	100	Sensor	CAN Frame	Tx	3	DB 0B A5
2.230497 CAN 1	100	Sensor	CAN Frame	Tx	3	22 0C A5
2.240573 CAN 1	100	Sensor	CAN Frame	Tx	3	6E 0D A5
2.250649 CAN 1	100	Sensor	CAN Frame	Tx	3	BA 0E A5

Fig. 21. Data values generated by the E2E software library on the real CAN bus.

Resource	Utilization	Available	Utilization %
LUT	46	64000	0.07
FF	30	128000	0.02
BRAM	1	120	0.83
IO	54	400	13.50

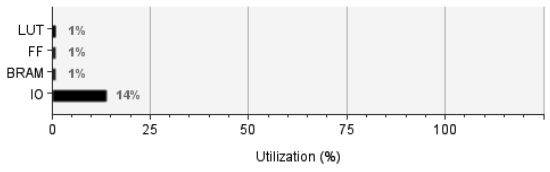

Fig. 22. Utilization rate of Xilinx Spartan 7 FPGA resources.

The results presented in this paper were obtained using a Xilinx Spartan 7 FPGA device placed on the Spartan-7 SP701 evaluation board (Fig. 23). The SP701 evaluation board is based on the XC7S100FGGA676 device, a member of the Xilinx 7 series FPGA family. It is optimized for low cost, low power, and high I/O performance. It comes with advanced high-performance FPGA logic based on real 6-input LUT, 36 Kb dual-port block RAM, support for DDR3L interface up to 1866 Mb/s, XADC with 12-bit 1 MSPA ADC with on-chip thermal and supply sensors, and powerful Clock Management Tiles (CMTs). The board is designed for high-performance and lower power with a 28 nm, 1 V core voltage process [34].

The standard AUTOSAR E2E library is used in automotive applications to ensure the safety of the communication in automotive networks. Right now, this mechanism is implemented in software on the sender and receiver sides. This paper presents a hardware model of the AUTOSAR E2E software library for use it inside of the basic sensors in automotive. The new hardware module will allow the sensor to send protected data according to the

safety requirements in communication with the ECUs via the automotive network (Fig. 11).

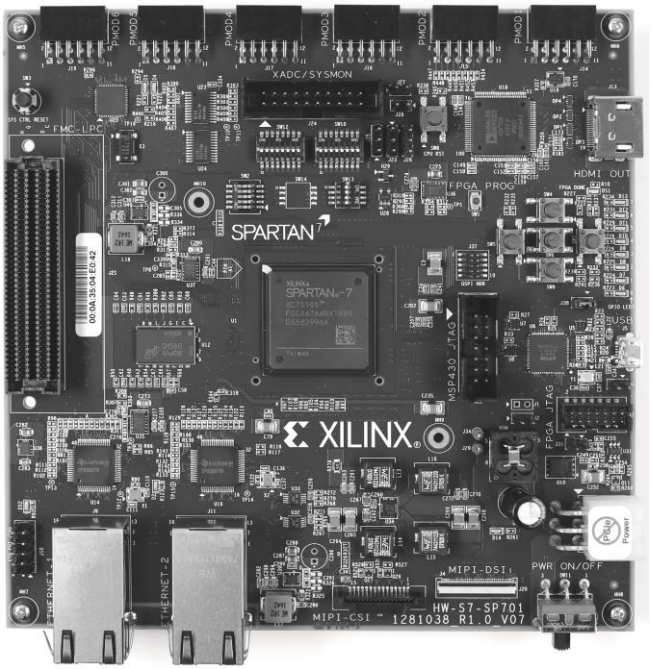

Fig. 23. Spartan-7 SP701 evaluation board [34].

This new approach has the advantage that all the ECUs will have direct access to the protected data provided by the sensor. In this case, there is no need for a specific ECU to compute the values of the protection signals (CRC and Counter) in software and to forward it towards other ECUs (nodes) into the network. This mitigates the risk of error occurrence in software or incorrect message sending on the bus. Regarding the timing, the output data rate of the model depends on the data rate on the communication bus (LIN, CAN, or Flexray) which is few milliseconds or tens of milliseconds. Either SW or HW implementation must fulfil these timings; in both cases the internal timing of producing data is not relevant for the performance itself as long as this timing is not greater than the data communication rate. Another advantage is that the new sensors can be placed anywhere inside the car because they can provide data directly to the communication bus. In case of acceleration sensors, e.g., they are placed in some cases directly on the ECU, and this ECU should be placed in the car on a certain position to be sure that they make the right measurements. By using the new sensor, the placement of the ECU is not a constraint. It is important in this case that only the sensor (small size) is placed at the right position.

VI. CONCLUSIONS

After all the work, the conclusion is that the AUTOSAR E2E protection communication mechanism is feasible to implement in hardware since, in case of a message of 3 bytes length (3 signals) in FPGA, it requires 46 Look-Up Table (LUT) elements for logic blocks, 30 Flip-Flop registers, 1 block RAM, and 54 I/O pins. The utilization rate of the internal modules of FPGA is 0.1 %, while for CRC LUT is 0.83 %, and for I/O pins is 13.5 %. By increasing the number of bytes/message, only the utilization rate of internal modules will be affected (increased number for Flip-Flop registers) since the CRC LUT and I/O pins will remain the same. The proposed hardware is capable of providing data on each millisecond (using a clock of 62.5 microseconds), which is an acceptable data rate for CAN communication. This data rate can be adjusted (according to the requirements) by increasing the clock timing inside the FPGA.

Future work will consist in designing an E2E module capable of processing many signals of a message to calculate the CRC that will have impact on the hardware complexity (e.g., a memory module to store the signals is needed). Computing the final CRC value will require many clocks per state in the 4-state machine (1 clock/byte). Another direction in this research to improve the concept presented is to migrate all the E2E defined profiles to hardware at the sending site [9]. Moreover, there is room to exploit this experience to also integrate the checking part on the receiver side, e.g., the integration of an E2E checking module into actuators. The new actuator models can benefit from such a hardware module to check the safety data received from the communication bus.

Another direction in future research is to investigate the possibility of integrating the E2E protection mechanism together with the communication driver inside of an FPGA [35] and making qualitative and quantitative evaluations of the new hardware. If the results meet the expectations, then the proposed model could be directly used as a baseline for future sensors/actuators in the automotive industry.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] C. Falconi and S. Mandal, "Interface electronics: State-of-the-art, opportunities and needs", *Sensors and Actuators A: Physical*, vol. 296, pp. 24–30, 2019. DOI: 10.1016/j.sna.2019.07.002.

[2] *Road Vehicles - Functional Safety*, International Standardization Organization ISO 26262-1:2018.

[3] *AUTOSAR Standards*. [Online]. Available: https://www.autosar.org/standards/

[4] *Spartan-7 FPGAs: Meeting the Cost-Sensitive Market Requirements*, Xilinx Inc. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/white_papers/wp483-spartan-7-intro.pdf

[5] F. Luckinger and T. Sauter, "AUTOSAR-compliant clock synchronization over CAN using software timestamping", in *Proc. of 2021 17th IEEE International Conference on Factory Communication Systems (WFCS)*, 2021, pp. 49–52. DOI: 10.1109/WFCS46889.2021.9483588.

[6] W. Ma, G. Xie, R. Li, W. Liu, H. H. Li, and W. Chang, "Efficient AUTOSAR-compliant CAN-FD frame packing with observed optimality", in *Proc. of 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1899–1904. DOI: 10.23919/DATE51398.2021.9473962.

[7] M. Qiu, T. Antesberger, and R. German, "Multi-sensor system simulation based on RESTART algorithm", in *Proc. of 2021 Annual Reliability and Maintainability Symposium (RAMS)*, 2021, pp. 1–6. DOI: 10.1109/RAMS48097.2021.9605759.

[8] F. Molina, P. Corral, M. Aljaro, G. de Scals, and A. Rodriguez, "Implementation of an AFDX interface with Zynq SoC board in FPGA", *Elektronika ir Elektrotechnika*, vol. 26, no. 5, pp. 11–15, Oct. 2020. DOI: 10.5755/j01.eie.26.5.26008.

[9] J. Mitra and T. Nayak, "Reconfigurable very high throughput low latency VLSI (FPGA) design architecture of CRC 32", *Integration*, vol. 56, pp. 1–14, 2017. DOI: 10.1016/j.vlsi.2016.09.005.

[10] *Introduction to CAN*, Vector GmbH, 2021.

[11] A. Hafeez, H. Malik, O. Avatefipour, P. Rongali, and S. Zehra, "Comparative study of CAN-Bus and FlexRay protocols for in-vehicle communication", SAE Technical Paper 2017-01-0017, 2017.

DOI: 10.4271/2017-01-0017.
[12] M. Staron, *Automotive Software Architectures: An Introduction*, 2nd ed. Springer Cham, 2021. DOI: 10.1007/978-3-030-65939-4.
[13] *AUTOSAR Layered Software Architecture*, AUTOSAR organization. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
[14] S. Piao, H. Jo, S. Jin, and W. Jung, "Design and implementation of RTE generator for automotive embedded software", in *Proc. of 2009 Seventh ACIS Int. Conf. on Software Engineering Research, Management and Applications*, 2009, pp. 159–165. DOI: 10.1109/SERA.2009.35.
[15] D. K. Branstad, "Security aspects of computer networks", in *Proc. of AIAA Computer Network Systems Conference*, 1973, AIAA Paper no. 73–427.
[16] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design", *ACM Trans. Comp. Syst.*, vol. 2, no. 4, pp. 277–288, 1984. DOI: 10.1145/357401.357402.
[17] *Specification of SW-C End-to-End Communication Protection Library*, AUTOSAR Standards, Document ID 428: AUTOSAR_SWS_E2ELibrary. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_E2ELibrary.pdf
[18] *E2E Protocol Specification*, *AUTOSAR FO R20-11*, AUTOSAR Standards, Document ID 849: AUTOSAR_PRS_E2EProtocol, 2020. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/foundation/20-11/AUTOSAR_PRS_E2EProtocol.pdf
[19] *Standard SAE-J1850 8-bit CRC*, SAE, 2006.
[20] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) polynomial selection for embedded networks", in *Proc. of Int. Conf. on Dependable Systems and Networks, 2004*, 2004, pp. 145–154. DOI: 10.1109/DSN.2004.1311885.
[21] T. Forest and M. Jochim, "On the fault detection capabilities of AUTOSAR's End-to-End communication protection CRC's", SAE Technical Paper 2011-01-0999, 2011. DOI: 10.4271/2011-01-0999.
[22] T. Arts and S. Tonetta, "Safely using the AUTOSAR End-to-End protection library", in *Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science()*, vol. 9337. Springer, Cham, 2015, pp. 74–89. DOI: 10.1007/978-3-319-24255-2_7
[23] W. J. Fleming, "Overview of automotive sensors", *IEEE Sensors J.*, vol. 1, no. 4, pp. 296–308, 2001. DOI: 10.1109/7361.983469.
[24] M. A. Pillai, S. Veerasingam, and Y. S. D., "Implementation of sensor network for indoor air quality monitoring using CAN interface", in *Proc. of Int. Conf. on Advances in Computer Eng.*, 2010, pp. 366–370. DOI: 10.1109/ACE.2010.85.
[25] Y. Xie, Y. Guo, S. Yang, J. Zhou, and X. Chen, "Security-related hardware cost optimization for CAN FD-based automotive cyber-physical systems", *Sensors*, vol. 21, no. 20, p. 6807, 2021. DOI: 10.3390/s21206807.
[26] X. Dong and Y. He, "CRC algorithm for embedded system based on table lookup method", *Microprocessors and Microsystems*, vol. 74, art. 103049, 2020. DOI: 10.1016/j.micpro.2020.103049.
[27] S. M. S. Trimberger, "Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology: This paper reflects on how Moore's law has driven the design of FPGAs through three epochs: the age of invention, the age of expansion, and the age of accumulation", *IEEE Solid-State Circuits Magazine*, vol. 10, no. 2, pp. 16–29, 2018. DOI: 10.1109/MSSC.2018.2822862.
[28] *Vivado Design Suite User Guide*, Xilinx Inc. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_2/ug910-vivado-getting-started.pdf
[29] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection", in *Cryptographic Hardware and Embedded Systems - CHES 2007. CHES 2007. Lecture Notes in Computer Science*, vol. 4727. Springer, Berlin, Heidelberg, 2007, pp. 63–80. DOI: 10.1007/978-3-540-74735-2_5.
[30] I. Z. Mihu and H. V. Căpriță, "Architectural improvements and FPGA implementation of a multimodel neuroprocessor", in *Proc. of 9th Int. Conf. Neural Inform. Proc.*, 2002, pp. 1749–1753, vol. 4. DOI: 10.1109/ICONIP.2002.1198975.
[31] 7 *Series FPGAs Configurable Logic Block*, Xilinx Inc. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB
[32] *CANalyzer the Tool for Comprehensive ECU and Network Analysis*, Vector GmbH, 2016. [Online]. Available: https://cdn.vector.com/cms/content/products/canalyzer/canalyzer/Docs/Fact%20Sheets/CANalyzer_FactSheet_EN.pdf
[33] *CAPL Used with CANoe and CANalyzer*, Vector GmbH, 2015. [Online]. Available: https://www.vector.com/gb/en/know-how/capl/#
[34] *SP701 Evaluation Board User Guide*, Xilinx Inc. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug1319-sp701-eval-bd
[35] Y. Son, J. Park, and T. Kang, "Design and implementation of CAN IP using FPGA", *Journal of Institute of Control, Robotics and Systems*, vol. 22, no. 8, pp. 671–677, 2016. DOI: 10.5302/J.ICROS.2016.16.0089.