On Demand Dependent Deactivation of Automotive ECUs

Christoph Schmutzler, Martin Simons
Daimler AG
HPC: 050/G007, 71059 Sindelfingen, Germany
Email: {christoph.schmutzler, martin.simons}@daimler.com

Jürgen Becker
Institute for Information Processing Technology (ITIV)
Karlsruhe Institute of Technology (KIT), Germany
Email: becker@kit.edu

***Abstract*—We describe details of a technology under development that allows selective deactivation of electronic control units in automotive networks as a means to increase a vehicle's energy efficiency: *intelligent communication controllers*. In particular, we provide details on an ICC's estimated energy savings potential, prove by experiment that ICCs are unique enablers for deactivation of FlexRay ECUs, and describe a prototypical implementation.**

I. Introduction

As energy efficient mobility becomes increasingly important even a vehicle's electronic systems have entered the scope of intense energy optimization efforts. In fact, as more and more functions and innovations of modern vehicles are electronics based, their energy consumption is no longer negligible. For the past couple of years, our group has developed concepts [1] as well as hardware and software technologies [2] to contribute to this industry-wide effort.

A vehicle's electronic systems operate in a coordinated manner governed by its *Electric/Electronic*-architecture (E/E-architecture). From an architecture point of view we found it helpful to distinguish three angles of attack when trying to optimize the energy efficiency of the electronic systems. First and most effective is the local, component-specific optimization of electronic control units (ECUs), actuators and sensors. Second is the selective deactivation of components at times when their function is not required. Last we see energy efficient computing techniques such as voltage or frequency scaling. Our work focuses on the second angle of attack: even though component-specific optimizations promise the greatest optimization potential they typically come at a price and a trade-off analysis may prove that it is more cost-effective to deactivate components when not in-use.

Deactivating individual components when they are interconnected in a complex network of cooperating systems is not straightforward. Two technologies are emerging to support the selective deactivation of ECUs in automotive networks: partial networking and intelligent communication controllers. Partial networking is a distributed concept and allows the activation and deactivation of virtual networks and their hosting components [2], [3]. We give a brief summary in Section II. The ECU-centric concept of intelligent communication controllers (ICCs) is the focus of this paper. An ICC-enabled ECU can assume a sleep mode that is transparent to the rest of the network, i.e., other nodes are unable to detect sleeping ECUs based on just network traffic. Partial networking and intelligent communication controllers are complementary and not mutually-exclusive in that they cater for different use cases. In addition, ICCs provide a solution for FlexRay-connected ECUs where the approach taken by partial networking does not work.

The organization of this paper is as follows: in Section II, we give an overview of different approaches to optimizing an E/E-architecture's energy efficiency and then outline the ICC concept. To substantiate the validity of the ICC concept, we compare in Section III our early theoretical estimations for an ICC's energy saving potential with concrete measurements. To substantiate our claim that FlexRay's wake-up mechanism does not fit the partial networking approach, we describe in Section IV an experimental analysis of commercially available FlexRay transceivers. Finally we outline in Section V our new prototyping platform which allows us to test FPGA-based IP blocks, such as an ICC, with regular microcontrollers that are being used in automotive development without impacting the application software. This allows testing of novel IPs using existing application software and target hardware at a very early stage. We conclude and outline our next steps in Section VI.

II. Optimizing Energy Efficiency of E/E-Architectures

Electronic components in modern cars, such as ECUs, and sensors and actuators, are interconnected by a complex heterogeneous communication network.

Depending on bandwidth, timing and safety requirements, CAN, LIN, MOST and FlexRay are typically used to connect ECUs. Ethernet is only just emerging. A good overview of the different bus technologies can be found in [4], [5].

Communication in most automotive networks is dominated by a cyclic communication paradigm: the vast majority of signals are sent with a defined cycle time, even if their value has not changed compared to the last transmission. Cycle times vary between a few milliseconds and several seconds, depending on the signal's criticality. Cyclic communication results in lower system complexity and increases robustness against temporary network failures or inadvertent resets of ECUs. Missing signal reception for several cycle times is treated as an error, handled by the receiving node.

A common drawback of today's automotive buses is that they only provide a reliable support for a bus wide sleep: as soon as any node starts to communicate, all sleeping nodes connected to the bus become active. They also do not provide mechanisms for an ECU-selective activation of sleeping nodes:

978-3-9810801-8-6/DATE12/©2012 EDAA

a node can only request a bus wide wakeup. Considering that today's vehicles are equipped with up to 70 ECUs and use over 6000 signals for communication between nodes, the network is active almost constantly, which in turn means that all ECUs are effectively always on if the vehicle is in use.

Given only traditional networking technology - in particular CAN and FlexRay - demand dependent deactivation of ECUs is thus next to impossible or very cumbersome. MOST and LIN are typically used for applications that are better suited for a bus-wide deactivation. In [2], we describe the available sleep mechanisms of the different bus systems and their advantages and disadvantages concerning energy efficient designs in more detail.

Although there are several bus independent mechanisms for an ECU-selective sleep, such as *clamps* and dedicated *wakeup lines*, none of them provide the necessary flexibility for a truly demand depended deactivation with a reasonable increase in system complexity and cost. Therefore, we have been working on concepts that allow for an ECU-selective sleep with existing bus system: for CAN, new *Partial Networking*-transceivers are being developed [1], [3], [6], [7]. A PN-transceiver can be put into PN-sleep mode, thereby deactivating the rest of the ECU. It now ignores regular bus traffic that would lead to an immediate wakeup with regular CAN transceivers. Other nodes can wakeup sleeping ECUs by sending a dedicated CAN frame. Hence, the use of PN requires major software changes throughout the architecture.

No similar transceiver based mechanism is feasible for FlexRay because of the much higher protocol complexity and bus speed. As an alternative to a transceiver based solution we therefore propose the concept of intelligent communication controllers. By extending the functionality of existing Communication Controllers (CCs), ICCs allow microcontrollers (μCs) to go to sleep if a static communication behaviour is sufficient. ICCs also do not require modifications of other nodes because ICC sleep mode is not detectable based on network traffic. Even when the rest of an ECU is not powered, an ICC can

- filter bus-traffic for wakeup reasons,
- store important messages that are required after the ECU resumes normal operation,
- and send static cyclic status messages.

This allows ECUs, which are temporarily not required based on the current state of the car, to sleep. We presented the ICC concept, plans for integration in the standardised automotive software architecture AUTOSAR [8] and an FPGA-based hardware demonstrator in [2], [9], [10]. Section V provides a more detailed description of the ICC concept.

III. Energy Saving Potential

The ultimate goal of deactivating individual ECUs when their function is not required is to conserve electric energy. The current rules of thumb that relate electric energy consumption with fuel consumption and CO_2 emissions are: reducing energy consumption by 1 kWh saves approx. 377 ml of petrol, which in turn lowers CO_2 emissions by 867 g [2], [11].

In the following two sections, we provide results of previous estimations and explain why we use a worst-case approach for our theoretical assumptions. We then compare the estimations with our measurements of ECUs that are suited for the use of an ICC.

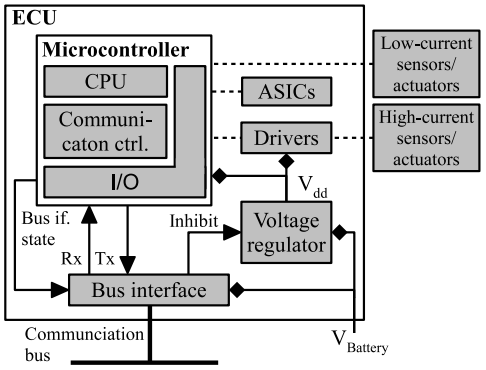


Figure 1. Simplified layout of an automotive ECU.

A. Theoretical Estimation

Until now, we used theoretical assumptions to approximate expected energy savings gained by a demand dependent ECU deactivation. For a reliable estimation of the saving potential, we restricted our estimations to worst case figures, i.e., the amount of power that can definitely be saved: assuming that an ECU is already 100% optimized, e.g., all components are deactivated the instant they are not needed, one can only save the current still required to continue communication (e.g., to power the μC, CC, and bus interface). We grouped commonly used μCs in different power categories. Depending on the category, the average supply current varies between 20 mA and 400 mA [2].

For a mid-level vehicle, we estimate that an average of at least 15 ECUs out of 42 ECUs total could sleep for 80% of the time during a regular driving scenario [2]. This results in savings of 3.96 W, reducing CO_2 emissions by 102 mg / km in the *Motor Vehicle Emission Group A* (MVEGA) driving cycle [12] that is used to rate fuel consumption of vehicles in the EU. Starting in 2015, even those fairly low values would equal a reduction of OEM penalty taxes by €9.72 for each car sold in the EU that emits more than 130 g CO_2/km.

Additional savings are likely considering the typical layout of an ECU, shown in Figure 1. A bus interface (e.g., CAN or FlexRay transceiver) connects the ECU to the communication bus. It converts analog bus signals into digital values and vice versa. A communication controller that de- and encodes frames is connected to the bus interface. In most cases, the CC is already integrated into the μC. Depending on current consumption, sensors and actuators are either connected directly to the μC, or are driven by dedicated power semiconductors or relays that are controlled by the μC. Especially peripherals, such as electric motors, valves, or lights require separate drivers. The μC might also be connected to separate ASICs designed for specialised tasks, for example signal or video processing. One or more voltage regulators supply the ECU components. The regulators are connected to the vehicles electrical system. Usually, at least one voltage regulator is controlled by the transceiver's *Inhibit-pin*.

Most ECUs only power off when bus communication stops. When the application software does not need to perform any local tasks, it switches the transceiver state machine into sleep

mode. The transceiver then disables its Inhibit-pin, which in turn powers off the μC and all other components. When the transceiver detects bus activity, it enables its Inhibit-pin and the μC starts again.

Depending on the task of an ECU, the number and type of peripherals - especially high-powered actuators - varies greatly. The resulting variance makes it difficult to give a precise savings estimate without a detailed analysis of each ECU. Compared to our theoretical assumptions, we expect the biggest additional power consumers to be idle peripherals, especially when supplied by power semiconductors, and ASICs.

B. Real world power consumption of ECUs

To further refine our worst case assumptions of the achievable energy savings, we measured the power consumption of ECUs that would be suited for the ICC concept in a current mid-level vehicle. To cover different vehicle states, the test drive included time in an urban environment, as well as a rural road and a highway section. The ECU measurements were performed by replacing the fuses for corresponding ECUs with special probes with an integrated shunt from Klaric GmbH. A separate probe was used to measure the supply voltage. The probes are connected to a KLARI-FUSE multi-channel voltmeter that transfers measurement data via CAN and RS232 [13]. Depending on the measuring range, the resolution varies between 14 and 50 μA/bit and 170 μV/bit, respectively,

Sleeping ECUs shall not result in different vehicle behaviour, e.g., a noticeable increase in reaction times upon user request. This can be ensured by a careful definition of deactivation and activation criteria. For the measured ECUs, one could, for example, use speed thresholds that forbid an opening of the trunk lid or a door.

Figure 2a shows the power consumption of the *trunk closure module* (TCM) that controls the automatic opening and closing of the trunk lid. Assuming that the ECU is not required above a speed of 18 kph, the deactivation of the ECU would save 68% of energy. We highlighted the time where the TCM could sleep. Since the battery is not fully charged at the start of our test drive, charging management raises generator voltage at higher speeds until the target capacity is reached. This explains the variations in power consumption in the first minutes.

In Figure 2b, we show the power consumption of the *door control module* and the battery voltage. After 3 minutes, battery charging is finished. Apart from several peaks that occurred when raising or lowering the windows, power consumption is nearly constant. Again, we highlighted to amount of energy that can be saved by a deactivation above 18 kph.

As expected, other ECUs also show an almost constant average power consumption that is virtually independent of the current state of the car. Table I shows their average, minimal and maximal current consumption, as well as the designated μC power class. The door and seat control modules show high maximum consumption figures when actually adjusting the seat, or using power windows.

Figure 2c shows the accumulated average power savings during the MVEGA driving cycle. Over the test duration of 1180 s, 2.8 Wh could be saved when using the deactivation criteria from Table I. For just five ECUs, this equals a CO_2 reduction of 219 mg/km, lowering OEM penalty tax by €20.76 - 213% of our worst case assumptions that considered all ECUs of a car!

IV. Experimental evaluation of the FlexRay wakeup mechanism

As discussed in Section II, ECUs typically use the transceiver to power off: the μC switches the transceiver state machine into sleep mode to disable its Inhibit-pin. This switches off a connected voltage regulator that stops powering the ECU components. If the transceiver detects a wakeup event, it enables its Inhibit-pin and the ECU is powered again.

FlexRay defines a special wakeup pattern that can be used to trigger a bus wide wakeup of all sleeping transceivers. In principle, this mechanism is roughly comparable to Partial Networking for CAN and could be used for a selective shutdown of ECUs: if a node is not required based on the current vehicle state, it could inform the network about its pending sleep state, and then power down. If other nodes require the ECU, they could trigger a bus wide wakeup and then inform previously sleeping nodes whether they are required or not. If wakeups don't occur to frequently, ECUs could sleep inbetween requests, thereby saving energy.

FlexRay does, however, not explicitly forbid false wakeups, e.g., caused by an erroneous detection of the pattern by the transceiver. The experimental analysis of the wakeup mechanism discussed in the next sections shows, that the number of unwanted wakeups is very high.

A. FlexRay wakeup pattern

The FlexRay wakeup pattern consists of at least two wakeup symbols. To form one symbol, the sender has to transmit a low level for 6 us followed by an idle phase of 18 us.

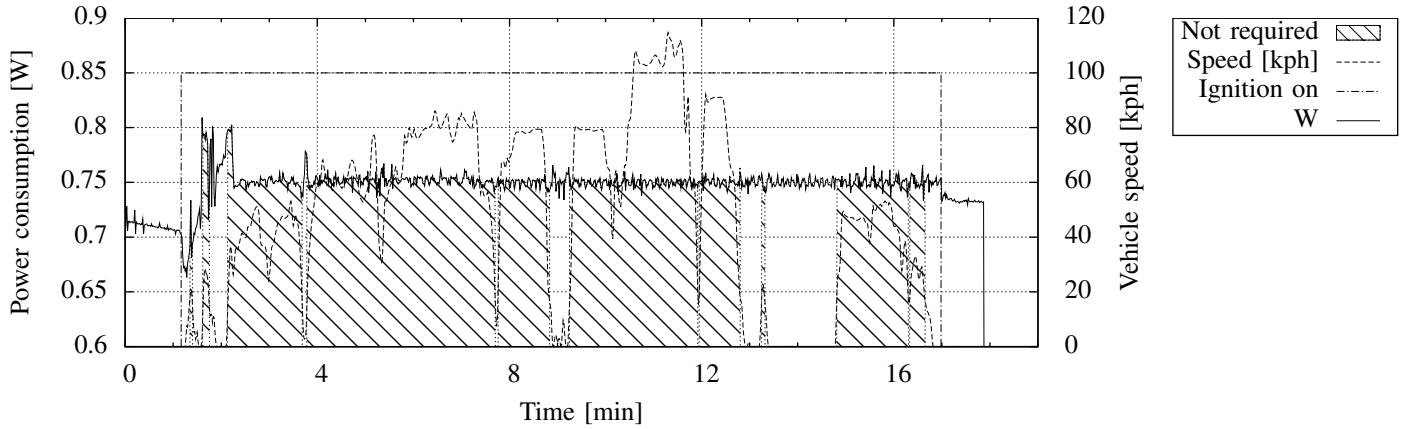
Even if two senders transmit a wakeup pattern simultaneously or in short succession, the resulting pattern always consists of a low, idle, low, idle sequence. Each low and idle phase is guaranteed to have a minimal duration of 4 us, the total duration of the two symbols is always less then 48 us [14, pp. 17]. Collision avoidance mechanisms ensure that no more than two patterns can overlap each other.

The main goal of the FlexRay specification is to provide a reliable bus wakeup mechanism, but not an avoidance of false wakeups: it does not prohibit a wakeup event if the pattern has not been explicitly sent, but was detected in regular bus traffic. It also accepts variations in length and glitches during the low or idle phases. Furthermore, the timer used by the pattern detector must work in very large temperature ranges. On the receiver side, the wakeup pattern is, therefore, allowed to be as long as 140 us, the minimal length of the idle and low phases may vary between 1 to 4 us.

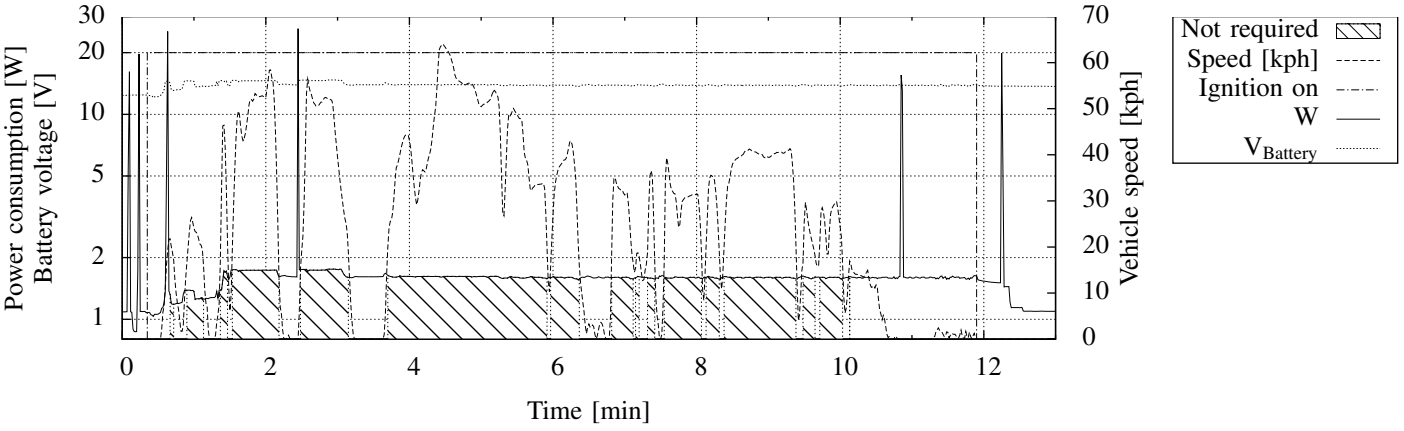
B. Test setup and results

To test whether the existing wakeup mechanism could still be used for a selective sleep of single ECUs, we measured the probability of false wakeups during normal communication. If the probability is low, the mechanism could be used in a similar manner as Partial Networking for CAN.

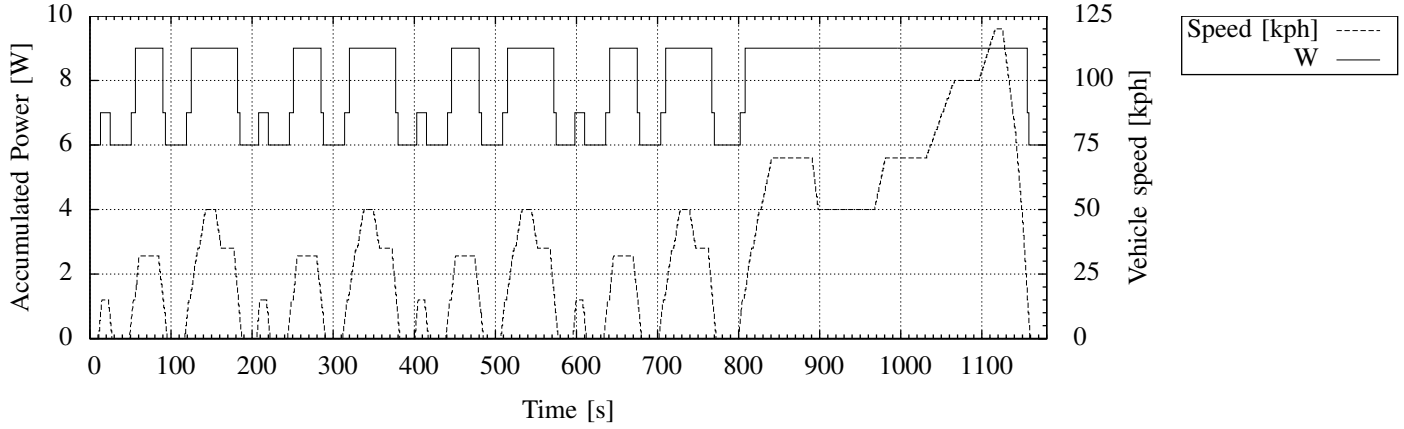
We use *Vector's CANoe* network analysis, simulation and testing tool [15] to replay a bus trace that was recorded in



(a) Power consumption of the trunk closure module.



(b) Power consumption of the door control module.



(c) Accumulated average savings of the five measured ECUs during the MVEGA driving cycle.

Figure 2. Measured power consumption and periods where the ECUs are not required and could be disabled.

Table I
MEASURED POWER CONSUMPTION DURING TEST DRIVE

ECU	Supply current [mA]			Assigned μC power class	Possible deactivation criteria
	Min.	Max.	Avg.		
Trailer control module	13.1	25.4	17.91	low (7 mA)	no trailer attached, speed > 8 kph
Radio control module	265.7	362.36	312.8	high (70 mA)	Radio off
Seat control module	148.7	1693.5	170	mid. (20 mA)	seat not occupied
Trunk control module	51.6	56.2	53.8	mid. (20 mA)	speed > 18 kph
Door control module	85.9	1161.9	121.39	mid. (20 mA)	according seat not occupied, door closed, speed > 18 kph

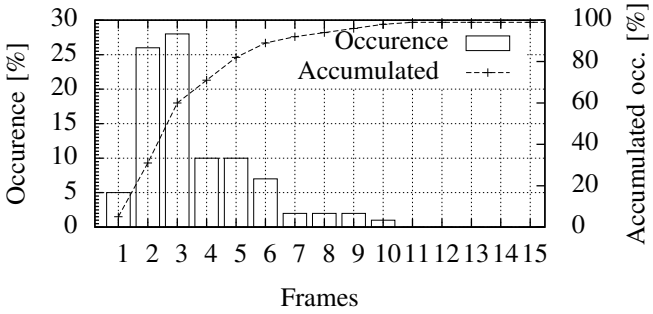


Figure 3. Distribution of received frames until wakeup.

a FlexRay based vehicle. CANoe is running on a regular PC and uses an USB-FlexRay interface to connect to the bus.

To measure how often a sleeping transceiver wakes up, we use an Altera Stratix III FPGA that is connected to two NXP TJA1080 transceivers A and B. Both transceivers are connected to the same channel. Transceiver A is always active, transceiver B is put into sleep mode as soon as it becomes active. A custom capture block on the FPGA analyses bus traffic via transceiver A and counts received frames while B is sleeping. Once transceiver B issues a wakeup, we store the total number of frames that were received while B was asleep, including the wakeup frame. B is then put into sleep mode again.

Figure 3 shows the distribution of the number of received frames until B issued a wakeup. On average, a wakeup is triggered every 2.79 frames. At most, 29 frames were received. The resulting average sleep duration is less then 600 us.

We performed a similar test with ECUs that were equipped with a NXP TJA1080 and an austriamicrosystems AS8221 transceiver at room temperature. Since we can not record the number of received frames while the transceiver is asleep, we measured the number of wakeups over the total test duration. On average, the NXP transceiver triggered a wakeup every 7.2 frames, the AS8221 woke up every 7.8 frames. The results confirm the high number wakeups of our FPGA-based measurement. Please note those values also include the time required to go to sleep again, hence the lower average wakeup rate compared to the FPGA measurements.

In summary, the transceiver based wakeup mechanism has a high probability of triggering false wakeups. This means that the existing FlexRay wakeup mechanism is not suited for a demand dependent deactivation of ECUs and can not be used in a similar manner as Partial Networking for CAN. Therefore, it is no viable alternative to our ICC concept.

V. ICC Test Platform

The goal of our ICC concept is to enable ECUs to go to sleep when they are not required, even if the network is still active. In contrast to Partial Networking, software changes shall be limited to the node that uses the ICC. This results in two main requirements:

1) the ICC must be able to detect wakeup events without an explicit request sent by another node,
2) and the ICC shall be able to send data when the rest of the ECU is sleeping.

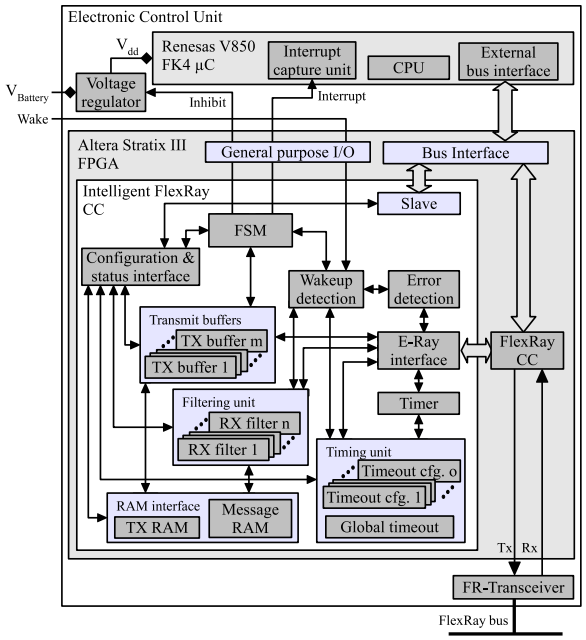


Figure 4. Test setup: the FPGA is connected to the FK4 via the MEMC interface. The application running on the μC can access the ICC implementation without further software modifications.

The first requirement can be met by providing a configurable signal filtering mechanism that monitors network traffic for wakeup reasons, such as a certain speed range, a pressed button, the ignition state, or the absence of a signal for a certain time. An ECU that sends changing signal values still requires CPU operation and can not go to sleep. Only if the send behaviour is static, the CPU can be deactivated. The second requirement can, therefore, be restricted to static data.

The structure of our prototypical ICC implementation is shown in Figure 4. The main components are the transmit buffers, the filtering and timing units and the interface to an internal memory. The memory contains the configuration of the receive and timeout filters, as well as the payload and cycle time of messages that are to be sent. It is also used to store messages that have passed the receive filters. This reduces wakeup reaction time because required messages are available immediately after resuming operation. The number of transmit buffers, receive and timeout filters are configurable. A more detailed description of the prototype can be found in [10]. For communication, we use Bosch's *E-Ray* FlexRay CC [16]. The E-Ray is also used by the CPU during normal operation.

For future commercial implementations, we expect the ICC to be integrated in an μC and not to be realised as a separate ASIC. In this case, all μC components except for the ICC and CC can be put into an energy efficient state, for example by power or clock gating mechanisms, when the ICC is active. Apart from the transceiver and the μC's voltage supply, the rest of the ECU can be powered off. Most μCs with integrated FlexRay controllers are usually high-performance devices, with an average supply current of 200 mA to up to 400 mA. Even though the ICC and transceiver must still be powered, we therefore expect the remaining energy savings to be very close to the figures provided in Section III, if not higher compared to mid-grade μCs.

In the past, we implemented and tested the ICC using a System-on-Chip (SoC) design implemented on a FPGA. The SoC included Altera's Nios II softcore CPU, which executes software that mimics the behaviour of a regular FlexRay node. The setup can be used to test the ICC implementation, as well as ICC-related software configuration and management mechanism in an existing network infrastructure.

Even though this approach is well suited to evaluate the hardware implementation, it does not allow for a comprehensive concept validation. In particular, it does not allow us to use:

- a standardized software stack, i.e., a common embedded OS used on all ECUs,
- existing configuration tools that are used for automated software generation, especially the networking code,
- and commercial μCs used in ECU development.

This makes it hard to integrate and test the ICC in existing application software, since the complete stack needs to be ported to the FPGA. The changed hardware also dilutes parameters, such as reaction time on wakeup events, which are important to design and calibrate the application software.

To address these shortcomings, we developed a new test platform that allows us to use our FPGA-based ICC implementation with the high-performance V850E2/FK4 μC from Renesas Electronics [17]. The FK4 supports an external memory controller interface (MEMC). The MEMC interface uses a multiplexed *address/data* bus with variable data width, and dedicated *write*, *read*, and *waitrequest* lines. The MEMC interface can be accessed via a designated address space.

As shown in Figure 4, the FK4 MEMC interface is connected to an Altera Stratix III FPGA (EP3SL150F1152C2). A custom wrapper block is used to connect the MEMC bus to the FPGA's on-chip bus. This allows us to directly access IP blocks on the FPGA from the FK4. From a software perspective, it is not noticeable whether a component is located on the μC, or on the FPGA.

The FPGA is also connected to a flash memory that is used to configure the FPGA after power on, a JTAG debug interface and a 2MB SRAM. The PCB design and layout was developed by Renesas Electronics. The setup allows us to integrate the ICC into our standard embedded OS (AUTOSAR), and into existing application software without the need to port the software to a different platform. It also means we can measure and evaluate performance figures, such as reaction times, or required software resources for ICC configuration and management using the target hardware. In summary, this allows us to perform a complete validation of the ICC concept.

VI. Conclusion and Outlook

Today's automotive bus systems do not allow for a selective deactivation of ECUs when the car is in use. This severely limits demand driven energy management strategies for electronic based consumers in future vehicles. In this article, we elaborated on the concept of an intelligent communication controller (ICC) which we are developing to allow selective activation and deactivation of FlexRay-connected ECUs.

To validate our concept, we compared previously published theoretical saving estimates that only incorporate the power an ECU requires for communication, with our measured power consumption of suited ICC candidates in a current mid-level vehicle. The resulting figures prove the benefit and potential of a demand dependent ECU deactivation. We have also shown that the existing FlexRay wakeup mechanism can not be used for a reliable sleep mode while the bus is active, and is, therefore, no alternative to ICCs. Lastly, we outlined our new experimental platform that allows us to use prototypical, FPGA-based IP implementations with regular μCs that are used for regular ECU development.

Based on the new FK4-FPGA testing platform, we currently develop a demonstrator using an application software adapted for the use of an ICC. We are planning on integrating the demonstrator into a pre-production vehicle to verify ICC behaviour, and to explore influences on the user experience. Safety related aspects that need to be taken into account when using an ICC are currently under consideration. We are also planning to extend the development tool chain to include support for an automated generation of ICC configuration.

References

[1] T. Weber, V. Lauer, D. Mann, and M. Simons, "The Comprehensive Energy Management - From the Conventional Powertrain to the Full e-Drive," in *4. VDI-Tagung: Baden-Baden Spezial 2010 Elektrisches Fahren machbar machen*, 2010.

[2] C. Schmutzler, A. Krüger, F. Schuster, and M. Simons, "Energy effcient automotive networks: State of the art and challenges ahead," *International Journal of Communication Networks and Distributed Systems, Special Issue on Optimization Issues in Energy Efficient Distributed Systems*, 2012, to be published.

[3] B. Elend and H. Huber, "Reduktion des CO_2 Ausstoßes durch Teilnetzbetrieb in der Fahrzeugvernetzung," in *2. Elektronik automotive congress*, May 2010.

[4] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1223, 2005.

[5] D. Paret, *Multiplexed networks for embedded systems : CAN, LIN, Flexray, Safe-by-Wire...* Wiley, 2007.

[6] M. Fuchs, P. Scheer, and A. Grzemba, "Selektiver Teilnetzbetrieb im Fahrzeug: Eine Realisierung für den CAN-Bus und Adaption auf andere Bussysteme," in *AmE 2010 - Automotive meets Electronics*, 2010, pp. 15–18.

[7] R. Weber, R. Watroba, and C. Schmitz, "CO_2-Reduktion durch CAN-Teilnetzbetrieb," *ATZ elektronik*, vol. 2, 2011.

[8] *Automotive Open System Architecture*, www.autosar.org, AUTOSAR, 2003, www.autosar.org.

[9] C. Schmutzler, A. Krüger, M. Simons, and J. Becker, "Ansätze zur Integration von energieeffizienten Intelligenten Kommunikationskontrollern für FlexRay in Autosar," in *Informatik 2011 - 41. Jahrestagung der Gesellschaft für Informatik*, ser. Lectures Notes Informatics (LNI), Oct. 2011.

[10] C. Schmutzler, A. Lakhtel, M. Simons, and J. Becker, "Increasing Energy Efficiency of Automotive E/E-Architectures with Intelligent FlexRay Communication Controllers," in *System on Chip (SoC), 2011 International Symposium on*, Nov. 2011.

[11] C. Koffler and K. Rohde-Brandenburger, "On the calculation of fuel savings through lightweight design in automotive life cycle assessments," *The International Journal of Life Cycle Assessment*, vol. 15, pp. 128–135, 2010.

[12] *Emission Test Cycles for the Certification of light duty vehicles in Europe, EEC Directive 90/C81/01*, European Union, 1999.

[13] *KLARI-FUSE*, Stefan Klaric GmbH & Co. KG. [Online]. Available: http://www.klaric.de/text/produkte/klari-fuse-de.html

[14] *FlexRay Communication System, Electrical Physical Layer Specification Version 2.1B*, www.flexray.com, FlexRay Consortium, Nov. 2006.

[15] *CANoe*, Vector Informatik GmbH. [Online]. Available: www.vector.com/vi_canoe_en.html

[16] *Product Information E-Ray IP Modul*, Robert Bosch GmbH, Jul. 2009. [Online]. Available: www.semiconductors.bosch.de/media/en/pdf/ipmodules_1/flexray/bosch_product_info_eray_ip.pdf

[17] *V850E2/FK4 Microcontrollers*, Renesas Electronics Corporation. [Online]. Available: http://www2.renesas.com/micro/en/product/subseries.php?subseries=V850E2%2FFK4