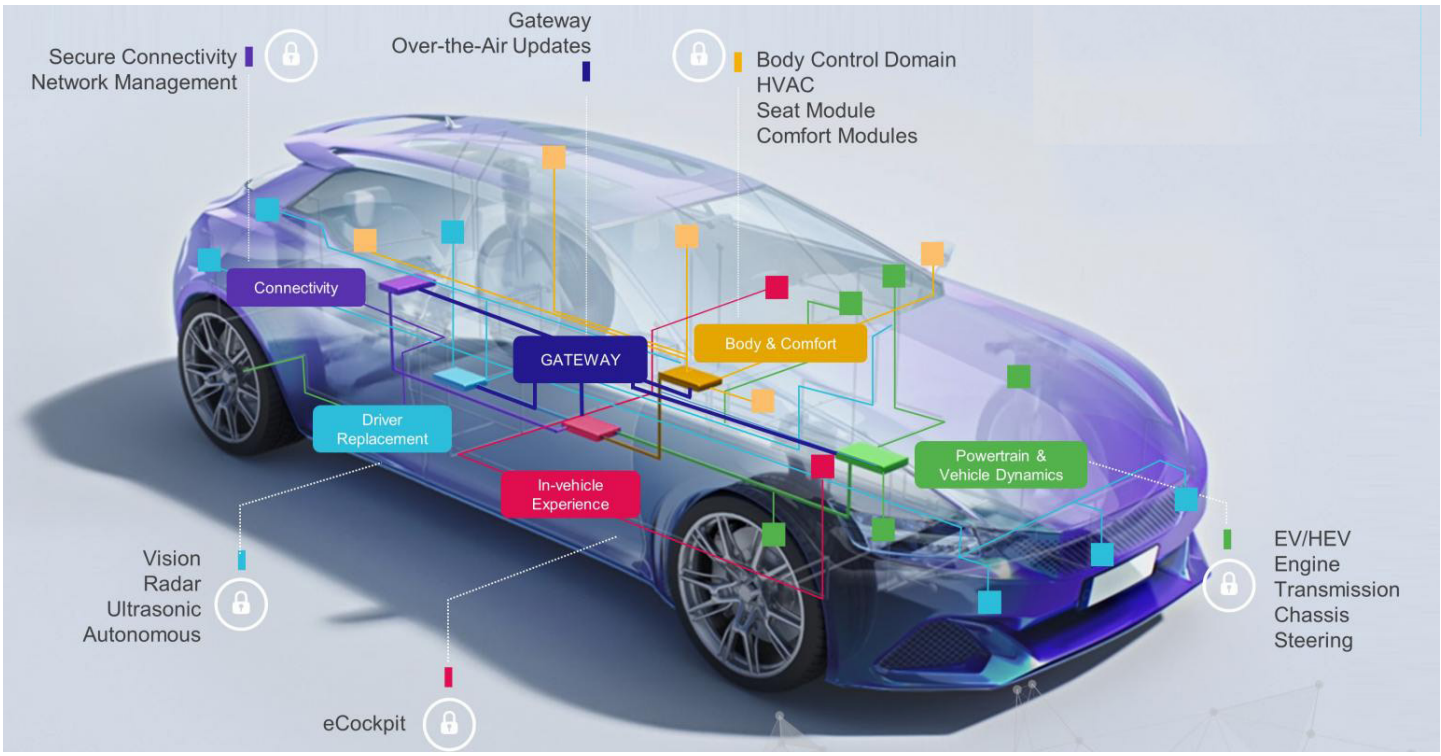
VIVIDSPARKS

Design and Comparative Evaluation of GPU and Tez based ECU Architectures for Secure, Dependable, and Real-Time Automotive CPS



Introduction

Contemporary automobiles integrate a multitude of heterogeneous digital processors (also called electronic control units (ECUs)), radio interfaces, in-vehicle networks and protocols, and hundreds of megabytes of complex embedded software. Next generation of automobiles (also known as cybercars) will further escalate the profusion of novel distributed control applications. Emergence of x-by-wire systems, where electronic controllers replace traditional mechanical and/or hydraulic subsystems, is a prominent example of recent modernization in the automotive industry. However, x-by-wire systems (e.g., steer-by-wire, brake-by-wire, etc.) have stringent real-time performance and reliability requirements, which pose significant challenges for implementation over traditional, bandwidth limited controller area network (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. Furthermore, FlexRay 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 cybercar design becomes imperative. The continuously escalating complexity of automotive systems and increasing integration with wireless entities (e.g., smart phones) exacerbate the security vulnerabilities of cybercars ([1]). Furthermore, harsh operational environment combined with external noise and radiation render ECUs vulnerable to permanent and transient faults. Hence, in order to make cybercars robust to faults and security vulnerabilities, cybercars 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 (CPS). The advancements in cryptanalysis and 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.

To address the above mentioned security, dependability, and performance challenges in automotive CPS design, we devise multicore Tez based ECU architectures which are secure, dependable, high-performance, energy-efficient, and flexible. We consider the interplay between temporal performance and dependability in our ECU designs. We emphasize that although temporal performance (measuring timing constraints) is a quality of service (QoS) measure, the temporal performance must also be considered as a dependability measure beyond a certain critical threshold as the driver can totally lose control of his/her vehicle if the response time exceeds that critical threshold.

To demonstrate temporal performance, energy efficiency, and error resilience of our proposed architectures, we consider steer-by-wire (SBW) as a case study. We further compare the performance and energy efficiency of our proposed ECU architectures with a GPU that embodies the security and dependability features for future cybercars.

Tez based ECU Architecture

Cybercars integrate a multitude of microcomputer units as ECUs to implement different automotive functions. automotive ECUs require high computational power to integrate newly emerging cybercar applications and services. Tez is flexible and scalable ECU architectures that simultaneously integrate security and dependability primitives with low resources and energy overhead.

Figure. 1 shows the generalized internal architecture of the Tez-based ECU. The ECU consists of an 32 multicore CPU in a grid and application-specific coprocessors. This application processor provides interface to the in-vehicle networks (e.g., CAN, CAN FD, FlexRay, 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 is POSIT based digital signal processor (DSP), 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 performance bus (AHB). The application processor of one ECU can communicate with the application processor of another ECU through in-vehicle networks (e.g., CAN, CAN FD, or FlexRay). During normal 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 through PCIe bus.

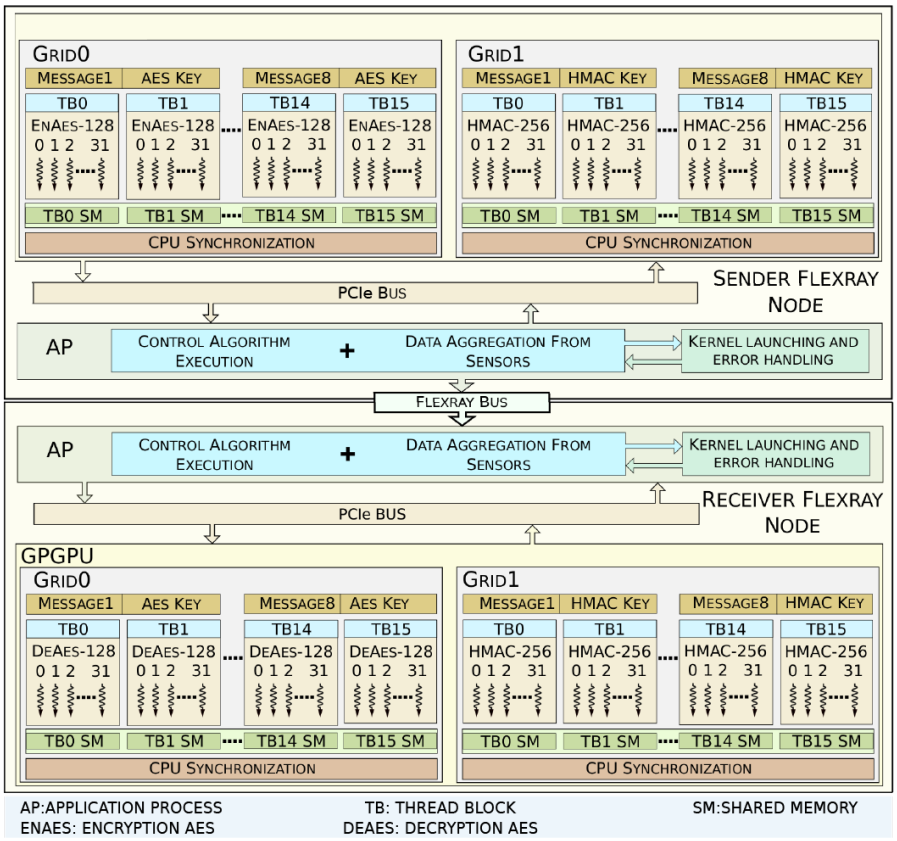


Figure 1: Tez based ECU architecture

Security: The cryptographic module with AES-128 encryption and SHA-3-based HMAC are implemented in Tez based co-processor. It process a a batch of eight 128-bit messages at once, we adopt this batch processing mechanism to utilize the massive computational power of Tez and to enhance the throughput of the cryptographic module. Furthermore, the implementation in Tez exploits the largely byte-parallel operations of AES-128 and lane-parallel operations of SHA-3. Our implementation adopts parallel granularity of byte-per thread for AES-128. Each byte of AES-128 is mapped to a thread in Tez. 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 Tez to ensure fast memory access.

Dependability: Each thread-block, and its redundant counterpart, performs a complete AES (or HMAC) computation. In the FT-RMT mode, 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 threadblocks, 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. In addition to error resilience, the fault tolerance provides resistance against the fault attacks that tries to inject soft-errors in the operating ECUs.

Modeling and Analysis of a Step-By-Wire Subsystem

In this section, we expands on the timing model of a SBW subsystem that leverages our purposed ECUs to incorporate security and dependability. We use this timing model to compute the quality of service (QoS) and behavioral reliability of the SBW 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. The SBW architecture is depicted in Figure. 2. In this work, we focus on on the FAC 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 ECU-to-ECU connection, we experiment on all three commercial automotive buses: CAN, CAN FD, and FlexRay. We evaluate and present a comparison of the SBW response time and error resilience when using these three buses. The operation of our SBW subsystem is same as in [2], however, our SBW subsystem leverages our proposed ECU architectures.

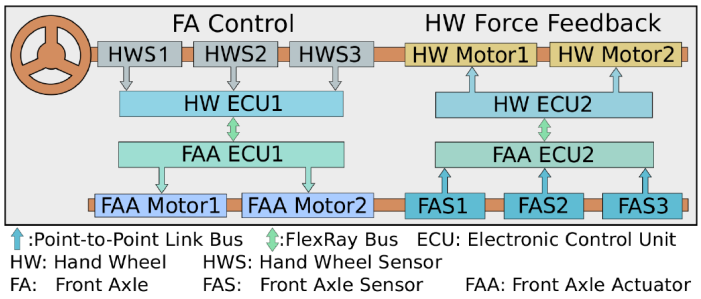


Figure 2: SBW architecture

B. Timing Model of SBW to Compute the QoS and Behavioral Reliability

The end-to-end delay/response time (τ_r) is the delay between the driver's request at the HW and the corresponding response at the front axle actuator (FAA). τ_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 τ^{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 (τ_r) is modeled as the sum of pure delay (σ_p), mechatronic delay (σ_m), and sensing delay (σ_s), as, $\tau_r = \sigma_p + \sigma_m + \sigma_s$. The mechatronic delay is introduced by the actuators (electric motor in our case). 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* [3]. For our secure and dependable approach, the pure delay (σ_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 (σ_p) for our FAC function can be written as,

$$\delta_p = rcc1 \cdot \delta_{hw}^{ecu1} + rtc \cdot \delta_{bus} + rcc2 \cdot \delta_{faa}^{ecu1}) \leq \delta_p^{max}, \qquad (1)$$

where σ^{ecu1}_{hw} and σ^{ecu1}_{faa} denote the computation time at HWECU1 and FAA-ECU1, respectively; σ_{bus} represents the transmission time for a message on an in-vehicle bus (CAN, CAN FD, or FlexRay) from HW-ECU1 to FAA-ECU1; *rcc1* and *rcc2* 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 σ^{max}_p represents maximum allowable σ_p. According to Wilwert et al. [4], with a

minimum tolerable QoS score of 11:13, the critical limit for pure delay σ_p^{max} is 35 *ms*, beyond which the vehicle becomes unstable and risks the driver's safety.

σ_{hw}^{ecu1} and σ_{faa}^{ecu1} are calculated as the sum of execution time of application processor, execution time of co-processor, Figure. 2). 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 σ_{hw}^{ecu1} and σ_{faa}^{ecu1}.

Results

We have implemented the security and dependability primitives in Tez architecture and on NVIDIA's RTX3090 GPU. The NVIDIA's GEFORCE RTX3090 GPU has 10496 CUDA cores and runs at 1.70 GHz clock speed with operating voltage 0.87. Tez has 32 cores runs at 300MHz with operating voltage 1.15V. The fault tolerant (FT) cryptographic modules are coded in PyTorch.

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.

Table 1 shows execution time and energy consumption profile for processing one ECU message for NVIDIA GPU GEFORCE RTX 3090 and Tez architectures. We evaluate our architectures with both NFT and FT operational modes. The comparison between GPU and Tez reveals that FT Tez is 2X faster than FT GPU and NFT Tez 5.5x faster than NFT GPU.

Energy Analysis: Energy efficiency is an important metric for automotive CPS as it implies greater fuel-efficiency for combustion engine vehicles and longer battery lifer for hybrid and electric vehicles. All of Tez ECU architectures are energy-efficient. Table I shows that the Tez yields better energy efficiency than the NVIDIA GPU. The NFT Tez and the FT Tez consumes 2.75X and 1.34X less energy than the NFT GPU and the FT GPU, respectively. This is because of the POSIT based energy efficiency of the Tez architecture and less execution time than that of the GPU.

Conclusion

In this paper, we have evaluated GPU architecture and Tez based ECU architecture. The salient features are dependability primitives while adhering to stringent real-time constraints of automotive CPS; the ability to perform compute-intensive applications, such as cryptography and audio, video, graphics, and image processing, in an energy-efficient manner; and the resistance of the ECU against fault injection and analysis attacks. Furthermore, we have quantified and compared temporal performance, energy, and error resilience of GPU and Tez ECU architectures for a SBW case study over CAN, CAN FD, and FlexRay in-vehicle networks. Results reveal that Tez can attain a speedup of 2X and 5.5X, respectively, while consuming 2.75X and 1.34X less energy, respectively, than the contemporary ECU architectures.

		NVIDIA GPU			Tez		
In Vehicle mode	Operational mode	FT mode	Time (us)	Energy (uJ)	FT mode	Time (us)	Energy (uJ)
Sender Node	NFT	none	67.18	2.675	None	12.11	1.433
	FT	FT-RMT	70.32	3.143	FT-RMT	34.81	2.187
Receiver Node	NFT	none	71.34	4.801	none	47.44	1.321
	FT	FT-RMT	72.67	3.098	FT-RMT	36.23	2.459

Table 1: Performance and Energy results for GPU and Tez

References

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

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

[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] 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.