Reconfigurable Computing in Next-Generation Automotive Networks

Shanker Shreejith, Suhaib A. Fahmy, and Martin Lukasiewycz

***Abstract*—Modern vehicles incorporate a significant amount of computation, which has led to an increase in the number of computational nodes and the need for faster in-vehicle networks. Functions range from noncritical control of electric windows, through critical drive-by-wire systems, to entertainment applications; as more systems are automated, this variety and number will continue to increase. Accommodating the varying computational and communication requirements of such a diverse range of functions requires flexible networks and embedded computing devices. As the number of electronic control units (ECUs) increases, power and efficiency become more important, more so in next-generation electric vehicles. Moreover, predictability and isolation of safety-critical functions are nontrivial challenges when aggregating multiple functions onto fewer nodes. Reconfigurable computing can play a key role in addressing these challenges, providing both static and dynamic flexibility, with high computational capabilities, at lower power consumption. Reconfigurable hardware also provides resources and methods to address deterministic requirements, reliability and isolation of aggregated functions. This letter presents some initial research on the place of reconfigurable computing in future vehicles.**

***Index Terms*—Automotive applications, automotive electronics, field buses, field-programmable gate arrays (FPGAs), reconfigurable architectures.**

I. Introduction

VEHICLES presently incorporate up to one hundred electronic control units (ECUs), which control and coordinate both critical and noncritical functions. Many existing functions like antilock braking system (ABS) and adaptive cruise control (ACC), as well as emerging capabilities like drive-by-wire, rely on complex control algorithms that are computed under hard or soft real-time requirements. These computations are performed on data collected from a multitude of sensor nodes over the in-vehicle communication infrastructure. Currently, such functions are implemented as software on processor-based platforms to provide flexibility and upgradability. The hard requirements of real-time responsiveness and fail-safe operation imposed by critical systems determine the choice of node architecture and in-vehicle communication protocols.

Modern automotive systems are moving on from the event-triggered controller area network (CAN)—the most widely used automotive networking protocol since the early 1990s—to more advanced and complex time-triggered protocols like FlexRay and switched Ethernet [1]–[3]. Time-triggered protocols like FlexRay provide multiple bus access schemes to support deterministic data transfer and priority-based volume data transfer, to address the requirements of determinism, bandwidth and reliability. As more capabilities are added to vehicles, the number of ECUs and functions per ECU increases, putting additional strain on both the computational devices and the network. Complex scheduling can extend the effective bandwidth of the network without impacting reliability. A FlexRay switch exploits branch-level and slot-level parallelism in the communication schedule to optimise available bandwidth, allowing a switch to support different schedules for each communication cycle [4]. This complex scheduling requires more capable nodes and network controllers.

The consistently increasing number of functions adds further challenges: additional weight, installation space and energy consumption. Multiple functions should ideally be aggregated onto fewer ECUs to optimise these factors. In electric vehicles, the total weight of the on-board computing and communication systems and their power consumption are critical factors. Leveraging the computational capabilities of reconfigurable hardware, it is possible to create optimised nodes that integrate ECUs and network controllers on a single piece of hardware, resulting in less power consumption, and weight [5]. Field-programmable gate arrays (FPGAs) provide us the flexibility and resources to implement complex real-time applications while integrating networking functionality like FlexRay switching or extensions like FlexRay to CAN/Ethernet/MOST bridges.

FPGAs improve upon processor-based ECUs by providing better determinism and segregation. The customizable and reconfigurable nature of the fabric can be exploited by implementations addressing a wide range of applications for modern and future in-vehicle systems. Non-safety-critical applications like multimedia and driver assistance can leverage the high computational capability of FPGAs while taking advantage of dynamic reconfigurability to multiplex functions. FPGAs also enable techniques for implementing multiple levels of fault-tolerance and redundancy to support FPGA-based safety-critical ECUs for the drive train and drive-by-wire systems.

II. Proposal

The in-vehicle network architecture can be partitioned into different domains based on performance and/or safety require-

Manuscript received November 23, 2012; accepted January 18, 2013. Date of publication January 30, 2013; date of current version February 22, 2013. This work was supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) Program. This manuscript was recommended for publication by J. Hoe.

S. Shreejith is with the School of Computer Engineering, Nanyang Technological University, Singapore 637553, Singapore, and also with TUM CREATE Centre, Singapore (e-mail: shreejit1@e.ntu.edu.sg).

S. A. Fahmy is with the School of Computer Engineering, Nanyang Technological University, Singapore 637553, Singapore (e-mail: sfahmy@ntu.edu.sg).

M. Lukasiewycz is with the TUM CREATE Centre, Singapore (e-mail: martin.lukasiewycz@tum-create.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2013.2243698

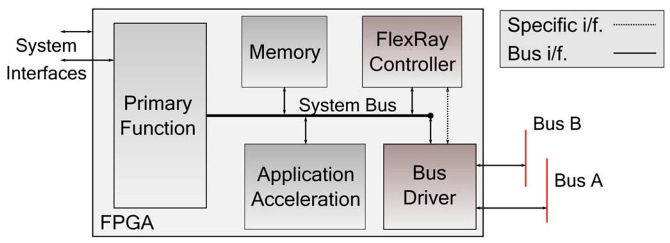

Fig. 1. Complete ECU on Reconfigurable Fabric.

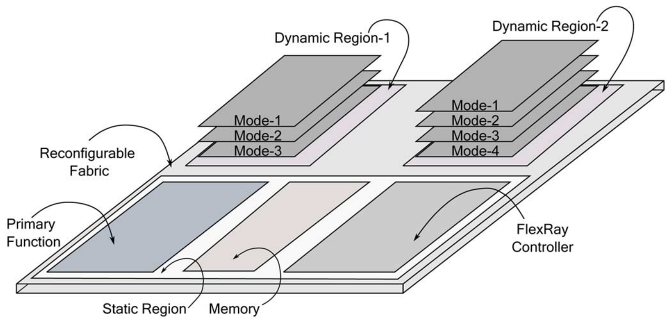

Fig. 2. Custom computing ECU systems on FPGAs using PR.

TABLE I
MODES OF OPERATION

Functionality	Modules in Dynamic Region 1	Modules in Dynamic Region 2
Park Assist	Custom Logic	Sensor Interfaces
Application Acceleration	Softcore Processor	Custom Logic Interfaces
Cruise Control	Adaptive Logic	Sensor Interfaces
Safety-critical ECU	ECU Function	Redundant ECU

ments. These require different levels of service, measured by response time, bandwidth, redundancy, error detection among others, and are often referred to as quality of service (QoS) levels [6]. ECU functions are presently implemented as software on general purpose processors. Discrete network interface chips (or integrated IPs) enable access to the in-vehicle networks like CAN, FlexRay, or others. Depending on the domain the implemented functions fall into, an ECU may also have distinct interfaces to one or more of the in-vehicle networks.

Having a flexible ECU implemented on reconfigurable hardware, allows for an architecture that can be seamlessly integrated onto multiple domains. Fig. 1 shows an architecture that encapsulates the interface (FlexRay, in our case) and the functional unit. The primary function implements an algorithm that requires complex computations to be performed on data using a provided acceleration logic. Communication to the sensors and other ECUs over the FlexRay bus is controlled by the FlexRay communication controller (CC) and the bus driver module. This ECU architecture on a reconfigurable platform can be augmented to implement complex gateways between multiple in-vehicle networks, among other possible applications.

A. Non-Safety-Critical Systems

Non-safety-critical systems usually include user-oriented features like multimedia, telematics, remote diagnostics, and future systems like vehicle-to-vehicle (V2V) communication. Such systems are characterized by the high volume of data handled, high throughput requirements and complex computation, an area where FPGAs represent an ideal implementation platform. Indeed the computational power of custom hardware on FPGAs enables applications that would otherwise be infeasible on low-power processors. Since FPGAs implement computation spatially, we can split the available resources among multiple functions, maintaining the predictability of each while ensuring complete isolation between them.

Furthermore, we can time multiplex applications that are not needed concurrently by using dynamic partial reconfiguration (PR) [7]; mutually exclusive functions are mapped to the same dynamic region on the device, which can be reconfigured at run time. Primary ECU functionality can be defined in the static, nonchanging region of the device, while each dynamic region would have specific functions or accelerators for the current operating mode, as illustrated in Fig. 2. The illustration described in Table I shows the different functions that can be integrated on a smart-node and the distinct interfaces or modules required to implement them. The FPGA design is partitioned to incorporate concurrent modules in different dynamic regions, considering the computational and bandwidth requirements of each. The dynamic regions can then be reconfigured, when needed, to integrate multiple nonconcurrent functions on the same node. This architecture can be extended to integrate complex adaptive systems for current and future in-vehicle applications, on a much smaller device.

Future automotive systems significantly increase the amount of data that is gathered for processing and use algorithms that are significantly more complex. Examples are driver assistance systems like pedestrian detection or blind spot warning. Software implementations of such algorithms require specialized processors or multicore systems while hardware implementations provide more performance at lower power. The computational capabilities of FPGAs can be exploited to provide an efficient and flexible solution that also integrates the communication interfaces to the various sensors and in-vehicle networks on the same hardware, saving significantly on infrastructure. Furthermore, PR can be employed to reuse the dynamic areas of an FPGA to offload computations from another ECU, while these resources are not being used by the current application.

Future technologies like V2V and vehicle-to-grid (V2G) communication further push in-vehicle computing requirements. V2V and V2G allow vehicles to communicate relevant data for advanced driver assistance, collision avoidance and remote assistance. Communication is achieved over a multihop self-organizing wireless network, comprising the mobile units (vehicles) and fixed infrastructure [8]. Beyond the ad-hoc network nature, authentication, security and timeliness of data present further challenges, since delayed, fraudulent or tampered data can result in misguiding commands to the user [9]. Hardware implementation of security and authentication protocols is more efficient, since many computations can be easily handled at bit level in specialised hardware [10]. Reconfigurable computing allows us to tightly couple the security requirements with an application at marginal extra cost in a manner that discrete devices cannot.

B. Safety-Critical Systems

Safety-critical systems like drive-by-wire, ABS or occupant safety systems are hard real-time systems requiring high levels of determinism and isolation. They may have to interface with multiple in-vehicle networks to control and coordinate operations of critical systems like the drive-train. Safety-critical systems are often implemented to support fail-safe or fault-tolerant operation. A fail-safe system ensures that a critical failure at the node does not cause a catastrophic failure of other systems. A node maintains a very basic set of tasks, without requiring a halt due to fatal errors [11]. Fail-safe operation requires simpler hardware and is hence often chosen in production for noncritical systems.

A fault-tolerant system is more robust and can adapt and recover from faulty situations without severely degrading system performance, often achieved by redundancy. FPGA-based designs can instantiate multiple instances of identical ECUs within the same device to aid redundancy while providing better determinism [7], [11]. FPGA-based designs that incorporate PR provide alternative solutions for redundancy since PR allows us to reconfigure only necessary logic rather than the whole FPGA. In a fault-tolerant scheme, an error causes the logic to switch to a redundant mode which operates with lower specifications. PR enables us to reconfigure the faulty region alone, without affecting current operations, resulting in faster turnaround times. The fault detection logic on the FPGA triggers the switch to the redundant mode of operation and the subsequent reconfiguration of appropriate dynamic region(s), when a critical error is detected. Also, multiple implementations of an application with differing levels of error tolerance can be swapped in on the fly to deal with changing conditions. In addition, a single region of programmable fabric can be assigned as the redundant region for multiple functions, rather than the need for distinct circuitry for all systems to be present at the same time.

Deterministic behavior is easily factored into systems implemented on reconfigurable hardware. FPGA-based designs are synchronous, event-triggered systems and hence respond to events in a deterministic manner. Hardware-level parallelism can be exploited by designs to ensure that multiple simultaneous events can be handled independently without contention. Specific events like single-event upsets (SEUs) can be mitigated in logic, using either fail-safe or fault-tolerant design methods. Incorporating PR, the erroneous ECU (or function) alone can be reconfigured without affecting other regions, providing higher determinism.

Processor-based ECU systems suffer from a lack of isolation when many ECU functions are aggregated onto a single processor. An operating system controls and coordinates the different software functions and shares processor resources, data memory and caches, among them, which can result in unpredictable contention unless specific steps are taken to manage this. Moreover, the overhead and the associated delay in interrupting one function to process another request in a traditional system depends on a range of parameters including the current state of system and thus, can be unpredictable. On the other hand, aggregating ECU functions onto an FPGA can be done by partitioning the device so that different functions do not share hardware resources, thus allowing them to operate simultaneously and completely independent of each other. This setup provides higher levels of determinism and isolation, allowing for much better aggregation of safety-critical and/or non-safety-critical applications on the same hardware, hence reducing device count.

TABLE II
SPARTAN-6 IMPLEMENTATION OF ECU ON CHIP

Utilisation	FlexRay CC	Hardware Accelerator	Complete ECU
Registers	4922	4216	11778
LUTs	7969	3221	13566
Occupied Slices	2665	1088	5005
BlockRAM	13	11	60
DSP48	3	44	48
Power Consumption		291 mW	

III. RESULTS

We have already seen that FPGA-based custom solutions provide multiple advantages over traditional processor-based ECU systems. We now further quantify this by demonstrating a custom FlexRay communication controller, designed to leverage the heterogeneous resources on modern FPGAs. We present an ECU node that combines this controller with a Microblaze softcore processor on a Xilinx Spartan-6 XC6SLX45 device. We have built a FlexRay traffic generator to allow us to test the system. The ECU functions as a front-end processing node for radar-based cruise control and is built using Xilinx FFT IP cores and pipelined logic which performs target detection using the constant false alarm rate (CFAR) scheme [12]. The test data generates 1024 data points every 30 ms, which is transformed to the frequency domain and processed by the pipelined logic. Results are passed onto the FlexRay bus in preconfigured slots. Table II details the resource utilization and power consumption measured during operation in hardware. Such an application would require specialized DSP processors, since the latency cannot be met by software implementation on a general purpose processor [12]. The key advantage here is that integrating ECU functionality and the network interface onto the same device only increases the total power usage marginally, compared to similar architectures built around standalone controllers like the Infineon CIC310 (which alone consumes 150 mW of power) [13], and this interface can be shared between multiple functions on the same FPGA.

Consolidating multiple nonconcurrent ECUs on a single device reduces the number of ECU modules, bus drivers, and the associated wiring, all contributing towards a better in-vehicle ecosystem. Traditionally, each ECU uses discrete (or integrated) controllers to access the bus. In an FPGA-based node, PR can be utilized to consolidate multiple functions at much lower power consumption, while sharing the interface between multiple functions can be managed in hardware in a predictable, fair manner.

We have integrated the above-mentioned cruise control application with an intelligent parking solution [14] on a Xilinx ML605 development board containing a Virtex-6 FPGA

TABLE III
VIRTEX-6 IMPLEMENTATION OF ADAPTIVE SMART ECU

Mode		Utilisation		Switching Time	Dynamic Power
Design	Operation	Registers	LUTs		
PR	Idle	12223	13695	-	240 mW
	Park Assist Cruise Control	4125	2130	936.4 ms	340 mW
Static	All Modes	32484	31558	NA	640 mW

(XC6VLX240T). The parking algorithm design is based on fuzzy logic. Since these applications are mutually exclusive, we can utilise PR to create an adaptive node, which modifies its functionality based on current requirements. We use the Xilinx hardware ICAP module (XPS_HW_ICAP), integrated with the Microblaze processor, to manage reconfiguration when a mode change is required. The various components are connected using a Processor Local Bus (PLB) interface.

The PR region is normally filled by a blank bitstream in *IDLE* mode. A mode switch is triggered by user commands, and is transmitted over the FlexRay bus. The custom FlexRay controller processes the received command, bypasses the datapath and issues a high priority interrupt to the Microblaze processor, initiating the reconfiguration process. The ISR reads the partial bitstream from the compact flash card through the System ACE controller and sends it to the ICAP. Table III shows the implementation metrics for this design, including resources consumed by these modules (in the partial region), and the dynamic power consumption while operating in these modes. Since the partial bitstream size is the same for both modes, the time taken to switch between them is identical. The results also show that the adaptive node has a definite advantage in terms of power consumption and utilization compared to a purely static implementation, integrated as two isolated functions on the same device. Use of custom high speed reconfiguration controllers [15] would enable a much faster turnaround time, making FPGA-based fault-tolerant nodes more viable for safety-critical applications [16].

The Virtex-6 device was chosen for its native PR support. However such a device would not normally be considered for in-vehicle implementation due to higher power consumption and cost. In the next-generation 7 series FPGAs from Xilinx, PR is supported across the full range including the low-power devices that would be used in vehicles, paving the way for integration of such smart nodes for in-vehicle adaptive systems. Xilinx 7 series FPGAs also provide intelligent clock gating, which can help reduce the power consumption in *IDLE* mode. Furthermore, the Xilinx Zynq-7000 series, equipped with a dual-core ARM processor and a fabric supporting PR, would enable aggregation of multiple functions on the fabric while retaining traditional software capabilities. This type of consolidation is at the heart of the reconfigurable paradigm for automotive systems.

IV. CONCLUSION

Complex functions are being added to in-vehicle systems to enhance human machine interaction and performance. User-oriented applications like multimedia or telematics require high computational throughput. Meanwhile, safety-critical systems demand determinism, time-bound response and fail-safe (or fault-tolerant) operation. Reconfigurable architectures allow us to cater to the demands of both ends of this application spectrum, by providing high computational capability, predictability, determinism and better isolation of aggregated functions, while also allowing us to augment functionality through reconfiguration. While addition of newer functions using a legacy scheme would have to be justified in terms of the cost of additional ECUs and infrastructure, the reconfigurable paradigm opens the doors for the proliferation of more advanced and complex in-vehicle systems, but with the reduced system footprint. Techniques like PR can be leveraged to consolidate more functions on existing resources, allowing top end vehicles to incorporate more cutting edge features. Moreover, FPGAs can also be updated in-field to provide improved functionality.

The potential advantages offered by next-generation FPGAs, that incorporate encrypted bitstreams, gated clocks and PR, outweigh traditional limitations. FPGA-based designs can enable future context-aware adaptive computational systems, with lower power consumption and weight (through consolidation), both of which are critical for next-generation electric vehicles.

REFERENCES

[1] *CAN Specification, Version 2.0*, , 1991, R. Bosch GmBh, Std.

[2] *FlexRay Communications System, Protocol Specification Version 2.1 Revision A*, , Dec. 2005, FlexRay Consortium Std..

[3] H.-T. Lim, L. Vlker, and D. Herrsche, "Challenges in a future IP/ethernet-based in-car network for real-time applications," in *Proc. Design Autom. Conf. (DAC)*, 2011, pp. 7–12.

[4] P. Milbredt, B. Vermeulen, G. Tabanoglu, and M. Lukasiewycz, "Switched FlexRay: Increasing the effective bandwidth and safety of FlexRay networks," in *Proc. Emerg. Technol. Factory Autom. (ETFA)*, 2010, pp. 1–8.

[5] S. Chakraborty, M. Lukasiewycz, C. Buckl, S. Fahmy, N. Chang, S. Park, Y. Kim, P. Leteinturier, and H. Adlkofer, "Embedded systems and software challenges in electric vehicles," in *Proc. Design Autom. Test Eur. (DATE) Conf.*, 2012, pp. 424–429.

[6] N. Navet, Y. Song, F. S. Lion, and C. Wilwert, "Trends in automotive communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1204–1223, Jun. 2005.

[7] F. Fons and M. Fons, "FPGA-based automotive ECU design addresses AUTOSAR and ISO 26262 standards," *Xcell J.*, no. 78, pp. 20–31, 2012.

[8] J. Luo and J.-P. Hubaux, A Survey of Inter-Vehicle Communication School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, Tech. Rep. IC/2004/24, 2004.

[9] P. Caballero-Gil*, Mobile Ad-Hoc Networks: Applications*. New York, NY, USA: InTech, 2011, ch. 4, Security Issues in Vehicular Ad Hoc Networks.

[10] X. Guo, Z. Chen, and P. Schaumont, "Energy and performance evaluation of an FPGA-based SoC platform with AES and PRESENT coprocessors," in *Proc. Int. Workshop Embed. Comput. Syst.: Arch., Modeling, Simulation (SAMOS)*, 2008, pp. 106–115.

[11] N. Chujo, "Fail-safe ECU system using dynamic reconfiguration of FPGA," *R&D Rev. Toyota CRDL*, vol. 37, pp. 54–60, 2002.

[12] J. Saad, A. Baghdadi, and F. Bodereau, "FPGA-based radar signal processing for automotive driver assistance system," in *Proc. IEEE/IFIP Int. Symp. Rapid Syst. Prototyping (RSP)*, 2009, pp. 196–199.

[13] SAK-CIC310-OSMX2HT, FlexRay Communication Controller Data Sheet Infineon Technologies AG, Jun. 2007.

[14] I. Song, K. Gowan, J. Nery, H. Han, T. Sheng, H. Li, and F. Karray, "Intelligent parking system design using FPGA," in *Proc. Int. Conf. Field Programmable Logic Appl. (FPL)*, 2006, pp. 1–6.

[15] K. Vipin and S. Fahmy, "A high speed open source controller for FPGA partial reconfiguration," in *Proc. Int. Conf. Field Programmable Technol.(FPT)*, 2012, pp. 61–66.

[16] S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz, "An approach for redundancy in FlexRay networks using FPGA partial reconfiguration," in *Proc. Design, Autom, Test Eur. (DATE) Conf.*, 2013.