Reconfigurable ECU communications in Autosar Environment

Hung-Manh Pham, Sebastien Pillement and Didier Demigny
University of Rennes 1
CAIRN IRISA/INRIA
6 rue de Kerampont, F-22300 Lannion
email: hpham@irisa.fr, pillemen@irisa.fr, demigny@irisa.fr

***Abstract*—Reconfigurable architectures become popular in consumer applications as they require low Non-Recurring Engineering costs. Certain FPGA families provide the Dynamic Partial Reconfiguration (DPR) mechanism which allows to modify parts of a device on-the-fly. The high reliability constraints in automotive systems requires safe communications between ECUs (Electronic Control Unit). This paper presents a support for fault-tolerant communication modes among ECUs by the use of the dynamic reconfiguration paradigm. Also in this paper, a method of integrating dynamic reconfiguration in Autosar environment is presented.**

***Index Terms*—Flexible Communication, FPGA, Dynamic Partial Reconfiguration, Autosar**

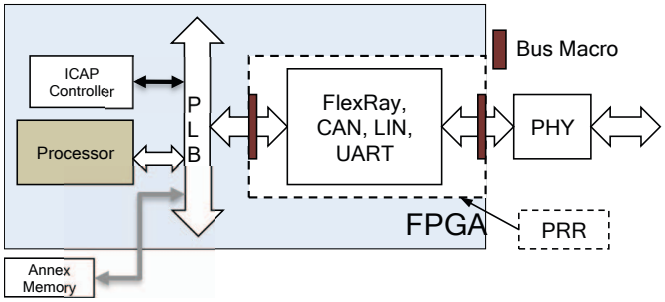

Fig. 1. Dynamic Automotive Platform

I. Introduction

In the automotive domain, beside the fault-tolerant requirements of electronic components, the communication networks must be guaranteed as well. In order to ensure good communications between ECUs inside the vehicle, one possible solution is to foresee a secondary communication channel. If the principal communication fails, the system can switch to a degraded mode (with the secondary protocol) and continue to run. Using ASICs (**A**pplication-**S**pecific **I**ntegrated **C**ircuit) to construct such system needs high costs to develop, to maintain and to upgrade. On the contrary, reconfigurable architectures need less costs to design and they are flexible.

So, to define a new embedded automotive platform based on reconfigurable architecture, the research activities of the CIFAER (Flexible Intra-Vehicle Communications and Embedded Reconfigurable Architectures) project [1] attempt to build dynamically adaptable interfaces supporting flexible communications. In CIFAER we advocate for the use of Radio Frequency (**RF**) and **P**ower-**L**ine **C**ommunication (**PLC**) for intra-vehicle communications [2]. The communication can be switched from one to the other by dynamically reconfiguring a defined communication zone on an FPGA. These two modes offers very flexible links inside a vehicle.

A dynamically reconfigurable system allows to change parts of his logic resources without disturbing the functioning of the remaining circuit. This property permits the system to change its behavior according to external events. Actually, the Xilinx Virtex FPGAs [3] are the only commercially available circuits supporting the DPR and large applications implementation. The dynamic reconfiguration takes place in **P**artially **R**econfigurable **R**egion (PRR) which can be partially reconfigured independently [3]. Designing a dynamically reconfigurable system always requires the declaration of PRRs. The partial bitstreams of these zones, are stored in an external memory and they contain all the information about the positions and functionalities of the considered PRRs. A dynamic reconfigurable system usually has a central processor connecting to the internal reconfiguration port (ICAP) and controlling the partial reconfiguration process by downloading bitstreams onto this port (**Fig. 1**). The ICAP and this controller are implemented in a static zone (i.e. not reconfigured) of the FPGA. Except for the dynamic zone which is being reconfigured, the whole FPGA is still on operation during the entire reconfiguration process.

In one PRR, several PRM (Partially Reconfigurable Module) could be loaded (one at a time). Each PRM is individually designed and implemented using the early access partial reconfiguration design tools [3]. All PRMs for a given PRR must be pin compatible with each other, i.e., have the same port definitions and entity names.

Another problem in designing an automotive system is the Autosar conformity. Autosar (AUTomotive Open System ARchitecture) is an architecture regrouping manufacturers, suppliers and tools developers to define a de-facto open industry standard for automotive Electric/Electronic (E/E) architectures. The main objectives of Autosar is to manage the increasing E/E complexity coming with the growth of functional components, to improve the flexibility and scalability for modification, update and upgrade during product lives, and to improve fault-tolerance capability of automotive E/E systems. But one major drawback of Autosar is the inflexibility of

978-1-4244-5347-4/09/$26.00 ©2009 IEEE

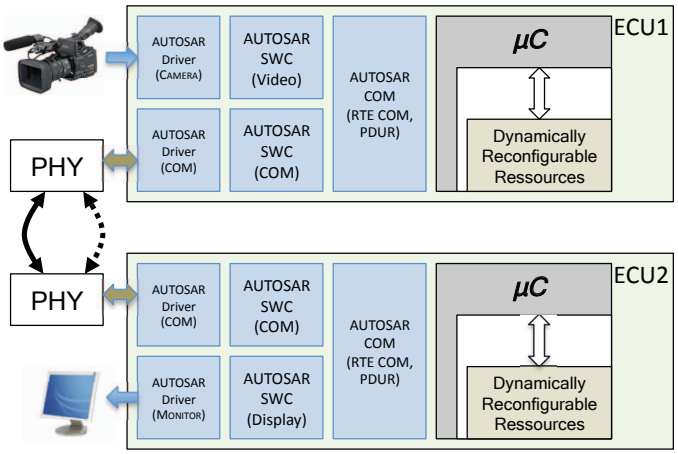

Fig. 2. System Overview with Autosar Integration

system parameter configurations, which require lots of definite sets at design time.

The goal of this paper is to study the possible solutions in order to integrate the dynamicity and the flexibility into Autosar architecture Since CPL and RF communication protocol are not yet standard in the automotive domain, we demonstrate the feasibility of the approach by the use of FlexRay [4] and CAN [5] standard automotive protocol.

II. Context

The works related to this paper aim at demonstrating the feasibility of the dynamic reconfiguration of a network during the communication in the context of automotive applications, and then studying the capability of the adaptation to the Autosar environment. The application envisaged in the CIFAER project (**Fig. 2**) includes 2 ECUs: one ECU connects to a camera, controls the video acquisition and sends the data through the primary communication links, while the other ECU receives the data and displays it on a screen. The communication links are ensured by the combination of RF and PLC Communication in dynamic reconfiguration. The switch between them is done according to a fault-tolerant scheme. When errors occur in the communications links, the communication protocol will be altered to a new protocol by reconfiguring the communication IP.

The **Fig. 3** shows the timing schedule of the application scenario. The video application programs (video record and display) and communication processes are executed in parallel thanks to their own hardware accelerators. There are 2 links for the communication between 2 ECUs: reception and emission. To detect a link failure, both ECUs regularly send to each other a detection frame (SYNC). If after a certain delay (t_e), one ECU do not receive the detection frame, this link is assumed to be failed (ERROR). If one or both links fail, there is a connection problem. In this case, the dynamic reconfiguration process will be realized to switch from one communication protocol to the other (SWITCH). Afterwards, both ECUs send a resynchronization frame (RESYNC) to re-establish the connection and start to transfer video data with

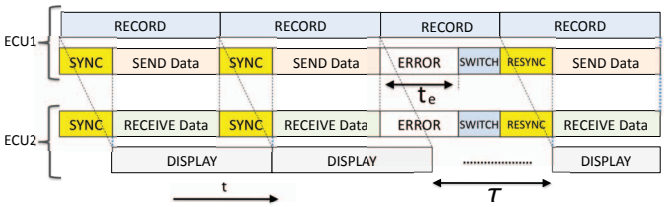

Fig. 3. System Timing Diagram

the new protocol. The video data lost because of link failure can be presented like the equation below:

$$\tau = t_e + t_{trans} + t_{switch} \tag{1}$$

where t_e is the synchronization duration limit, beyond that the link is supposed to be failed.

t_{trans} is the delay time during the transmission,it is assumed to be a zero delay transmission.

t_{switch} is the reconfiguration duration needed to switch from a protocol to another one.

$$t_{switch} = \frac{size[byte]}{TP[\frac{byte}{s}]} \tag{2}$$

where $size$ is the bitstream size in byte and TP is the process throughput when copying the bistreams from the stocking memory and downloading it onto the reconfiguration port.

Since t_e and t_{trans} depend significantly on the communication mode, the most touchable parameter is t_{switch} to reduce τ. In the **Equation 2**, size cannot be much modified. Therefore, one possible solution to reduce disfunction period is to accelerate the reconfiguration process by increasing the process throughput: increasing the processor frequency or using a high-speed stocking memory.

The synchronization and video data processing are similar in 2 communication modes. Nearly the same raw data will be processed with 2 different communication modes, that will lead to different processing durations. This aspect has to be taken into account at the application level.

III. Autosar Integration

Within the automotive context, the Autosar [6] architecture is defined by a consortium of main actors of the automotive domain to create and establish an open standards for automotive E/E platforms. The utilization of Autosar can cope with big challenges that we may face when integrating a platform into automotive system: huge effort to reuse and relocate functions between ECUs, lack of compatible tooling, a lot of time investigated in implementing and optimizing modules which is transparent to customer.

The **Fig. 2** presents the software structure of a reconfigurable ECU integrated in the Autosar architecture as we envisage it in the CIFAER project.

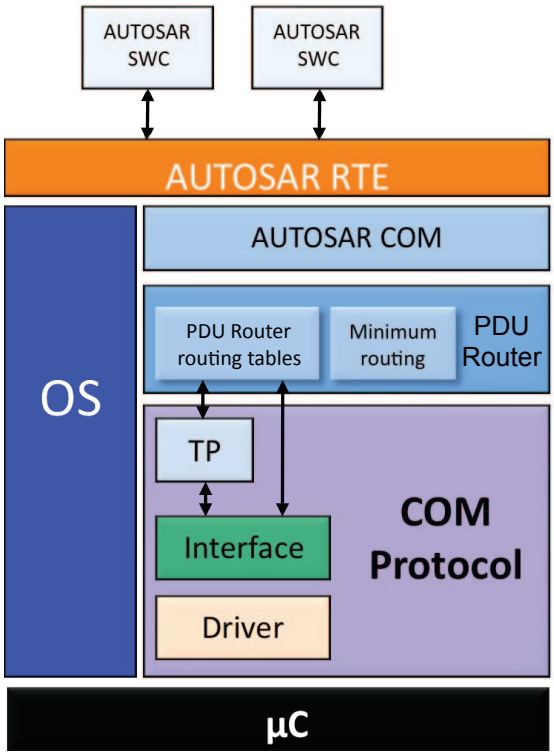

Fig. 4. ECU Architecture in Autosar

Autosar Communication Layer [7] (Autosar COM in **Fig. 2**) acts as a communication center for inter- and intra-ECU information exchange.

Access to the hardware is routed through the Autosar Drivers (Camera, Monitor and Communications) to avoid direct access to microcontroller registers from high-level software. As specified in the Autosar standard, the Communication Layer shall support static communications only. And actually, the μC, who controls the reconfiguration, have two different software programs for the two communication modes. In order to integrate the dynamicity into Autosar architecture, Communication Drivers and RTE Communication Layers must statically foresee the 2 cases of the reconfiguration.To do so, the post-build attribute in Autosar architecture is exploited. The principle of Autosar post-build mechanism is like following: the software may receive its configuration file that can be downloaded to the ECU separately, avoiding a re-compilation and re-build of the ECU software modules. So as to make the post-build time re-configuration possible, the re-configurable elements shall be stored at a known position in the ECU storage area.

The **Fig. 4** depicts an ECU in Autosar architecture. All the concerned components and layers need to be investigated following the top-down order.

The Autosar RTE of each ECU is established at design time, so all the interface with Autosar RTE are strictly fixed. Therefore all the modifiable parameters will be included in lower layers like PDU Router and COM Protocol components (**Fig. 4**). When the switch occurs, the appropriate communication drivers and communication components are selected depending on the chosen protocol model transparently from the Autosar RTE point-of-view.

In our system, the Communication Protocol (COM Protocol) is dynamic. So all the components of the protocol must be constructed in the post-build way, consequently idem for PDU Router and Autosar COM.

The Autosar Communication Layer can allow the configuration of communication at post-build time stage in 2 modes:

- **Loadable configuration**: at a given moment, only one configuration parameter set resides in the ECU. However, this configuration can be changed by flashing the dedicated memory segment of the ECU where contains the configuration.
- **Multiple configuration sets**: there is a set of multiple configuration parameters and one configuration is selected out by passing a specific pointer to the init function.

When the configuration change is requested, the Communication Layer chooses the proper parameters set according to the above modes. The change request will reflect the modifications upon PDU Router and Communication Protocol components without disturbing Autosar Communication Layer interfaces.

PDU Router [8] provides service for routing of data packets between:

- communication interface modules (e.g CAN, FlexRay)
- transport protocol modules (e.g CAN TP, FlexRay TP)
- Autosar COM and communication interface modules
- .....

The routing capability of PDU Router is managed through the routing tables. The PDU Router shall also support the update of the routing configuration post-build time. There are 2 modes of post-build routing tables like the Autosar Communication Layer: loadable configuration and multiple configuration sets. In case of loadable configuration, the PDU Router routing tables may be only updated when they are not in use, the update process takes place by flashing the memory segments where contains the table - a dedicated program may be loaded to update the tables.

Especially, the PDU Router provides a minimum routing capability (**Minimum routing** in **Fig. 4**) to be able to route specific data packets without using routing tables. These minimum routing settings are independent on the PDU Router routing tables and cannot be changed after build-time.

The Transport Layer (TP) [9], [10] is the module between the PDU Router and the Interface component. The main purpose of the module is to segment and reassemble messages which do not fit the protocol standard length. The routing capability depends on the incoming data packet identifier. The PDU Router is in charge of determining whether a Transport Layer need to be used or not. The main configurable items of the Transport Layer are mapping of data into frames, frame priority and frame timing.

The Interface component in the Communication Protocol belong to the Hardware Abstraction Layer. It provides to upper

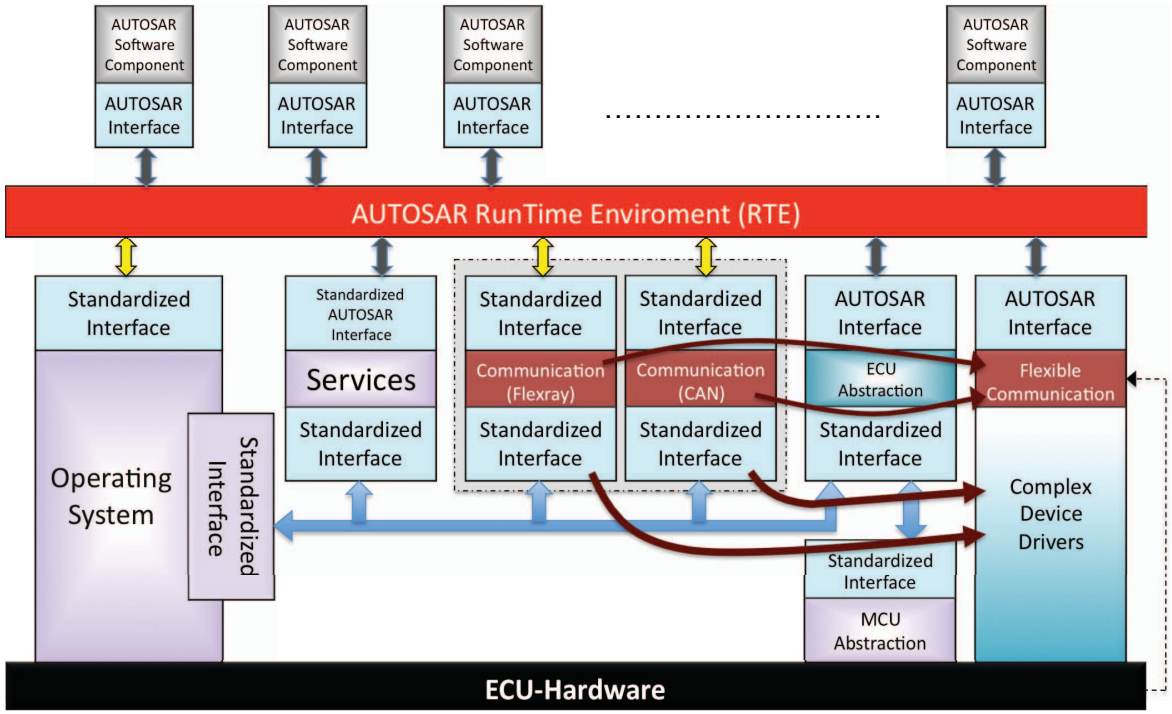

Fig. 5. Autosar Overview of Flexible Communication

layers an abstract interface to the communication system. At data transmission level, this interface shall be uniform for all bus system in Autosar. So upper layers like PDU Router and Transport Protocol may access all underlying bus systems for data transmission in a uniform manner. The most important aspect to be considered when altering the communication modes is the timing difference. Additionally, the interface is bus-specific, we then need to take into account dedicated features of each bus (in this case FlexRay and CAN) when merging them into one unified mode.

The drivers for the communication protocol in **Fig. 4** act as Autosar Complex Device Drivers. Even the FlexRay and CAN drivers are individually standard in Autosar, the merged driver is for the reconfigurable hardware which is not supported by Autosar. Therefore, this driver will be implemented proprietarily as a Complex Device Drivers [11]. In case of a communication request, the communication services invoke the Complex Device Drivers instead of the communication hardware abstraction to communicate. Due to the fact that the considered driver is an extension of already existing Autosar-standard drivers (FlexRay and CAN), it can be implemented according to the Autosar standards, which will not force the designer to re-engineer all existing components. This complex driver is the fusion of 2 Autosar standard drivers (**Fig. 5**). Upon the reconfiguration request, the appropriate driver will be loaded to be suitable to the chosen communication protocol.

IV. Implementation and Results

We used a XC5VSX50T Xilinx Virtex 5 device. A PRR zone in the device floorplan was declared (**pr_box** in **Fig. 6**) and the implemented floorplan of the system is displayed in FPGA Editor (**Fig. 7**). There is a MicroBlaze [12] processor controlling the reconfiguration process of the dynamic zone. Inside the PRR, we can implement either the FlexRay or CAN IP and they can be switched on demand. The MicroBlaze searches the appropriate bitstream and sends it to the internal configuration port. For each protocol, the central processor loads the correspondent software stack to control the communication.

The implementation results are shown in **TABLE I**. The PRR was defined as shown in ***PRR*** column. Obviously the size of the PRR needs to be as small as possible but greater than the Flexray and CAN IPs size. Their required resources are shown in the second and third column respectively. The required resources CAN IP are much less than the PRR size (20% of LUT comparison) so about 80% LUT of the PRR are unused, but they cannot be reused for other purposes. The estimated reconfiguration durations (t_{switch} in the **Equation**

	FlexRay	**CAN**	**PRR**
6-pin LUT	4288	1125	5760
Flip Flop	3255	802	5760
SliceL	805	608	900
SliceM	483	350	540
DSP48E	1	0	48
RAMBFIFO36	6	2	24
Bitstream Size	284KB	195KB	
Reconfig. Time	592ms	406ms	

TABLE I
Implementation Results

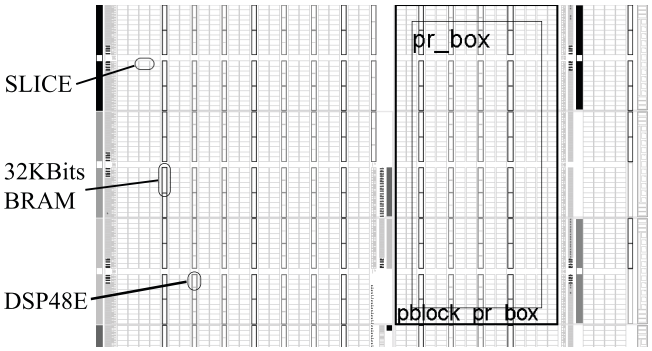

Fig. 6. System on a XC5VSX50T (displayed in Xilinx PlanAhead tool)

2) for the two modes are about 592ms and 504ms respectively (**TABLE I**).

V. Conclusion and Perspective

This paper presents a reconfigurable system applied in the automotive context for the support of flexible communication channels. We assume that there are only faults in the communication links. But the processor and the FPGA itself can also suffer electromagnetic attacks, this system is not completely fault-free. Possible solutions to deal with are to enhance the hardness of FPGA components or to design multi-processor systems increasing redundancy. The use of dynamically reconfigurable multi-processor systems being capable of reschedule system tasks according to environment changes would provide innovative and effective solutions.

The paper provides as well early concepts about integrating a dynamically reconfigurable architecture in Autosar environment. This supplies the first steps to bring the dynamicity into the automotive environment who is previously considered static. We also investigate the fusion possibility of all homologue modules in the architecture. The parameters alteration between communication modes can lead to the complexity growth of application software because the program at application level is responsible for switching between configuration sets. The possible impacts will be the program size and the timing overhead, which may decrease system performance. The most affected module should be the PDU Router because of various differences communication models. Upon the investigation results, we continue the idea of merging 2 static profiles into one so as to build a unified Autosar-conforming communication layer.

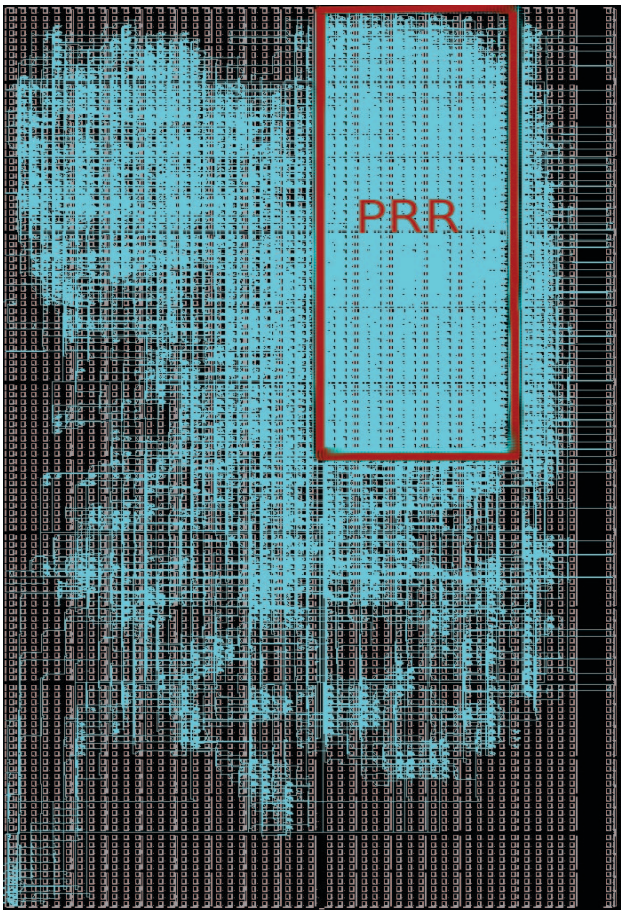

Fig. 7. Implemented floorplan of PRR (viewed in FPGA Editor)

References

[1] "http://www.insa-rennes.fr/ietr-cifaer."
[2] F. Nouvel, "Ad-Hoc Networking Combining Power Line and Wireless Communication for Automotive Embedded Networks," 2008.
[3] Xilinx, Inc, *Early Access Partial Reconfiguration User Guide UG208*, September 2008.
[4] "FlexRay Organization Consortium." [Online]. Available: http://www.flexray.com
[5] "CAN: Controller Area Network." [Online]. Available: http://www.can.bosch.com/
[6] AUTOSAR GbR, "AUTomotive Open System ARchitecture." [Online]. Available: http://www.autosar.org/
[7] ——, "AUTOSAR Specification of Communication v2.0.1," 2006.
[8] ——, "AUTOSAR Specification of PDU Router v2.0.1," 2006.
[9] ——, "AUTOSAR Specification of CAN Transport Layer v2.0.1," 2006.
[10] ——, "AUTOSAR Specification of FlexRay Transport Layer v2.0.1," 2006.
[11] ——, "AUTOSAR Technical Overview v2.0.1," 2006.
[12] Xilinx, Inc, "MicroBlaze Processor Reference Guide," *Embedded Development Kit EDK9.2i*, pp. 1–194, 2007.