Design and Evaluation of a Novel ECU Architecture for Secure and Dependable Automotive CPS

Bikash Poudel* and Arslan Munir†
Department of Computer Science and Engineering
University of Nevada, Reno
Email: *bpoudel@nevada.unr.edu, †arslan@unr.edu

***Abstract*—In this paper, we propose a new architecture for automotive ECUs that incorporates security and dependability primitives with negligible performance, energy, and resources overhead. We implement our proposed ECU architecture on Xilinx Automotive (XA) Spartan-6 FPGA. We demonstrate the effectiveness of our proposed architecture using a steer-by-wire (SBW) application over controller area network with flexible data rate (CAN FD) as a case study that considers major security-relevant use cases and constraints. We also optimize and implement a prior secure and dependable automotive work on the NXP quad-core iMX6Q SABRE automotive board. We further quantify performance, energy, and error resilience of our proposed architecture for our SBW case study. Results reveal that our proposed architecture can attain a speedup of 47.93× while consuming 2.4× lesser energy than optimized SABRE board implementation.**

***Index Terms*—Automotive, cyber-physical systems, multicore, FPGA, steer-by-wire, security, dependability**

I. Introduction and Motivation

Modern automobiles (also known as cybercars) are intricate distributed cyber-physical systems (CPS) comprising of hundreds of heterogeneous digital processors, numerous radio interfaces, hundreds of megabytes of complex embedded software, and multiple in-vehicle networks and protocols. The controller area network (CAN) is the most prevalent protocol for communication among ECUs in automotive CPS. To enhance energy efficiency, automated control, and user comfort, modern automobiles are forsaking traditional mechanical and hydraulic subsystems in favor of x-by-wire subsystems. However, x-by-wire systems have stringent real-time performance and reliability requirements, which pose significant challenges for implementation over limited bandwidth CAN. FlexRay with high speed data transfer and inherent fault tolerance features is a suitable replacement for CAN protocol. Nevertheless, the automotive industry is reluctant to adopt the FlexRay communication protocol because the transition requires a major revamp of the automotive electronic subsystems. CAN with flexibe data-rate (CAN FD) bridges the gap between CAN and FlexRay, and provides an easier alternative for implementing new automotive control applications, such as x-by-wire.

Realization of next generation automotive CPS applications, such as x-by-wire, requires incorporating dependability and security features in automotive ECUs and in-vehicle networks. Automotive CPS applications have stringent dependability requirements as stipulated by ISO 26262 [1]. The adherence to the ISO 26262 standard requires that at least one critical fault must be tolerated by the automotive applications without loss of functionality. Automobiles have to endure harsh operational environments (including external noise and radiations) that render electronic systems vulnerable to permanent and transient faults. Permanent faults can impair or stop the correct functionality of the system while transient faults induce soft errors in the system. Additionally, in-vehicle distributed control systems are traditionally designed without security in mind. All of the current in-vehicle communication protocols, such as CAN, CAN FD, and FlexRay, carry messages in plaintext format, which can be read and altered by any device connected to the bus system. These security threats are exacerbated by the increasing integration of automotive systems with external entities, such as consumer electronics, other vehicles, and networks.

The cardinal challenge in automotive CPS design is to integrate security and dependability while ensuring that hard real-time constraints of the automotive CPS applications are not violated. This paper addresses this cardinal challenge of automotive CPS design while also minimizing energy consumption. Temporal performance (meeting timing constraints) is often considered as a system's quality of service (QoS) measure. We affirm that the system's QoS must also be construed as a dependability measure that can impact the system's availability and safety beyond a certain *critical threshold* as the driver can totally lose the control of his/her car beyond that critical threshold. This critical threshold also defines the notion of behavioral reliability, which can be used to ensure the stability of the automotive CPS design [2]. Earlier works have addressed some of the security and dependability issues of in-vehicle distributed systems, vehicle-to-infrastructure communication, and vehicle-to-vehicle communication [2] [3]. [3] is the most relevant work to this paper, in which the authors proposed a secure and dependable approach for automotive systems, and presented a primitive implementation of the proposed approach on an Intel processor. However, the work did not implement the proposed approach on an automotive ECU. To overcome the limitations of earlier work, we propose a new secure and dependable automotive ECU architecture and compare it with prior multicore-based ECU architectures with respect to temporal performance, energy efficiency, and error resilience. Our main contributions are as follows:

- We propose a novel ECU architecture that incorporates security and dependability primitives while minimizing energy consumption and ensuring that real-time constraints of the automotive CPS applications are satisfied.
- Enhancement of the security structure of a prior se-

cure and dependable automotive approach (Munir et al. [3]), which we refer to as "baseline design" (BD). We reinforce the security of the BD by replacing secure hash algorithm-2 (SHA-2) based hash-based message authentication code (HMAC) with SHA-3 based HMAC. We further optimize and implement the BD on the NXP iMX6Q SABRE automotive board. We refer to this optimized BD as OBD.
- Implementation of our proposed ECU architecture on Xilinx Automotive (XA) Spartan-6 FPGA, which we refer to as EAF.
- Analysis of our proposed ECU architecture and approach using a SBW application over CAN FD as a case study. We compare the temporal performance, energy efficiency, and error resilience of our proposed ECU architecture with the existing state-of-the-art ECU architectures, such as NXP iMX6Q SABRE automotive board.

II. Related Work

Security for automotive embedded systems has been studied in prior works. Koscher et al. [4] analyzed internal and external attack surfaces of a modern automobile through which an attacker could control automotive subsystems (e.g., engine, brakes, windshield wipers) while ignoring the drivers input. Various prior works [5] discussed integration of message authentication codes (MACs) in CAN data frames to secure in-vehicle communication. In [5], security mechanisms based on MACs and counters were proposed specifically for CAN that could prevent masquerade and replay attacks. However, these works did not consider FT and message confidentiality while analyzing security over the CAN bus. Wolf et al. [6] presented a vehicular hardware security module (HSM) that was implemented in Xilinx Virtex-5 FPGA to secure in-vehicle ECUs and their communication. However, HSM did not incorporate any FT features. Furthermore, the work did not evaluate the interplay of performance, energy, and FT which is essential for secure and dependable cybercar applications.

Several earlier work explored dependability for automotive embedded systems. Beckschulze et al. [7] investigated FT approaches based on dual-core microcontrollers. Baleani et al. [8] discussed various FT architectures for automotive applications including lock-step dual processor architecture, loosely-synchronized dual processor architecture, and triple modular redundant architecture.

Munir et al. [3] presented a multi-core ECU based design of secure and dependable cybercars using SBW application as a case study. However, the work did not implement the proposed approach on an automotive-grade processor. This work is an extension of the work done by Munir et al. [3]. We enhance the security structure of the secure and dependable approach proposed in [3] (referred to as BD), and further optimize and implement the BD on an automotive-grade processor (referred to as OBD).

III. Secure and Dependable ECU Architecture

This section first summarizes our secure and dependable approach for cybercar design that leverages our proposed ECU. We then present the contemporary security and dependability standards employed in state-of-the-art automotive ECUs. We further elaborate our proposed ECU architecture focusing on the architecture's dependability and security aspects.

A. Secure and Dependable Approach for Cybercar Design

We leverage our proposed ECU in an enhanced version of a prior secure and dependable automotive approach (i.e., BD: Munir et al. [3]). The BD considers CAN as the in-vehicle communication protocol whereas our enhanced approach (OBD) considers CAN FD, which is more amenable for safety-critical automotive CPS applications.

Both of the approaches leverage encryption and message authentication codes (MACs) to integrate confidentiality, integrity, and authentication in the design. However, our proposed approach OBD is "encrypt-and-MAC" as compared to "MAC-then-encrypt" approach of BD. Encrypt-and-MAC provides comparable security to MAC-then-encrypt, however, encrypt-and-MAC has lesser computation and computation overhead than MAC-then-encrypt. The BD [3] uses 128-bit advanced encryption standard (AES-128) for integrating confidentiality and SHA-2/SHA-256 based hash-based message authentication code (HMAC) for integrating message integrity and authentication. Our approach reinforces the BD security by replacing SHA-2 based HMAC with SHA-3 based HMAC.

Our OBD approach requires only one HMAC computation and one AES-128 encryption of the original message as compared to three AES-128 encryptions and one HMAC computation in the BD. In our "encrypt-and-MAC" approach, the HMAC of message is not encrypted because the HMAC is collision resistant, and is computed with a secure secret key that is only known to legitimate sender and receiver nodes. The output of the HMAC computation (256-bit *message digest*) and the AES-128 encryption (128-bit *ciphertext*) are concatenated to generate a 384-bit (48-bytes) message. The sending nodes then transmits the 384-bit concatenated message to the receiver node via the CAN FD bus. The payload size of the CAN FD bus is 64 bytes so it is able to transfer 48 bytes of the message in one cycle. Hence, our enhanced approach saves two AES-128 computations at the sender node and speeds up CAN data transfer by $6\times$.

The BD and OBD uses two types of FT: FT by redundant multi-threading (FT-RMT) and FT by redundant multi-threading for quick error detection (FT-RMT-QED). Both the FT approaches leverage RMT on a dual-core architecture. RMT uses two threads: a master thread and a normal thread, to execute the same safety-critical computation. At the end of the computation, the results obtained from two threads are compared to detect an error in the calculation. If there is an error then recomputation is carried out on both the threads. Most of the time recomputation rectifies errors because soft errors, which are caused by transient faults, constitute a majority of errors in the computation. FT-RMT-QED enhances FT-RMT with QED [9] by inserting check instructions at different points in the master thread. When an error is detected earlier by FT-RMT-QED rather than at the end of the computation,

the erroneous computation is aborted, and the computation is restarted to obtain an error-free result [3]. These FT techniques can tolerate one permanent fault and multiple soft errors (by recomputations), and therefore adheres to the requirement of ISO 26262 standard [1].

B. Security and Dependability Standards for Contemporary Automotive ECUs

Contemporary automobiles leverage variety of microcontroller/microprocessor units (MCUs/MPUs) as ECUs depending on the automotive functions to be implemented on the ECUs. The security features in these MCUs/MPUs are based on one of the following standards: Secure Hardware Extension (SHE) [10]; hardware security modules (HSMs) from EVITA [11]; and Trusted Platform Module (TPM) [12] from the trusted computing group (TCG). TPM is neither designed nor applicable for automotive security as the TPM lacks cost efficiency, performance, robustness, and/or enough security functionalities. However, SHE and HSM are now widely accepted and used automotive standards. Although SHE and HSM provides all relevant security functionalities, the internal architecture of SHE and HSM based modules is not FT or dependable. If there is an error in the operation of these SHE and HSM based modules, then the whole automotive subsystem is prone to failure because these security modules influence the internal bus transactions and other activities of on-board electronic components. ISO 26262 is state-of-the-art automotive dependability standard. However, current automotive ECUs do not integrate security and dependability simultaneously.

C. Architecture Overview

To overcome the security and dependability limitations of existing automotive ECUs, we propose a novel ECU architecture that provides a more robust and flexible security and dependability solution. Fig. 1 depicts the overview of our proposed ECU architecture. This architecture is inspired by Xilinx Zynq-7000 All Programmable System-on-Chip. Our proposed ECU architecture has two programmable parts: an ARM based application processor or real-time processor (AP/RTP) and a field-programmable gate array (FPGA)-based programmable logic fabric (PLF). The AP/RTP implements the automotive function's (e.g., SBW) control algorithms. The type of AP/RTP used depends on the automotive function and the corresponding control algorithm to be executed. The FPGA-based PLF implements cryptographic functionality (i.e., cryptographic algorithms and protocols) for secure on-board communication and authentication. The AP/RTP and the PLF are connected by a high speed communication interface.

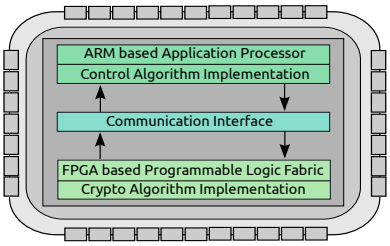

Fig. 1: High level architecture of proposed ECU.

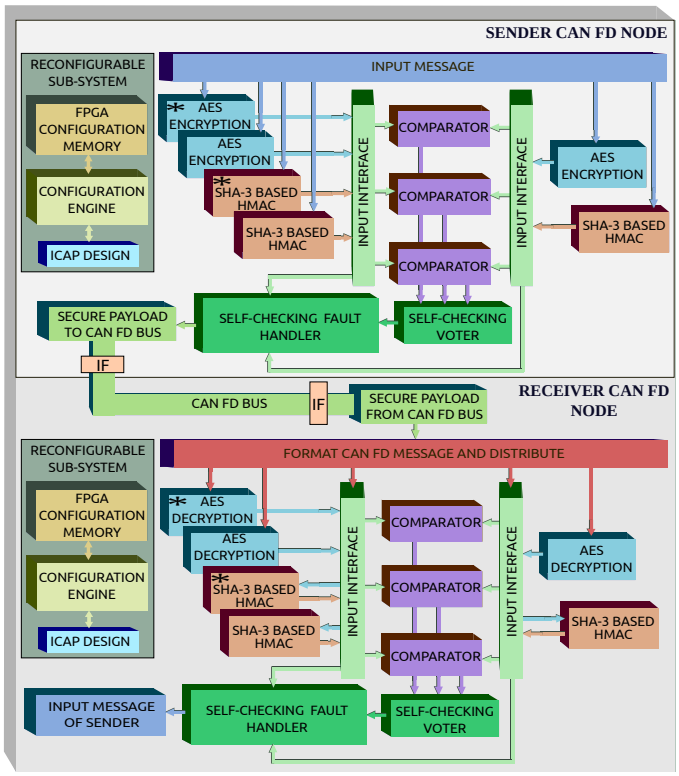

Fig. 2: Internal architecture of the secure and dependable cryptographic module (FT CM) implemented in PLF of the proposed ECU.

Our proposed ECU architecture provides a multitude of benefits. First, the PLF implements an FT cryptographic module (CM) as shown in Fig. 2. The FT is realized by using dynamic partial self-reconfiguration. Second, the PLF is a suitable platform for implementing real-time constrained compute-intensive algorithms, such as audio, video, and image processing algorithms. The PLF can be extended to implement these compute-intensive algorithms without incurring extra cost. To add new applications on top of the CM, the designer simply needs to write a hardware descriptive language (HDL) module for the new application, and then embed this module with the CM by modifying the bitstream in the PLF. Third, the internal communication interface between the ECU's AP/RTP and PLF is inherently secure since the AP/RTP and the PLF are connected by the internal bus, which is not accessible to an attacker.

D. Dependability

The automotive systems dependability requirements stipulated by ISO-26262 [1] can be met by designing an FT CM. The main dependability features of the FT CM in our design (Fig. 2) include: (1) dual modular redundancy (DMR) with an extra spare module (marked by * in Fig. 2); (2) Berger code based totally self-checking combinational circuit (TSC) [13]; and (3) partial reconfiguration feature of Xilinx Automotive (XA) Spartan-6 FPGA [14].

At the sender node, during non-faulty state, the input message is read by two AES encryption and two SHA-3 based

HMAC modules depicted on the left and the right part in Fig. 2. The spare modules (*) do not read the input message because the spare modules are disabled by the self-checking fault handler (SCFH) module. The outputs of the two AES encryption and the two SHA-3 based HMAC modules are fed to two input interfaces, which route these outputs to the comparators in triple modular redundancy (TMR). The comparators generate the comparison outputs and provide them to the self-checking voter (SCV). The SCV is a majority voter, which is designed based on Berger code based TSC. In case of a fault, the SCV can flag itself as a faulty unit to the SCFH. The output of the SCV is then passed to the SCFH. The SCFH generates all the necessary control signals for the operation of the FT CM (omitted in figure for clarity). Additionally, SCFH has a small buffer that stores the results of recent computations of all the modules in the FT CM.

If there is no error in the AES and HMAC computations, then the SCFH concatenates the HMAC to the AES encrypted message and sends the concatenated amalgam to the AP/RTP via a communication interface (IF), which then transmits the message to the receiver ECU via CAN FD bus. If there is a fault in AES and/or HMAC computation, then first recomputation is done to expunge any soft error that might have caused the fault. If recomputation fails to rectify the fault, both of the suspected faulty modules are deactivated, and the spare module is activated because it is not possible with DMR to discern which of the two suspected faulty modules is actually faulty. The spare module recomputes with the previous input and the result is routed to the SCFH via the input interface. The SCFH compares this new output with the output in buffer from the previous computation to identify the faulty module. Finally, the SCFH signals the reconfiguration subsystem to partially reconfigure the faulty module. If both of the modules in the DMR configuration are faulty, then both modules are reconfigured while the system continues operation with the spare module until the reconfiguration is completed. However, if only one of the modules is faulty in the original DMR configuration, then the non-faulty and the spare module start operating in DMR. The rationale for using the spare module in the new DMR configuration is that the reconfiguration takes longer time (in order of tens of millisecond) and the CM must be functional during that reconfiguration period for fulfilling the security and dependability requirements of automotive CPS.

At the receiver node, first, the CAN FD message is formatted to separate the AES ciphertext and the HMAC. Then, the ciphertext is decrypted by the AES decryption module to generate the original message. The original message is sent to the HMAC computing module and to the comparators via the input interface. The three comparators independently compare the AES decryption results and the outputs of the comparators are fed to the SCV, which then informs its decision to the SCFH. During the operation of the comparators and the SCV, the SHA-3 based HMAC module generates the local HMAC in parallel. Since the comparators and the SCV are faster than the SHA-3 based HMAC module by orders of magnitude, correctness check for AES decryption, and local HMAC calculation can be done in parallel without any conflict. However, this parallel operation incurs some additional signaling overhead on the SCFH. If there is no error in the AES decryption, then correctness of local HMAC is assessed after the local HMAC calculation. The FT operation to heal the faulty module is similar at both the sender and the receiver nodes.

Our proposed ECU architecture heals the faulty modules by exploiting partial reconfiguration (PR) technology as discussed in the following. Reconfigurability means the capability of programmable hardware devices, such as FPGA, to change customized design by loading different bitstreams. A more advanced reconfiguration technology is PR where a subset of FPGA operational logic is modified by downloading a partial configuration file/bitstream. Typically, PR is achieved by overwriting the current design in the FPGA configuration memory with the partial bitstream of the new design. Xilinx FPGAs provide a dedicated internal configuration access port (ICAP) that directly interfaces to the configuration memory and accesses it. We use LogiCORE IP XPS HWICAP [15] to perform dynamic partial reconfiguration. The XPS HWICAP IP enables MicroBlaze processor as configuration engine to read and write the FPGA configuration memory through the ICAP at run time and perform the partial reconfiguration.

E. Security

The CM provides three security services: message confidentiality, message integrity, and ECU authentication. We leverage AES-128 (128-bit) encryption to provide confidentiality and SHA-3 based HMAC for authentication and message integrity. Our approach assumes that initial AES and HMAC keys are stored in secure tamper resistant memories of ECUs by original equipment manufacturers (OEMs). Moreover, these keys are refreshed deterministically over time by the participating ECUs. Our security approach is resilient to passive and active eavesdropping, traffic analysis, and message injections. This is possible because there are no known brute-force and analytical attacks against AES and SHA-3 based HMAC computations.

IV. Steer-by-Wire System

A SBW system is one in which the heavy mechanical steering column is replaced by an electronic system, which reduces the vehicle weight and eliminates the risk of steering column entering into the cockpit in the event of a car crash. However, these benefits come with the stringent real-time performance and reliability requirements for SBW system. This section elaborates our SBW system that leverages multicore ECUs to incorporate security and dependability primitives.

A. Steer-by-wire Operational Architecture

In order to successfully replace the conventional steering column, SBW system needs to provide two main services: front axle control (FAC) and hand-wheel (HW) force feedback (HWF). The SBW architecture used in our study is depicted in Fig. 3. For our study, we focus only on the front axle control

part to compute the response time and error resilience of our FT approaches (refer Section III-A). Redundancy at ECU-, sensor-, and actuator-level provides FT in the SBW system. The sensors are connected to ECUs via point-to-point link while ECU-to-ECU communication is accomplished via CAN FD bus. The operation of our SBW system is similar as in previous work [3]. However, our SBW system leverages our proposed ECU architecture and the ECU-to-ECU communication is carried out via CAN FD bus instead of CAN bus to enable high speed data transfer.

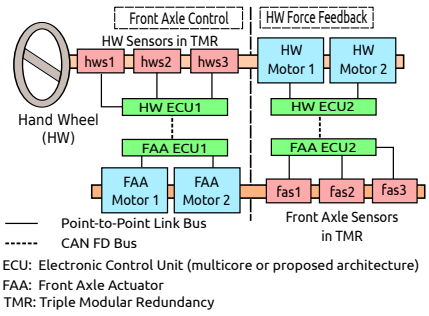

Fig. 3: SBW operational architecture.

B. QoS and Behavioral Reliability

The delay between the driver's request at HW and the response at FAA has significant impact on the reliability of SBW system. This end-to-end delay/response time ($\mathcal{T}_{res}$) is regarded as QoS that impacts safety and availability if it exceeds a critical threshold value, $\mathcal{T}_{max}$. This $\mathcal{T}_{max}$ is determined by automotive OEMs. The probability that the worst-case response time is less than the critical threshold is termed as behavioural reliability. The response time, $\mathcal{T}_{res}$, is subject to variation because of various vehicular operational aspects. The impact of this variation on vehicle's performance and stability is measured in terms of QoS score, S. In fact, there exist a linear relationship between S and $\mathcal{T}_{res}$ for instantaneous rotation of HW. According to Wilwert et al. [2], with a minimum tolerable S of 11.13, the critical limit $\mathcal{T}_{max}$ for the response time is 11.5 ms, beyond which the vehicle becomes unstable and the safety of driver can be at risk.

In the following, we analytically model the response time and the error resilience provided by our FT approaches (refer Section III-A) for the SBW system subject to implicit timing constraints imposed by behavioural reliability. We consider the FAA part of our SBW for our analytical modeling. The end-to-end delay/response time is modeled as the sum of pure delay ($\mathcal{D}_p$), mechatronic delay ($\mathcal{D}_{mech}$), and sensing delay ($\mathcal{D}_{sens}$), as $\mathcal{T}_{res} = \mathcal{D}_p + \mathcal{D}_{mech} + \mathcal{D}_{sens}$. Further, the pure delay comprises of ECUs' computational delay for processing the control algorithm and transmission delay including bus arbitration. For our secure and dependable architecture, pure delay also includes computation delay of our security and dependability primitives processing. Since $\mathcal{D}_{mech}$ and $\mathcal{D}_{sens}$ can be upper bounded by a constant value (3.5 ms [16]), we focus on pure delay for our reliability and error resilience analysis. The pure delay has the critical limit ($\mathcal{D}_p^{max}$) of 8 ms corresponding to $\mathcal{T}_{max}$ of 11.5 ms. The behavioural reliability can be modeled as $\mathcal{P}_{BR} = \mathcal{P}[\mathcal{D}_p^{wc} < \mathcal{D}_p^{max}]$, where $\mathcal{D}_p^{wc}$ is the worst case $\mathcal{D}_p$ and $\mathcal{P}_{BR}$ is the behavioural reliability. Finally, pure delay function for our FAC function can be written as, $\mathcal{D}_p^{FAA} = \mathcal{D}_{hw}^{ecu1} + \mathcal{D}_{canfd}^{channel} + \mathcal{D}_{faa}^{ecu1}$, where $\mathcal{D}_{hw}^{ecu1}$ and $\mathcal{D}_{faa}^{ecu1}$ denote the computation time at HW-ECU1 and FAA-ECU1, respectively. $\mathcal{D}_{canfd}^{channel}$ represents the time that CAN FD bus takes to transport message from HW-ECU1 to FAA-ECU2. Worst-case $\mathcal{D}_p^{wc}$ can be modeled as

$$(\mathcal{D}_p^{wc} = rcc1 \cdot \mathcal{D}_{hw}^{ecu1} + rtc \cdot \mathcal{D}_{canfd}^{channel} + rcc2 \cdot \mathcal{D}_{faa}^{ecu1}) \leq \mathcal{D}_p^{max}, \quad (1)$$

$\forall\ rcc1, rcc2, rtc \in \mathcal{Z}^+$, where $rcc1$ and $rcc2$ represent the number of recomputations that need to be done at HW-ECU1 and FAA-ECU1, respectively, and rtc represents the number of retransmissions that needs to be done for error-free sending of secure message over CAN FD bus. Since the two ECUs, HW-ECU1 and FAA-ECU1 are placed at different parts of the vehicle, they suffer from different types of hazards. So, we have two counters, $rcc1$ and $rcc2$, to count the number of recomputations that needs to be done at the two ECUs. Eq. 1 helps to analyse the number of recomputations and retransmissions that are allowed without violating the QoS and behavioural reliability requirement.

V. Results

In this section, we present our experimental setup and evaluation results comprising of timing analysis, energy analysis, QoS and behavioral reliability, and feasibility analysis.

A. Experimental Setup

Multicore-based ECU Design Implementation: We implement both the BD and the OBD (Section III-A) on *NXP quad-core iMX6Q SABRE* development board which has *Cortex-A9* CPU core. The 32-bit *Cortex-A9* processor runs *Ubuntu 14.04.4 LTS* at 396 MHz clock speed. The security primitives are coded in C. *OpenMP* is used to provide RMT-based FT on the multicore architecture. The OBD exploits efficient coding strategies for 32-bit ARM platform, such as loop-unrolling and cache-aware programming to attain better performance.

Proposed ECU Design Implementation: Our proposed ECU architecture has both the AP/RTP and the PLF. We have implemented the security primitives in the PLF, which is realized by *automotive grade Spartan-6 FPGA*. Our overall secure and dependable system/approach (Fig. 2) as well as each of the individual security modules are coded in *Verilog HDL* in *Xilinx ISE 14.7* and the functional verification is done using *ModelSim*. Further, *post-place and route simulation model* is generated using *ISE 14.7*. We have run the simulation model in *ModelSim* to get the execution time. We have used this procedure to get an accurate estimation of real-world execution time on the board. The total power consumption (both static and dynamic) is obtained by *XPower Analyzer* that comes with the *Xilinx ISE 14.7*. We use the power and execution time values to compute the energy consumed by the FT security module.

Vector CANoe based Setup: We simulate our SBW system in *Vector CANoe 8.5* [17] with CAN FD bus set to 48-byte payload, 1 $Mbps$ arbitration-phase baud rate and 3 $Mbps$ data-phase baud rate. We use CAPL (CAN Access Programming Language) to implement the SBW functions on ECUs.

TABLE I: Performance and energy results for BD, OBD, and EAF.

CAN FD Node	Operational Mode	Baseline Design (BD)			Optimized Baseline Design Implementation (OBD)			FPGA Implementation (EAF)		
		FT Mode	Time (μs)	Energy (μJ)	FT Mode	Time (μs)	Energy (μJ)	FT Mode	Time (μs)	Energy (μJ)
Sender Node	NFT	x	257	13.137	x	189	9.661	x	4.90	2.170
	FT	FT-RMT	411	21.010	FT-RMT	207	10.581	FT-SR-DMR	6.53	6.647
		FT-RMT-QED	589	30.109	FT-RMT-QED	313	16.000			
Receiver Node	NFT	x	235	12.013	x	184	9.406	x	9.00	3.996
	FT	FT-RMT	401	20.499	FT-RMT	203	10.377	FT-SR-DMR	10.63	10.831
		FT-RMT-QED	497	25.406	FT-RMT-QED	297	15.182			

Operational Parameters: For our SBW system, we assume the steering wheel sensor sampling rate to be fixed at $420\ Hz$, i.e. $\mathcal{T}_{sens} = 2.38\ ms$. For multicore based SBW, the ECU operates at $396\ MHz$ clock. The operational current is calculated as $36\ mA$ and the operational voltage as $1.42\ V$. For our proposed ECU architecture, the PLF, *Spartan-6-XA FPGA*, operates in $50\ MHz$ clock while running the CM.

B. Evaluation Results

Timing Analysis: The timing analysis for the FT approaches is done by injecting soft errors at different points in the security module code for both BD and OBD. Table I shows the timing performance of BD, OBD, and EAF. The results show that for NFT operational mode, OBD is 1.36× and 1.27× times faster than BD at the sender and receiver nodes, respectively. Results reveal that for FT-RMT, OBD has a speedup of 1.98× and 1.97× over BD at sender and receiver nodes, respectively. Finally, for FT-RMT-QED mode, OBD attains a speedup of 1.88× and 1.67× over BD at sender and receiver nodes, respectively.

Comparison of BD and EAF reveals that NFT EAF is 52.45× and 26.11× times faster than NFT BD at the sender and receiver nodes, respectively. Furthermore, after embedding FT in BD by FT-RMT and in FPGA by FT-SR-DMR (fault tolerance using self-reconfiguration in DMR), EAF is 62.94× and 37.72× times superior than BD at the sender and receiver nodes, respectively. Lastly, EAF with FT-SR-DMR provides a speedup of 90.19× and 46.75× at sender and receiver nodes, respectively, over BD in FT-RMT-QED mode.

Comparison of EAF and OBD shows that NFT EAF is faster than NFT OBD by 38.57× and 20.44× times at sender and receiver nodes, respectively. Moreover, FT-SR-DMR in EAF surpasses FT-RMT in OBD by 31.69× and 19.1× times at sender and receiver nodes, respectively. Furthermore, a speedup of 47.93× and 27.94× at sender and receiver nodes, respectively, is achieved with FT-SR-DMR in EAF over OBD with FT-RMT-QED.

Energy Analysis: Table I and Fig. 4 depict the energy consumption of our implementations. Results reveal that NFT OBD consumes 1.35× and 1.27× times lesser energy than NFT BD at the sender and receiver nodes, respectively. The FT-RMT based OBD is 1.98× more energy efficient than BD where as FT-RMT-QED based OBD is 1.88× more energy efficient than BD at the sender node. At the receive node, OBD consumes 1.97× and 1.67× times lesser energy than BD for FT-RMT and FT-RMT-QED, respectively. These energy savings stem from the modification of the security architecture and optimization of the security primitives codes for 32-bit ARM platform.

The comparison between EAF and BD reveals that NFT EAF is 6.05× and 3× times more energy efficient than NFT BD at sender and receiver nodes, respectively. At the sender node, EAF with FT-SR-DMR is 3.16× more energy efficient than BD with FT-RMT and 4.52× more energy efficient than BD with FT-RMT-QED. Similarly, at the receiver node, EAF with FT-SR-DMR is 1.89× more energy efficient than BD with FT-RMT, and 2.34× more energy efficient than BD with FT-RMT-QED.

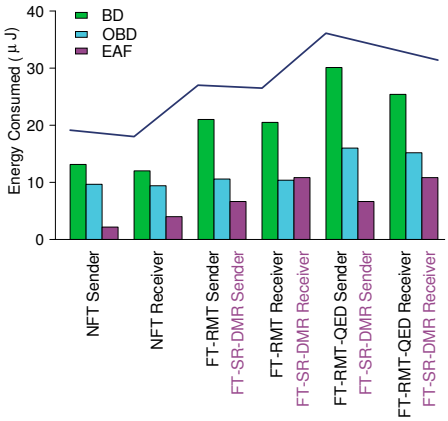

Fig. 4: Energy consumption of BD, OBD, and EAF.

The comparison between EAF and OBD divulges that NFT EAF results in 4.45× and 2.35× times more energy savings than NFT OBD at sender and receiver nodes, respectively. Additionally, at the sender node, EAF with FT-SR-DMR engenders 1.59× more energy savings than OBD with FT-RMT, and 2.4× times more energy savings than OBD with FT-RMT-QED, respectively. At the receiver node, EAF with FT-SR-DMR consumes 1.04× times more energy than OBD with FT-RMT. This is because of the fact that FT-SR-DMR uses redundant modules for FT, which increases the static power consumption of EAF.

QoS and Behavioral Reliability: We conduct experiments to determine the maximum number of allowable recomputations at SBW ECUs to yield error-free results subject to the critical pure delay $\mathcal{D}_p^{max} = 8\ ms$ and $\mathcal{D}_{canfd}^{channel} = 0.118\ ms$. The channel delay is obtained from the Vector CANoe simulations. The number of allowable recomputations represents the number of faults (soft errors in this context) the ECU can tolerate. Table II, which is obtained using Eq. 1, depicts $rcc1$, $rcc2$, and rct to yield correct result for the FAC function during faults.

TABLE II: The maximum number of allowed recomputations on ECUs and retransmissions on CAN FD bus to yield correct result for the FAC fuction during faults.

Baseline Design Implementation				
rtc	rcc1 with rcc2 = 1		rcc2 with rcc1 = 1	
	FT-RMT	FT-RMT-QED	FT-RMT	FT-RMT-QED
1	18	12	18	14
2	17	12	18	14
5	17	11	17	13
10	15	10	15	12
Optimized Baseline Design Implementation				
rtc	rcc1 with rcc2 = 1		rcc2 with rcc1 = 1	
	FT-RMT	FT-RMT-QED	FT-RMT	FT-RMT-QED
1	37	24	37	25
2	36	23	37	25
5	34	22	35	23
10	31	20	32	21

We have calculated rct because CAN and CAN FD buses do not provide FT and it is imperative that ECUs must be able to retransmit the message in case of transmission errors on the bus for correct and safe operation of the SBW system. We note that the number of faults tolerated at HW-ECU1 and FAA-ECU1 are given by $(rcc1 - 1)$ and $(rcc2 - 1)$, respectively, since error-free computations at HW-ECU1 and FAA-ECU1 still require one computational run-time for the FAA function. Table II presents the results for both BD and OBD. Results indicate that OBD can tolerate up to $(rcc1-1)+(rcc2-1) = 71$ faults with one transmission error without violating the $\mathcal{D}_p^{max}$ critical limit for FT-RMT technique. Comparison between BD and OBD reveals that OBD can tolerate 113% (2.13×) and 94% (1.94×) more faults on average than BD for FT-RMT and FT-RMT-QED, respectively. The table also shows the number of allowable retransmissions under the $\mathcal{D}_p^{max} = 8\ ms$ constraint. Results indicate that OBD permits 16.19× and 29.81× more retransmissions on average, respectively for FT-RMT and FT-RMT-QED operation mode, than BD for the same critical pure delay because of the lesser time in the ECU computations.

Feasibility Analysis: For our secure and dependable approach (Section III-A) leveraging our proposed ECU design, the pure delay and worst-case pure delay are calculated to be $0.13516\ ms$ and $0.15232\ ms$ respectively. These response times are well within $\mathcal{D}_p^{max} = 8\ ms$ of the critical threshold limit. Furthermore, our secure and FT ECU design permits much more time for control processing of the implemented automotive function (e.g., SBW) as compared to the multicore-based implementations. Moreover, our secure and FT ECU design is self-healing (i.e., recover from faults autonomously), and is able to meet the critical delay even in the presence of faults. These results verify the feasibility of our proposed secure and dependable approach and ECU architecture for automotive CPS.

VI. Conclusions

To better meet automotive security and dependability requirements while adhering to stringent real-time constraints of automotive CPS, we have proposed a novel architecture for automotive ECUs. We have implemented our proposed ECU architecture on Xilinx Automotive (XA) Spartan-6 FPGA. We also propose a secure and dependable approach for cybercar design that leverages our proposed ECU architecture. We demonstrate the effectiveness of our proposed architecture and approach using a SBW application over CAN FD as a case study. We further optimize and implement a prior secure and dependable automotive work on the NXP quad-core iMX6Q SABRE automotive board. Results reveal that our optimized design can tolerate 113% (2.13×) more faults on average than the prior work (baseline design). Results also divulge that our proposed ECU architecture can attain a speedup of 90.19× while consuming 4.52× lesser energy over baseline design. Furthermore, our proposed architecture can attain a speedup of 47.93× while consuming 2.4× lesser energy than optimized SABRE board implementation.

Acknowledgments

This work was supported by the National Science Foundation (NSF) (NSF-CRII-CPS-1564801). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] ISO26262. (2016, July) Road vehicles – functional safety. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43464
[2] C. Wilwert, Y. Song, F. Simonot-Lion, Loria-Trio, and T. Clement, "Evaluating quality of service and behavioral reliability of steer-by-wire systems," in *IEEE ETFA*, Lisbon, Portugal, Sep 2003, pp. 193–200.
[3] A. Munir and F. Koushanfar, "Design and performance analysis of secure and dependable cybercars: A steer-by-wire case study," in *IEEE CCNC*, Las Vegas, Nevada, USA, Jan 2016, pp. 1066–1073.
[4] K. Koscher et al., "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy*, Berkeley, California, USA, May 2010, pp. 447–462.
[5] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," in *ICCS*, Washington, DC, USA, Dec 2012, pp. 1–7.
[6] M. Wolf and T. Gendrullis, "Design, implementation, and evaluation of a vehicular hardware security module," in *ICISC*, Seoul, Korea, Nov-Dec 2012, pp. 302–318.
[7] E. Beckschulze et al., "Fault Handling Approaches on Dual-Core Microcontrollers in Safety-Critical Automotive Applications," in *IEEE ISoLA*, Porto Sani, Greece, October 2008.
[8] M. Baleani et al., "Fault-Tolerant Platforms for Automotive Safety-Critical Applications," in *ACM CASES*, San Jose, California, October-November 2003.
[9] T. Hong et al., "QED: Quick Error Detection Tests for Effective Post-Silicon Validation," in *IEEE ITC*, Austin, Texas, November 2010.
[10] Fujitsu. (2012, Feb) Secure hardware extension. [Online]. Available: https://www.escrypt.com/fileadmin/escrypt/pdf/WEB_Secure_Hardware_Extension_Wiewesiek.pdf
[11] F. SIT. (2008) E-safety vehicle intrusion protected applications. [Online]. Available: http://www.evita-project.org/
[12] T. C. Group. (2016) Trusted computing group - open standards for security and technology. [Online]. Available: http://www.trustedcomputinggroup.org/
[13] S. Kundu and S. M. Reddy, "Embedded totally self-checking checkers: A practical design," *IEEE Design Test of Computers*, vol. 7, no. 4, pp. 5–12, Aug 1990.
[14] *Spartan-6 FPGA Configuration User Guide (v2.8)*, Nov 2015.
[15] *LogiCORE IP XPS HWICAP v5.01a Product Specification*, Jun 2011.
[16] K. Klobedanz, C. Kuznik, A. Thuy, and M. Wolfgang, "Timing modeling and analysis for autosar-based software development - a case study," in *IEEE/ACM DATE*, Dresden, Germany, Mar 2010, pp. 642–645.
[17] I. Vector CANtech. (2016, Mar) CANoe ECU development and test. [Online]. Available: https://vector.com/vi_canoe_en.html