Asian Journal of Applied Science and Technology (AJAST)
(Open Access Quarterly International Journal) Volume 2, Issue 3, Pages 47-58, July-September 2018

ARM Based MPSoC ECU Architecture for Secure, Reliable and Real-time Self-Propelled CPS

V.Karthikeyan[1], B.Harshitha[2], P.Hemasruthi[3], D.Gowselya[4], R.Keerthana[5], N.Indhu[6]

[1]*Assistant Professor, Department of Electronics and Communication Engineering, Vivekanandha College of Engineering for Women, India.*
[2, 3, 4, 5, 6]*UG Scholars, Department of Electronics and Communication Engineering, Vivekanandha College of Engineering for Women, India*

Article Received: 19 April 2018 *Article Accepted: 28 June 2018* *Article Published: 31 July 2018*

ABSTRACT

In this paper, Electronic Control Units architectures follow the multiprocessor system on-chip (MPSoC) design paradigm wherein the ECUs have multiple heterogeneous processing engines with specific functionalities. The first architecture, General, leverages an ARM-based application processor and a General Purpose Graphic Process Unit-based co-processor. The second architecture, RED, integrates an ARM based application processor with a Field Programmable Gate Array-based co-processor. We quantify and compare temporal performance, energy, and error resilience of our proposed architectures for a steer-by-wire case study over CAN, CAN FD, and Flex Ray in-vehicle networks. Here implement two Electronic Control Unit (ECU) architectures for real-time automotive cyber-physical systems that incorporate security and dependability primitives with low resources and energy overhead.

Index Terms—Automotive, cyber-physical systems, ARM, steer-by-wire, security, dependability.

I. INTRODUCTION

Contemporary automobiles integrate a multitude of heterogeneous digital processors (ECUs)), radio interfaces, in-vehicle networks and protocols, and hundreds of megabytes of complex embedded software. Next generation of automobiles (known as cyber cars) will further escalate the profusion of novel distributed control applications. Emergence of x-by-wire systems, where electronic controllers replace traditional mechanical / hydraulic subsystems, is a prominent example of recent modernization in the automotive industry. However, x-by-wire systems have stringent real-time performance and reliability requirements, which pose significant challenges for implementation over traditional, bandwidth limited CAN. Since CAN is the most prevalent protocol for in-vehicle communication and most of the car manufacturers are reluctant to adopt a completely new protocol, CAN with flexible data rate (CAN FD) is a viable replacement of CAN for x-by-wire applications. Flex Ray is another recent protocol that is well suited for x by-wire applications as the protocol offers high speed data transfer and fault tolerance features. As electronic components permeate into safety-critical automotive functions, integration of security and dependability in cyber car design becomes imperative. The continuously escalating complexity of automotive systems and increasing integration with wireless entities exacerbate the security vulnerabilities of cyber cars. Furthermore, harsh operational environment combined with external noise and radiation render ECUs vulnerable to permanent and transient faults. Hence, in order to make cyber cars robust to faults and security vulnerabilities, cyber cars must incorporate dependability and security features. When retrofitting the in-vehicle architectures with security and dependability mechanisms, a prime challenge is to ensure that hard real-time constraints of the automotive cyber-physical applications are not violated. The evolving nature of cyber-attacks presents another challenge in integrating security primitives in automotive cyber physical systems. *Advancements in cryptanalysis* Attack technologies might render various security mechanisms ineffective. Most of the global initiatives on future

automotive CPS focus on design of dedicated security solutions that could be embedded in future automotive ECUs. *Security solutions are based on one of the following standards*

1) Secure hardware extension

2) Hardware security Module

3) Trusted platform module, all of which are non fault-Tolerant and are dedicated inflexible hardware Designs.

Hence, a successful attack on the security mechanisms embodied in these standards would require a complete replacement of the ECUs leveraging these security standards, which would be very costly or in worst case, infeasible. Unfortunately, the problem is not only limited to non-fault tolerance and inflexibility of these security mechanism. The constant craving of automotive industry to accommodate new cyber car applications requires ECUs with high computational power. These novel cyber car applications may not be effectively handled by contemporary microcontroller-based ECUs. Our main contributions are:

PROPOSE AND IMPLEMENT

We two novel secure and fault-tolerant MPSoC based

ECU architectures: a general-purpose graphics processing unit (GPGPU)-based ECU design (GED) and a reconfigurable ECU design (RED), which are able to effectively meet security, dependability, and real-time requirements of automotive CPS in an energy-efficient manner. We our proposed GED architecture in NVIDIA's Jetson TK1 board, and our proposed RED architecture in Xilinx Automotive Spartan-6 field programmable gate array (FPGA) board. Furthermore, we consider a baseline ECU design (BED) architecture, which is implemented in NXP's automotive-grade iMX6Q SABRE board.

*MODEL AND COMPARE_* We a SBW subsystem and quantify and compare the temporal performance, energy, and error resilience of our proposed ECU architectures. We response times of a SBW subsystem leveraging our proposed ECU architectures over three in-vehicle Networks: CAN, CAN FD, and Flex Ray.

II. RELATED WORK

Various previous works have studied security of automotive systems. Koscher have examined multitude of internal and external attack surfaces of a modern automobile through which an adversary could infiltrate in-vehicle networks to control automotive subsystems while ignoring the driver's input. This work is followed by scores of studies on embedding security primitives in vehicle networks. A number of prior works including have examined integration of message authentication codes (MACs)in CAN data frames to secure in-vehicle data interaction. Furthermore, in order to defend against masquerade and replay Attacks, have proposed MACs with counters. Although these methods provides message authentication, these approaches do not provide confidentiality of in-vehicle data. That could provide security for in-vehicle communications. These studies only consider security aspects but fail to analyze the simultaneous integration of security and dependability in real time automotive CPS which is cardinal to safety and security of modern automobiles. The dependability for automotive embedded

systems has been explored by a number of earlier works. Beck schulzeet al. have investigated FT approaches based on dual core microcontrollers. Baleani et al. Have discussed various architectures for automotive applications, such as lock-step dual processor architecture, loosely-synchronized dual processor architecture, and triple modular redundant architecture. However, these works have not considered the interplay of Performance, dependability, and security for modern automotive CPS. Munir et al. is the first work that has proposed multicore ECU based design for secure and dependable cyber cars. Yet, this work has not implemented the proposed approach on an automotive-grade processor.

III. SECURE AND DEPENDABLE APPROACH FOR CYBER CAR DESIGN

A. Security Threat Model

With a large number of ECUs operating inside cyber cars, there are plenty of security attack surfaces that impact most of the in-vehicle systems. This situation is further exacerbated by the connection to increasingly wide range of external networks, from Wi-Fi, cellular networks, and the Internet to service garages, toll roads, drive-through windows, gas stations, and a rapidly growing list of automotive after-market applications. In order to elucidate the need for incorporating security primitives in ECUs, we discuss various methods in the following by which an adversary could penetrate the in vehicle networks (e.g., CAN, Flex Ray) to accomplish various security attacks.

Need for confidentiality:

In-vehicle networks cart a mix of operational and personally identifiable information, such as current location, previous destinations, navigation history, call history, microphone recordings, and financial transactions, etc. An adversary invading an in-vehicle network could perform passive eavesdropping and traffic analysis, thus, obtaining critical information about the driver and the vehicle. In addition, for the x-by-wire systems, if an adversary knows the initial Location of the vehicle, then, by eavesdropping on the steering angle, accelerator value, and braking value, the adversary could track the car which might put the driver and passenger sat risk. Hence, the confidentiality of messages and data over in-vehicle networks is critical for operational security, privacy, And consumer trust.

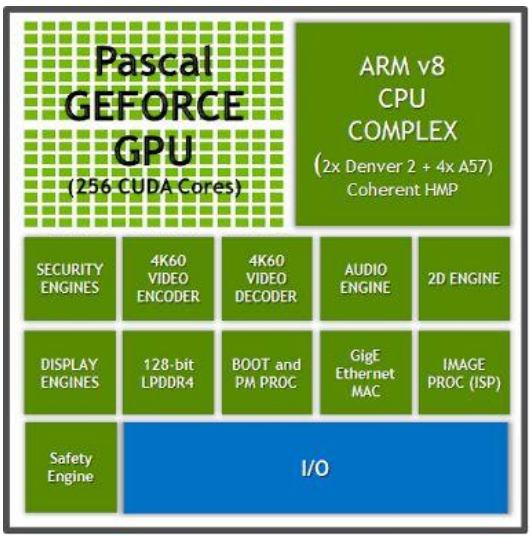

Fig. 1: BED Architecture.

Need for authentication and integrity:

An attacker invading an in-vehicle network may attach his/her own device or compromise a valid user device in order to send fraudulent requests into the system. Furthermore, the attacker's device may impersonate a valid ECU or gateway for malicious activities that may jeopardize safety of the driver and the vehicle. Additionally, the adversary may perform spoofing attacks by actively injecting and/or modifying messages in the in-vehicle network. Thus, entity authentication and message integrity verification are required in in-vehicle networks to defend against these vulnerabilities.

B. Baseline ECU Design Approach

We leverage our BED approach in an enhanced version of a prior secure and dependable automotive design [13]. The BED approach is implemented in NXP's iMX6Q SABRE automotive Development board. The BED represents the contemporary Microcontrollers-based design used in conventional automotive ECUs. The basic security and dependability features used in our proposed ECUs are incorporated in BED. Then, the GED and the RED architectures are compared with BED architecture to access the gains in performance and energy of our proposed designs.

Security:

Our BED approach leverages AES-based encryption and SHA-3-based HMAC to integrate confidentiality, integrity, and authentication. Fig. 1 shows our security approach used in sender and receiver Flex Ray nodes. The sender node uses "encrypt-and-MAC" security approach, where the 128-bit message (64-bit ECU message plus 64-bit counter) is encrypted in parallel with HMAC computation using separate 128-bisecret keys. We have used advanced encryption system (AES-128) to provide confidentiality, and secure hash algorithm-3. Based message authentication code to provide entity and message authentication. The basis of AES security is its robustness to brute force attacks as the key space of AES-128 is 3:4_1038 keys. Even at a sustained rate of 1 tera-keys/second, it would take 1019 years to exhaust this key space. SHA- 3 provides 128-bit security level for collision attacks and 256-bit security strength for pre-image and second pre-image attacks. Additionally, to strengthen the security primitives, the secret keys for AES and HMAC are stored in secure tamper resistant memory [9] and are refreshed deterministically over time by participating ECUs.

Dependability: The dependability requirements as stipulated in ISO 26262 [15] require that at least one critical fault must be tolerated by automotive applications without loss of functionality. The BED leverages FT using redundant multi-threading (FT-RMT) to provide dependability. Redundant multi-threading (RMT) uses two threads–a master thread and a slave thread, which execute same operations with the same data set. In FT-RMT, the results of master and slave threads' operations are compared at the end of computation by the master thread. If there is an error, then re-computation is carried out on both the threads. The rationale behind using re-computation after error is that re-computation rectifies soft errors caused by transient faults. FT-RMT can tolerate one permanent

fault and multiple soft errors, and therefore adheres to the dependability requirements specified in ISO 26262 standards.

IV. SECURE AND DEPENDABLE MPSoC BASED ECU ARCHITECTURES

Cyber cars integrate a multitude of microcomputer units as ECUs to implement different automotive functions. As discussed in Section I, the security features provided by various contemporary standards are often implemented in dedicated inflexible hardware and are not adaptive to evolving nature of security attacks. Additionally, future automotive ECUs may require high computational power to integrate newly emerging cyber car applications and services. To overcome these limitations in prior ECU designs, we propose flexible and scalable ECU architectures that simultaneously integrate security and dependability primitives with low resources and energy overhead.

A. Generalized Version

Fig. 2 shows the generalized internal architecture of the proposed MPSoC-based ECU. The ECU consists of an ARM based application processor as the main processor and one or more application-specific coprocessors. This application processor provides interface to the in-vehicle networks (e.g., CAN, CAN FD, Flex Ray, LIN, MOST, etc.), external sensors, other ECUs, and gateways. Furthermore, this application processor executes control algorithms, performs data aggregation from various sensors, and outsources computationally intensive applications to the application-specific co-processors. The application-specific coprocessor performs compute-intensive applications like image, audio, and video processing. The application-specific co-processor could be a digital signal processor (DSP), or crypto processor that carries out asymmetric and symmetric cryptographic operations, or Viterbi processor for the realization of maximum-likelihood decoding of convolution codes, etc. The application processor and co-processor communicates via advanced system bus (ASB) or advanced high Programmable AHB: Advanced High-performance Bus ASB: Advanced System Bus APB: Advanced Peripheral Bus performance bus (AHB) [2].

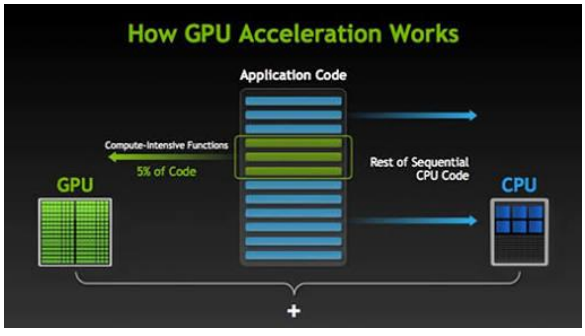

Fig. 2: A basic MPSoC-based ECU architecture.

The application processor of one ECU can communicate with the application processor of another ECU through in-vehicle networks During operation, the application processor collects data from the sensors. If this data is to be sent to another ECU, the application processor first sends the data to the coprocessor via high-speed AHB. The co-processor embeds the security primitives into the data thus creating a *secure data* and sends it back to the application processor. This is represented by path ”a” in Fig. 2. The *secure data* is send to another ECU via path

”b”. In our work, we have used two types of application-specific co-processor viz., GPGPU and programmable logic array. These processors are responsible to provide the cryptographic services, such as confidentiality, message authentication, and message integrity. Our work does not focus on the ECU authentication service. SRAM-based physical unclonable functions is one solution which can be deployed for ECU authentication. First design goal is to provide security services to the ECU. In order to provide security services to ECU, we implement the cryptographic algorithms in the co-processor. Second design goal is to provide dependability (or fault tolerance). We leverage FT based on redundant multi-threading (FT-RMT) for GED and dual modular redundancy based FT technique for RED. Finally, to assess whether our ECU performs its tasks without violating the real-time deadline of the tasks, we create a timing model of a SBW automotive subsystem that uses our ECU to provide steering functionality to the vehicle.

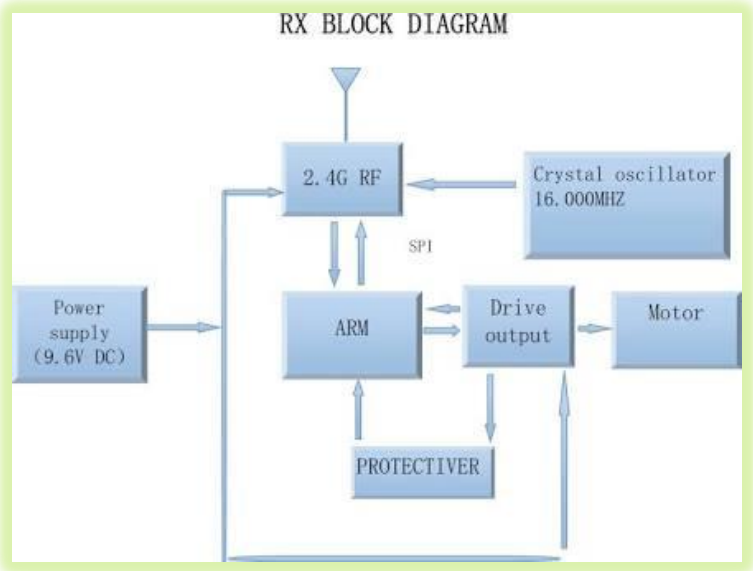

Fig. 3 RX block diagram

B. GPGPU based ECU Architecture

Fig. 3 depicts the detailed GED architecture which has the same high level architecture as in the generalized version discussed in Section IV-A. Here, an ARM-based application processor (AP) is the master processor and a GPGPU-based processor is the slave processor (or co-processor) that communicates with AP via a high speed internal bus like AHB. The AP provides interface to the in-vehicle networks, external sensors, other ECUs, and gateways. Furthermore, the AP executes control algorithms, performs data aggregation from various sensors, and outsource computationally intensive applications (e.g., image, audio and video processing, cryptographic operations, etc.) to the GPGPU-based co-processor. We implement FT symmetric cryptographic primitives in GPGPU-based coprocessor to provide security and dependability feature to the ECU. In addition, we have used this fault-tolerant (FT) cryptographic primitives implementation as an example use case for comparing the temporal performance and energy consumption of our proposed ECUs, viz., BED, GED, and RED.

Security:

The cryptographic module with AES-128 encryption and SHA-3-based HMAC are implemented in GPGPU based co-processor. The functional architecture of GED is identical to that of BED (refer Section III). However, the GED processes a batch of eight 128-bit messages at once unlike the BED which processes a single 128-bit message at a time. We adopt this batch processing mechanism to utilize the massive computational power of GPGPU and to enhance the throughput of the cryptographic module. Furthermore, the implementation in GPGPU exploits the largely byte-parallel operations of AES-128 and lane-parallel operations of Our implementation adopts parallel granularity of byte-per thread for AES-128. Each byte of AES-128 is mapped to a thread in GPGPU. A thread-block with number of threads equals to warp size of 32 is used to compute a complete AES-128 encryption/decryption. All threads in one warp are executed in a single instruction multiple data (SIMD) fashion. For SHA-3, parallel granularity of 8-byte (a *lane*) per thread is used.

A thread-block with 32 threads is used to compute complete SHA-3-based HMAC. Furthermore, frequently accessed S-boxes, round constants, and other index constants used in AES-128 and SHA-3 are stored on the on-chip shared memory of GPGPU to ensure fast memory access. At the sender Flex Ray node, eight plaintexts (ECU messages) are processed at once. For NFT mode of operation, the processing is carried out in GPGPU by launching 16 thread-blocks. Eight of these thread-blocks are used for AES- 128 computation and eight are used for HMAC computation. Each thread-block processes one ECU message. AES-128 and HMAC are computed in parallel thread-blocks because they are independent operations. However, at the receiver Flex Ray node, HMAC computation requires the output of AES computation. Hence, at first, eight thread-blocks are launched to compute AES-128 (decryption) to recover plaintexts of the eight cipher texts received from the sender node. Then, eight new thread-blocks are launched for HMAC computation of the recovered plaintexts.

Dependability: The design leverages FT-RMT to provide resilience against soft errors occurring due to transient faults. The FT-RMT is achieved by using Redundant thread-blocks. Each thread-block, and its redundant counterpart, performs a complete AES (or HMAC) computation. In the FT-RMT mode (Fig. 3), eight ECU messages are processed using 32 GPGPU thread-blocks: sixteen of these thread-blocks are used for AES-128 and sixteen are used for HMAC computation. In each group of 16 thread-locks, eight are master thread-blocks and eight are redundant thread-blocks. Each master and redundant thread-block pair processes one ECU message. The results obtained from the master and its redundant thread-block are compared to detect computational errors. The thread-blocks must be synchronized to compare the results. The synchronization is carried out by employing a CPU synchronization technique. If computational errors (soft errors) are detected, then recomputation is conducted.

C. Reconfigurable ECU Architecture

The ARM-based AP has same set of functionalities for both RED and GED. In RED, the AP outsources compute-intensive applications to an FPGA-based co-processor. *Security:* The cryptographic module is implemented in an FPGA-based co-processor and provides three security services: confidentiality, message

integrity, and authentication. The FPGA-based co-processor computes AES encryption and HMAC of the ECU message at the sender node. The coprocessor then relays the concatenation of the CT and the *message digest* to the AP which then transmits it to the receiver node via Flex Ray bus. The receiver AP then relays the received concatenation to its FPGA-based co-processor. The co-processor, first, decrypts the CT to recover the original plaintext and then computes HMAC of the plaintext to generate *local message digest*. The local message digest is compared with the received message digest to check the integrity of the received message. If the message has lost its integrity, then the message is retransmitted.

Dependability: The cryptographic module implemented in FPGA is resilient to multiple transient faults and one permanent fault. FT is provided by a combination of three methods: dual modular redundancy (DMR) with an extra spare module .Berger code based totally self-checking (TSC) combinatorial circuits, and dynamic partial re-configurability of Xilinx Automotive Spartan-6 FPGA. This FT mechanism is named as FT using self reconfiguration in dual modular redundant system (FT-SRDMR).DMR is used to detect errors in computation; TSC design method is employed to design a combinatorial circuit that flags itself as erroneous in case of faults; and dynamic partial reconfiguration is exploited to heal faulty modules.

At both sender and receiver nodes, AES and HMAC modules in DMR computes CT and message digest, respectively, from the ECU message. The SCCU generates the control signals for the cryptographic module and stores the result of recent AES and HMAC in a buffer memory. Whenever there is error in AES (or HMAC) computation, the SCCU activates the spare AES (or HMAC) module. The spare module(s) then computes the results corresponding to the input for which there was an error. The results of the spare module are then compared with the results of regular AES (or HMAC) module that were stored in buffer to localize the faulty module. The SCCU then activates the *reconfiguration subsystem* which performs partial reconfiguration of the faulty module. The cryptographic module operates with the spare module(s) during the reconfiguration period. The rationale for using the spare module during reconfiguration is that the reconfiguration takes longer time (in terms of tens of millisecond) and the cryptographic module must be functional during this period to fulfill the security and dependability requirements of automotive CPS.

V. MODELLING AND ANALYSIS OF A STEER-BY-WIRE SUBSYSTEM

A. Steer-by-Wire Operational Architecture

In an SBW subsystem, heavy mechanical steering column is substituted by electronic systems to reduce vehicle weight. This eliminates the risk of steering column entering into the cockpit in the event of a crash. The SBW subsystem provides the same functionalities as conventional steering column: front axle control (FAC) and hand-wheel (HW) force feedback. This paper focuses only on the Front Axle Control part to compute response time and error resilience of the FT approaches used in our proposed ECUs. Furthermore, the SBW subsystem is made FT by using redundant ECUs, sensors, and actuators. Point-to-point links connect ECUs to sensors and ECUs to actuators. For Electronic Control Unit to- Electronic Control Unit connection, we experiment on all three

commercial automotive buses: CAN, CAN FD, and Flex Ray. We evaluate and present a comparison of the SBW *response time* and *error resilience* when using these three buses.

B. Timing Model of SBW to Compute the QoS and Behavioral Reliability The *end-to-end delay*/*response time* (*_r*) is the delay between the driver's request at the HW and the corresponding response at the front axle actuator (FAA). *_r* is regarded as a QoS Metric but can also be interpreted as a dependability metric that impacts automotive safety and reliability if it exceeds a critical threshold value *_max r* , which is determined by automotive original equipment manufacturers (OEMs). Furthermore, the probability that the worst-case response time is less than the critical threshold is termed as *behavioral reliability*. In the following, we analytically model the response time for the SBW subsystem and error resilience of our FT approaches. *Response time* (*_r*) is modeled as the sum of pure delay (*_p*), mechatronic delay (*_m*), and sensing delay (*_s*), as, $_r = _p + _m + _s$. The sensing delay is the delay during the interaction of application processor of ECU with the sensors. The sensing and mechatronic delays are bounded by a constant value of 3:5*ms* .For our secure and dependable approach, the pure delay (*_p*) includes ECUs' computational delay for processing the control algorithm (depends on the execution time of application processor), computational delay for processing the incorporated security and dependability primitives (depends on the execution time of the co-processor), and transmission delay including bus arbitration (depends on the type of in-vehicle network used like CAN or CAN FD). Mathematically, pure delay (*_p*) for our FAC function can be written as, $_p = rcc1 \_ \_ecu1\ hw + rtc \_ \_bus + rcc2 \_ \_ecu1\ faa) \_ \_max\ p$; (1) where *_ecu*1 *hw* and *_ecu*1 *faa* denote the computation time at HWECU1 and FAA-ECU1, respectively; *_bus* represents the transmission time for a message on an in-vehicle bus (CAN, CAN FD, or Flex Ray) from HW-ECU1 to FAA-ECU1; *rcc*1 and *rcc*2 represent the number of recomputations that are needed to be done at HW-ECU1 and FAA-ECU1, respectively, to rectify soft errors; *rtc* represents the number of retransmissions required for an error-free transmission of a secure message over in-vehicle bus; and *_max p* represents maximum allowable *_p _ecu*1 *hw* and *_ecu*1 *faa* are calculated as the sum of execution time of application processor, execution time of co-processor, and bus time of AHB and APB .Since the co-processor is executing the computationally intensive cryptographic primitives, the execution time of the co-processor is far greater than the sum of bus times and execution time of application processor. Therefore, we use the execution time of the co-processor as the *_ecu*1 *hw* and *_ecu*1 *faa* .

VI. RESULTS

In this section, we present our experimental set up and evaluation results comparing timing analysis, energy analysis, response time, and QoS and behavioral reliability of the SBW subsystem with different in-vehicle buses, viz., CAN, CAN FD, and Flex Ray.

A. Experimental Setup

Baseline Design Implementation: We have implemented the BED on *NXP quad-core iMX6Q SABRE* development board which has a 32-bit *Cortex-A9* CPU core running *Ubuntu 14.04.4 LTS* at 396 MHz clock speed. The security and dependability primitives are coded in *C*. *OpenMP* is leveraged to provide FT-RMT. *GPGPU based ECU*

Implementation: We have implemented the security and dependability primitives of GED architecture on NVIDIA's Jetson TK1 GPGPU. The NVIDIA's Jetson TK1 GPGPU has *192 CUDA cores*, *1 streaming multi-processor*, CUDA capability of 3.2, and runs at 852 MHz clock speed. The application processor is ARM *Cortex-A15*. The FT cryptographic Modules are coded in CUDA 6.5 using C. *Reconfigurable ECU Implementation:* We have implemented the security and dependability primitives of RED architecture in *automotive grade Spartan-6* FPGA. The FT cryptographic modules are coded in *Verilog HDL* in *Xilinx ISE 14.7*.

We use *ModelSim* for the functional verification of the design. The total power consumption (both static and dynamic) is obtained through *XPower Analyzer* packaged with Xilinx ISE 14.7 suite. We use power and latency to compute the energy consumed by the FT cryptographic module. *Operational Parameters:* For our SBW subsystem, we assume the steering wheel sensor sampling rate to be fixed at 420 Hz, that is, $\_s = 2{:}38ms$. For the BED architecture, the ECU operates at 396 MHz with an operational current of *36 mA* and the operational voltage of *1.42 V*. For the GED architecture, the operating voltage is 0.87 V and the operating current is 54 mA. For the RED architecture, the ECU operates at 50 MHz.

B. Evaluation Results

Timing Analysis: In real-time automotive CPS, system response times must adhere to strict deadlines. Our evaluations demonstrate that the execution times of our proposed ECU architectures are in the range of microseconds, which conform to the real-time constraints of the SBW subsystem. Furthermore, results indicate that our proposed ECU architectures attain significant speedup as compared to a conventional ECU architecture (BED architecture).The comparison between NFT GED and NFT BED reveals that the NFT GED is 1:84_ faster than the NFT BED. Similarly, the FT GED is 1:79_ faster than the FT BED. This is because the GED exploits the massive computational power of GPGPU to perform cryptographic operations. However, the RED has better timing performance than the GED. The NFT RED and the FT RED attain a speedup of 15:83_ and 14:62_ over the NFT GED and the FT GED, respectively. Additionally, the comparison between RED and BED reveals that the NFT RED and the FT RED provide a speedup of 29:5_ and 26:4_ over the NFT BED and the FT BED, respectively. The RED is able to achieve this high speed up because of its high-performance parallel hardware architecture which is implemented in reconfigurable logic (FPGA). These speedup values are calculated as the average of speedups at the sender and the receiver ECUs. For each of the sender and the receiver ECUs, the measurements are averaged over 100 runs. *Energy Analysis:* Energy efficiency is an important metric for automotive CPS as it implies greater fuel-efficiency for combustion engine vehicles and longer battery life for hybrid and electric vehicles. All of our proposed ECU architectures are energy-efficient. Table I shows that the GED yields better energy efficiency than the BED. The NFT GED and the FT GED consumes 2_ and 1:95_ less energy than the NFT BED and the FT BED, respectively. This is because of the energy efficiency of the GPGPU architecture and less execution time for the GED than that of the BED. When comparing RED and BED, we observe that the RED offers 3:4_ and 1:5_ improvements in energy efficiency over the BED for NFT and FT operational modes, respectively. Results also indicate that the NFT RED is 1:68_ more energy-efficient than the NFT GED. This is because the execution time

for the NFT RED is 15:83_ lower than that of the NFT GED. However, the FT GED is 1:47_ Energy-efficient than the FT RED. The FT GED attains this energy savings because the FT technique (FT-SR-DMR) employed in FT RED has high static power consumption while the FT GED has significantly lower static power consumption. *QoS and Behavioral Reliability:* We conduct experiments to determine the impact of using different in-vehicle buses on the QoS and behavioral reliability of the SBW subsystem leveraging our proposed ECU architectures. Table II presents the end-to-end delay/response time of our SBW subsystem when using various in-vehicle buses in combination with different ECU architectures.

Results manifest that CAN FD and Flex Ray are better alternatives to conventional CAN bus as they provide higher bandwidths and lower latencies. In order to deliver the *48-bytes payload*, CAN FD takes 18:79_ less time than CAN bus. Flex Ray offers even less transport time, that is, the payload transport time of Flex Ray is 32:36_ lower than that of CAN bus and 1:5_ lower than CAN FD bus. Moreover, Flex Ray provides redundant communication channels which offers FT communication. When comparing the response time of the SBW subsystem, the system with ECUs integrating RED architecture yields 3:88_ and 2:53_ better response time than that of the system with ECUs leveraging BED and GED architectures, respectively. This is because the execution time of the FT cryptographic module in RED is much less than that of the BED and the GED architectures. In addition to bus latency and response time analysis, we orchestrate experiments to quantify maximum number of allowable recomputations at SBW ECUs to yield error-free results subject to the critical pure dealy $\_max\ p = 35ms$ and $\_bus$ (or bus latency) as shown in Table II. Based on the aforementioned $\_p$ constraint (refer Eq. 1 in Section V-B), we calculate $(rcc1 \square 1) + (rcc2 \square 1)$ with $rtc = 2$ which gives the number of tolerable faults at HW-ECU1 and FAAECU1. Results reveal that the ECUs with RED architecture can tolerate up to 3*;* 459 faults with one transmission error while the GED and the BED architectures can tolerate 231 and 127 faults, respectively, with one transmission error. It is apparent that the RED is more robust to soft errors since it can tolerate 27:23_ and 14:97_ more faults than the BED and the GED, respectively.

VII. CONCLUSION

In this paper, we have proposed and implemented two novel ECU architectures, viz., GED and RED, for real-time automotive CPS. The salient features of RED and GED are:

(1) Simultaneous integration of security and dependability primitives while adhering to stringent real-time constraints of automotive CPS;

(2) The ability to perform compute-intensive applications, such as cryptography and audio, video, graphics, and image processing, in an energy-efficient manner;

(3) The flexibility and scalability rendered by reprogram ability that enable upgrading the architectures to incorporate new applications in future; and (4) the resistance of the ECU against fault injection and analysis attacks. Furthermore, we have quantified and compared temporal performance, energy, and error resilience of our proposed ECU architectures for a SBW case study over CAN, CAN FD, and Flex Ray in-vehicle networks. Hardware implementation results reveal that RED and GED can attain a speedup of 31:7_ and 1:8_, respectively, while

consuming 1:75_ and 2_ less energy, respectively, than the contemporary ECU architectures. Furthermore, RED and GED can tolerate 27:23_ and 1:9_ more transient faults, respectively, than the contemporary ECU architectures.

REFERENCES

[1] M. Cortez, A. Dargar, S. Hamdioui, and G. J. Schrijen. Modeling sram start-up behavior for physical unclonable functions. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Albuquerque, New Mexico, Oct 2012.

[2] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici. Dynamic fault tolerance in FPGAs via partial reconfiguration. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 165–174, Napa Valley, California, Apr 2000.

[3] K. Klobedanz, C. Kuznik, A. Thuy, and W. Mueller. Timing modeling and analysis for autosar-based software development - a case study. In *2010 Design, Automation Test in Europe Conference Exhibition*, pages 642–645, Dresden, Germany, Mar 2010.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy*, pages 447–462, Berkeley, California, May 2010.

[5] C.-W. Lin and A. Sangiovanni-Vincentelli. Cyber-security for the controller area network (CAN) communication protocol. In *International Conference on Cyber Security (CyberSecurity)*, pages 1–7, Washington, DC, Dec 2012.

[6] A. Munir and F. Koushanfar. Design and performance analysis of secure and dependable cyber cars: A steer-by-wire case study. In *IEEE Annual Consumer Communications Networking Conference CCNC*, Las Vegas, Nevada, Jan 2016.

[7] D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *IEEE 68th Vehicular Technology Conference*, pages 1–5, Calgary, BC, Sep 2008.

[8] R. Soja. Automotive security: From standards to implementation. Technical report, Freescale, 2014.