Energy Saving Potential of Adaptive, Networked, Embedded Systems - A Case Study

Patrick Heinrich[1], Erik Oswald[1] and Rudi Knorr[1,2]

[1]Fraunhofer Institute for Embedded Systems and Communication Technologies ESK, Munich, Germany
[2]Chair for Communication Systems, Institute of Computer Science, University of Augsburg, Germany
E-Mail: forename.surname@esk.fraunhofer.de

***Abstract*—This paper presents and evaluates the energy saving potential of adaptive, networked, embedded systems. The aim is to demonstrate the benefits of modeling the energy demand during the development of such systems. For this purpose, the previous developed energy model is applied within a case study and different allocations of software components are compared. The estimated energy demands of these allocations are presented and discussed. The analyzed system of the case study represents an automotive system which executes two advanced driver assistance applications. The system is adaptive, which means that temporally unnecessary applications will be deactivated. Within the evaluated system this deactivation depends on the vehicle speed, which is derived by the New European Driving Cycle. Two different allocations of software components are evaluated. One represents the today's allocation, the other an energy-focused allocation. The case study shows an energy saving potential of about 18 %.**

Keywords–Embedded systems; Adaptivity; Networked systems; Energy estimation; Automotive; Case study.

I. Introduction

Networked, embedded systems of modern cars consist of up to 80 independent electronic control units (ECUs), which cooperate to enable complex applications. These systems are characterized by a high degree of interaction and consist of different (specialized) communication networks [1]. Future cars will probably enable autonomous driving, which further increases the number of ECUs and the interaction between these ECUs. However, future automobiles will also be adaptive to decrease the energy demand by deactivating temporarily unnecessary applications. Hereby, the chosen allocation of software components on the network's ECUs is relevant w.r.t. the energy demand [2]. This is caused by different applicable energy saving states and energy demand necessary to (de)activate hardware and software. The allocation is done by system designers relatively early within the development process [3], which makes it necessary to estimate the later energy consumption at that time using the available information.

In today's luxury-class vehicles for instance, the electrical and electronic components draw up to 2.5 kW [4][5]. Compared to what the vehicle engine requires (e.g., 55 kW), 2.5 kW seems small. However, the electrical components consume energy during every mode of operation, even when in standby mode. The vehicle engine consumes most of its energy during acceleration and even here the maximum power is seldom demanded. Communication and sensors/actuators cause an increasing amount of energy consumption of networked embedded systems. An increase of 100 Watt thus means that fuel consumption rises by 0.1 liter per 100 km, leading to an increase in CO_2 emissions of 2.5 gram per km [4]. This illustrates the considerable relevance for energy savings, an aspect that should be factored in during the development.

This paper presents an automotive case study to demonstrate the energy saving potential, which is usable when the energy demand of such systems is modeled previously to the allocation of software components. In Section II, the previous work and in Section III the characteristics of the system are presented on which the case study is based. The case study is described within Section IV. In Section V and VI, the energy demand estimation is presented and discussed. Finally, Section VII concludes the paper.

II. Related Work

The energy models used to estimate the energy demand within this case study are based on previous work, which has focused the various elements of adaptive, networked, embedded systems. The necessity to model the energy demand of adaptivity and the uncommon fact that suboptimal (w.r.t the energy demand) allocations of software components per system state may form an optimized adaptive system was described within [6]. Different approaches to estimate the energy demand previous to system integration, e.g., during development of the system, were presented within [7]. Based on this work the further developed models focused on the energy demand estimation during the development phase where the software components are allocated within the networked system. This enables a good trade-off between the inaccuracy of early energy estimations and the available energy saving potential of the later system – which is reduced after every decision within the development process [7]. Within [8] a model is presented which enables the energy demand estimation of software components executed on embedded systems. The estimation is based on program flowcharts and enables an accuracy between -11.9 % and +6.9 %. Embedded systems (ECUs) including sensors/actuators and offset energy demand were modeled within [9]. This also includes the effects of parallel execution of software components on the energy demand of adaptive ECUs. This work focused on aspects of adaptive systems where temporarily unnecessary software components are deactivated. The energy demand of communication between embedded system is modeled within [10]. These communication networks

$$E_{total} = \underbrace{\underbrace{\sum_{i=1}^{\#ECUs} \left[\overbrace{\sum_{j=1}^{\#SWCs} (E_{flow_j} \cdot \tau_{corr})}^{\text{software component}} + E_{SensAkt_i} + E_{ECU_i} \right]}_{\text{adaptive, embedded system}} + \overbrace{E_{com}}^{\text{communication}}}_{\text{adaptive, networked, embedded system}} \tag{1}$$

are characterized by heterogeneity and a high degree of interaction between the specialized parts of the network. The presented model is based on individual communication connections which significantly simplifies the energy demand estimation of adaptive systems. The model enables estimations within an error range of -2.4 % and +2.8 %, which is very accurate.
Equation (1) summarized the presented parts of the energy model. The energy demand of the system is represented by E_{total}. The networked system consist of a number of ECUs ($\#ECUs$) which communicate via communication networks demanding the energy amount represented by E_{com}. The energy demand to execute software components on an ECU is represented by the number of software components ($\#SWCs$), the energy demands of the program flowcharts (E_{flow}) and the correction factor τ_{corr} (cf. [8]). The symbol $E_{SensAkt}$ represents the energy demand of sensors and actuators, the symbol E_{ECU} the offset energy demand of ECUs. The detailed calculation of the symbols is presented within the considered previous work.

In the following section, the characteristics of adaptive, networked, embedded systems are presented.

III. Adaptive, Networked, Embedded Systems

Networked, embedded systems as found within the automotive sector are characterized by a high degree of heterogeneity [11]. This means different kind of ECUs by various suppliers and different types of communication protocols and topologies resulted by specialized technologies. A common hardware architecture of networked embedded systems is shown within Figure 1. Additionally, these systems are characterized by a high degree of interaction, which means that applications (which are experienced by the user) are spread over various ECUs within the networked, embedded system. These individual components are commonly used for more than one application, i.e., the rain sensors is used to control the windshield wipers and also to parametrize the Electronic Stability Program (ESP). Adaptivity within embedded systems enables the deactivation of temporarily unnecessary functionality including its hardware. This enables the design of more energy efficient systems, but significantly enlarges the number of possible system designs. Furthermore, factors such as time and energy for (de)activation and availability of different kind of sleep modes are relevant now. Adaptivity necessitates considering the whole systems instead of just looking at individual components, because it is

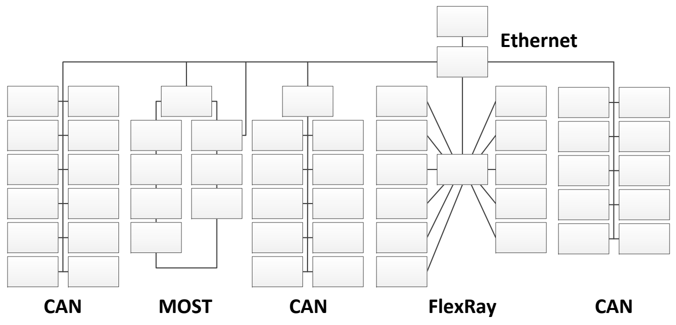

Figure 1. Typical networked embedded system (BMW 7 series) [adapted from BMW 2005, quoted from [12]]

possible that individual suboptimal systems form together a more energy efficient system [6].

Embedded systems within the automobiles commonly execute safety-critical applications, which results in real-time constraints, for example concerning transmission and response times. System designers of networked, embedded systems are also faced with a lot of other constraints and limitations (e.g. performance and memory limitations, timing constraints and heterogeneity). The energy demand is just one of the aspects system designer have to consider. However, this aspect has become increasingly important over the last few years.

In the following section, details concerning hardware and software architecture and the adaptivity of the system considered within the case study are presented.

IV. Case Study

Within this section the advanced driver assistance systems considered in the case study are explained and the resulted software architectures are presented. Afterwards, the considered hardware architecture including the energy relevant parameters are discussed. And finally the adaptivity of the system and the relevant system context is pointed out.

A. Advanced Driver Assistance Systems

The two advanced driver assistance systems which software components are allocated within the networked system are Adaptive Cruise Control (ACC) and Automatic Parking Assist (APA), which are explained in the following.

1) Adaptive Cruise Control: This system automatically adjusts the vehicle speed to maintain a given distance from a vehicle ahead. The distance is measured using sensors like radar sensors. To enable such a functionality different electronic control units (ECUs) have to interact, e.g., within an AUDI A8 80 ECUs are part of this system [13]. An ACC system consist of different components (cf. [14][13][15] [16][17]). An object recognition is used to detect vehicles ahead and determine the absolute speed. Therefore long-range and short-range radar sensors are used. Additionally, the data is used to predict the probable course of the own vehicle to avoid the detection of irrelevant objects. To do so, additional input from a path and curve calculation is necessary, which uses data from yaw rate and steering angle sensors. The main components of ACC are the control loops to control the vehicle speed and to follow the vehicle ahead. Depending on the final implementation, the calculated required torque or deceleration is transmitted to engine or brake control. Figure 2 represents the derived software architecture of the ACC system as it is used within the case study. A long-range and a short-range radar including the signal processing are used to recognize objects (vehicles) ahead. Sensors for wheel speed, yaw rate and steering angle are used to analyze the current and future status of the own vehicle (including path and curve calculation and course prediction). This information is used to enable the control loops, which are summarized within "ACC Control". Engine and brake control execute the calculated acceleration or deceleration. The arrows – refined with the number of bytes transfered – represent communication.

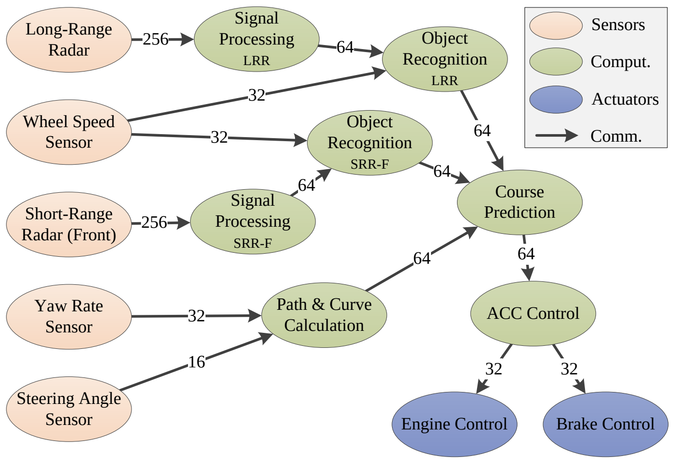

Figure 2. Software architecture of Adaptive Cruise Control

2) Automatic Parking Assist: This driver assistance system enables a driver to automatically park into suitable parking spaces (cf. [18]). Short-range radar or ultrasonic sensor at front and rear are used to measure the distance between the vehicles. The system controls engine, brake and steering wheel to maneuver and park the vehicle automatically. Sensors for wheel speed, yaw rate and steering angle are used to monitor the vehicle's parking operations. Figure 3 shows the derived software architecture of the automatic parking assist system. Two radar sensors including signal processing and object recognition observe front and rear of the vehicle. Wheel speed, yaw rate and steering angle sensor are additionally used to control the parking assistant. Engine, brake and steering control execute the calculated acceleration, deceleration and steering.

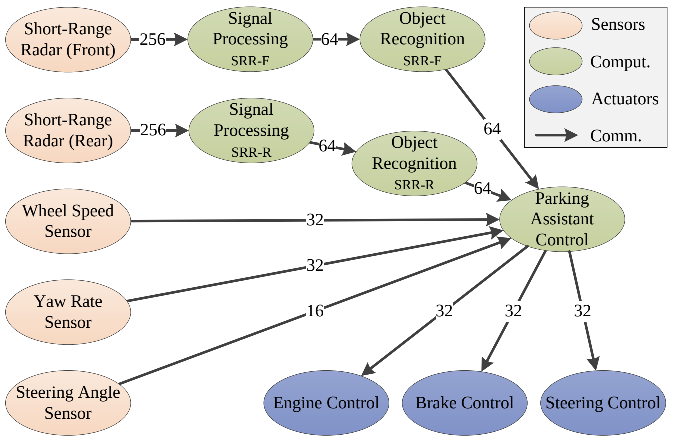

Figure 3. Software architecture of Automatic Parking Assist

Within this case study the driver assistance systems Adaptive Cruise Control and Automatic Parking Assist are used to represent an automotive system. These systems are real-time critical, which results in a suitable high control frequency. Hansson et al. [19] define the control frequency of ACC with 10 Hz, which represents an response time of 100 ms. This value is used as cycle time of the ACC's and APA's software components. Furthermore, some of the software components are represented within both applications. If ACC and APA are active at the same time, the identical software components are executed just once and provides the required information to the following software components of both applications. Table I presents the relevant parameters of the software components, which i.a. are derived from [19] and [20]. The relevant parameters are: execution time, cycle, time, deadline and produced (data) output.

In the following subsection, the hardware architecture and the energy relevant parameters of the system are introduced.

B. Hardware Architecture and Energy Parameters

Within this subsection, the properties of the system are presented. That means in detail, ECUs including sensors and actuators, software components and adaptivity. The hardware architecture of the case study is based on the networked, embedded system of a BMW 7 series (as shown in Figure 1). The chosen hardware architecture is presented within Figure 4, where the topology is similar but the number of ECUs is reduced. The case study's validity is not influenced through that reduction, because grouping software components is focused by both considered "allocation styles" (cf. Subsection VI-A). Equal distribution of software components normally increases the energy demand, caused by the offset energy demand of ECUs, which is independent of the number of software components running on an ECU.

Within the case study just one ECU is defined, which is equipable with different sensors and actuators. The reason for this is that ECUs with different energy efficiencies would concentrate the software components on the most energy

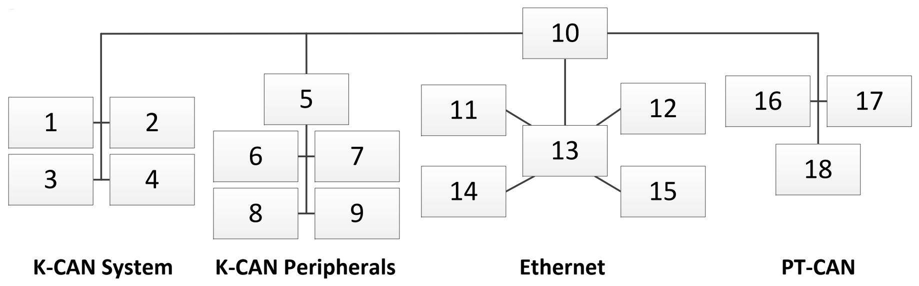

Figure 4. Simplified network topology of the case study, based on the networked, embedded system of the BMW 7 series (cf. Figure 1). Note: The numbers are used to identify the ECUs during allocation of the software components.

efficient ECUs. However, the purpose of this case study is to demonstrate the energy saving potential of adaptive, networked, embedded systems. Different ECUs would increase complexity without an advantage. Therefore, just one energy saving state of the components is defined. E.g. the µController has three states – running, idle and sleep. The characteristics of the used µController are derived from existing µControllers like the "MPC5674F" [21] of Freescale Semiconductor, which works at 264 MHz and has a power demand of 850 mA at 1.32 Volt. A current consumption of 100 mA during idle operations is assumed. The activation time depends on the phase-locked loop (PLL) lock time and was specified with 400 µs. The time for deactivation is assumed to bei 50 µs. This results in an energy demand of 0.5 mWs per deactivation including activation of the µController. The offset power demand of the defined ECU is specified with 1.5 W, which includes the energy demand of components such as the power supply unit and the voltage regulators. Weber et al. [22] have defined a Body Control Module with a power consumption of 10.75 W, where 1 W is used for the µController and 4 W is defined as ECU offset power consumption. The ECU offset power consumption within the case study is defined at the lower range of possible values, because the full capacity of the defined system is not used by the two driver assistant systems. Through that a falsification of the percentage ration of the energy demand within the system is prevented. The effect on other components is low, because most scale with the utilization of the ECU. (This is not the case for the states idle or sleep. However, the energy demand is lower and through that estimations errors have a lower impact.)

Due to works which try to reduce the energy demand of sensors and actuators (such as Benbasat [23]) the activation time is assumed to be much smaller than the cycle time of the used sensors and actuators. That means these components are deactivateable after the software component execution. (This is not the case for short-term interruption caused by scheduling.) This is another reason why activation time of sensors and actuators are necessary to reach more accurate energy demand estimations. Radar systems within the automotive area have a power demand of about 4 to 4.5 W (cf. [24] and [25]). The long-range radar used within this case study is defined to have a power consumption of 4.5 W and an activation time of 25 ms. The short-range radar consumes 1.75 W of power and need 25 ms to get active. The power consumption of "Engine-", "Brake-" und "Steering-Control" are derived from the Automotive Engine Control IC [26] of Freescale Semiconductors. (Note: Components such as relays, servomotors, spark plugs, etc. are not concerned within this work. That is why just the control unit is considered.) The Engine Control IC has a voltage range of 4.5 to 36 V with a maximum current of 14 mA. The power demand is defined to be 168 mW (12 V, 14 mA) and the activation time to be 1 ms. Rotations speed sensors such as "KMI17/4" [27] of NXP Semiconductors are defined with a turn on delay time of 1 ms. (This value is also used for the wheel speed and the angle sensor.) The energy demand is defined with 120 mWs, because a voltage of 12 V is assumed, which results a current of 10 mA. Angle sensors such as "KMA220" [28] of NXP Semiconductors need a voltage supply of 5 V and have a current demand of 10 to 21 mA. Through that, the "Steering Angle Sensor" of this case study is assumed to have a power demand of 75 mW. The properties of the used yaw rate sensor is derived by the combined inertial sensor for vehicle dynamics control "SMI650" [29] of Robert Bosch GmbH. The data sheet specifies a necessary voltage supply of 5 V and a current demand of 25 mA. This results in a power demand of 125 mW. An activation time of 650 ms is specified, which includes a detailed self test. It is assumed that the self test is just relevant during the initial start of the system and not after every sleep state. Through that an activation time of 5 ms is defined for the case study. Table II shows the derived parameters concerning the energy demand of sensors and actuators including the activation time. (Note: This work focuses on the electronic system, e.g., the components like the engine, which is controlled by "Engine Control" is excluded.)

The energy relevant parameters of the communication networks are derived from the measured values at [10]. The parameters for Ethernet are derived from data sheets. The CAN network within the case study uses a data rate of 500 kbit/s. Due to the specification [30] eight bytes per message are user data and 44 bits are communication overhead. The energy relevant parameters of the "High-Speed CAN Transceiver MCP2561" [31] and of the "Stand-Alone CAN Controller with SPI Interface MCP2515" [32] are used. The Ethernet network works with a data rate of 10 Mbit/s. A maximum of 1500 bytes per message are user data and 144 bits are communication overhead. The Ethernet transceiver "LAN8710A" [33] and the Ethernet controller "CS8900A" [34] were served as basis to derive the energy relevant parameters. However, a more efficient sleep state was assumed. Table III summarizes the energy relevant values of the used communication networks.

Within the work which models the energy demand of embedded communication networks [10] a energy demand to transfer data between networks is defined as "transfer energy". This energy demand depends on the used µController and the network on both sides of the gateway. Identical networks result a smaller energy demand, than different networks, e.g., because of the need to split user data or handle different data rates. Within the case study a energy demand of 0.12 mWs per message is defined and a transmission time of 0.1 ms. If the gateway is a specialized router or switch, than half of the energy demand is assumed. Caused by unproven values, the energy parameters are estimated at the lower range of the possible values to prevent distortions.

The energy demand necessary for the system's adaptivity is determined by the (de)activation of software components and ECUs. The energy demand to activate or deactivate a software component is determined to be 5.61 mWs. This value is estimated by the summarized value of 5 ms for self test, safe/load data, etc. and the power consumption of the used µController of 1122 mW. The (de)activation of entire ECUs is defined with the energy demand of 2.5 Ws. This value is resulted from the power consumption of a Body Control Modul [22], which has a power demand of 10 W and an activation time of 250 ms. An activation time of 100 to 200 ms is necessary to boot up the ECU and the additionally need to receive all necessary messages to get the actual data to work with [35]. This results in the value of 2.5 Ws.

Within the following subsection the characteristics of the system concerning adaptivity are presented.

C. *Adaptivity*

The analyzed system within this case study is adaptive as presented within Section III, i.e., deactivates temporarily unnecessary hardware and software components. (The necessary energy demand was already presented within the subsection before.) The need of a specific application arises by an user request or the actual context (environment) of the vehicle. For example, windshield wipers are just necessary, if there is rain or other kind of water at the windshield, or an automatic parking assist is not necessary at high speed.

The analyzed driver assistant systems within this case study are activated depending on the current speed of the vehicle, i.e., the relevant system context is "Vehicle Speed". This input is relevant to analyze the energy demand of the system. Within this case study the very common New European Driving Cycle (NEDC) is used to derive the speed profile. The NEDC is used to measure a compareable fuel consumption of vehicles by the manufacturer. The NEDC is standardized within the "Directive 98/69/EC of the European Parliament and of the Council" [36] and defines a drive of 11 kilometers within 1180 seconds. Four identical phases represent urban driving (average speed: 19 km/h) and a phase which represents an overland tour (maximum speed: 120 km/h). Figure 5 shows the resulting speed profile over time.

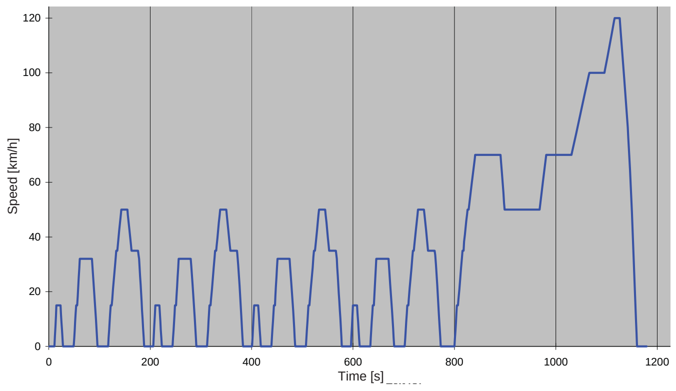

Figure 5. Speed profile over time derived from the New European Driving Cycle (NEDC) [Adapted from [37]]

The speed profile is used to determine the necessity of software components, sensors and actuators during the different speeds of the vehicle. ACCs are commonly activated at a speed larger than 30 km/h. An automatic parking assist can be reasonable used at a speed below 15 km/h. Table I shows the necessary components at three speed ranges (Column 2 to 4), which are necessary to enable the needed functionality of the vehicle.

In the following subsection, the implemented simulation is briefly outlined and the assumptions made are specified.

V. Discrete Event-Driven Simulation

Within this section an event-driven simulation is presented, which is applied to the previously presented system of the case study to estimate the energy demand of two different allocations of software components. The developed simulation uses the concept of an event-driven simulation as presented by Banks et al. [38] (cf. also [39] and [40]) and is implemented as discrete event-driven simulation, because the system only changes after discrete events. The considered system is adaptive (cf. Section III), i.e., the energy demand depends on the system context and the previously executed system state. That means it is necessary to simulate the context of the system as it is defined by Zeigler et al. [41], where the model has to generate the necessary data and the simulation the model's behavior.

The simulation focuses the allocation of software components within a given hardware architecture. This enables to optimize the allocation of software components or the optimized integration of new software components into an existing system. (The usage within Design Space Explorations would be possible, but is not considered within this work.) The position of necessary sensors and actuators can be predetermined, if there are hardware restrictions, or the position is resulted from the allocation of the affiliated software component. A mixture is also possible. However, within this case study the position of sensors and actuators are not predetermined. Furthermore, the

TABLE I. STATES OF SOFTWARE COMPONENTS, SENSORS AND ACTUATORS DURING DIFFERENT VEHICLE SPEEDS AND THEIR SOFTWARE RELEVANT PARAMETERS

Software Component, Sensors, Actuators	$v < 15$ km/h	$v \geq 15$ km/h $v \leq 30$ km/h	$v > 30$ km/h	Exec. Time	Cycle Time	Deadline	Output
Long-Range Radar	OFF	OFF	ON	25 ms	100 ms	100 ms	256 Bit
Signal Processing (LRR)	OFF	OFF	ON	9 ms	100 ms	100 ms	64 Bit
Object Recognition (LRR)	OFF	OFF	ON	30 ms	100 ms	100 ms	64 Bit
Short-Range Radar (Front)	ON	OFF	ON	15 ms	100 ms	100 ms	256 Bit
Signal Processing (SRR-F)	ON	OFF	ON	9 ms	100 ms	100 ms	64 Bit
Object Recognition (SRR-F)	ON	OFF	ON	30 ms	100 ms	100 ms	64 Bit
Short-Range Radar (Rear)	ON	OFF	OFF	15 ms	100 ms	100 ms	256 Bit
Signal Processing (SRR-R)	ON	OFF	OFF	9 ms	100 ms	100 ms	64 Bit
Object Recognition (SRR-R)	ON	OFF	OFF	30 ms	100 ms	100 ms	64 Bit
Path & Curve Calculation	OFF	OFF	ON	25 ms	100 ms	100 ms	64 Bit
Course Prediction	OFF	OFF	ON	25 ms	100 ms	100 ms	64 Bit
ACC Control	OFF	OFF	ON	5 ms	100 ms	100 ms	2x 32 Bit
Parking Assistant Control	ON	OFF	OFF	20 ms	100 ms	100 ms	3x 32 Bit
Engine Control	ON	ON	ON	3 ms	30 ms	30 ms	-
Brake Control	ON	ON	ON	3 ms	30 ms	30 ms	-
Steering Control	ON	ON	ON	3 ms	30 ms	30 ms	-
Wheel Speed Sensor	ON	ON	ON	5 ms	30 ms	30 ms	32 Bit
Steering Angle Sensor	ON	OFF	ON	5 ms	30 ms	30 ms	16 Bit
Yaw Rate Sensor	ON	OFF	ON	10 ms	50 ms	50 ms	32 Bit

TABLE II. ENERGY RELEVANT PARAMETERS OF SENSORS AND ACTUATORS

Sensor/Actuator	Power Consump.	Activation Time
Long-Range Radar	4.500 mW	25 ms
Short-Range Radar	1750 mW	25 ms
Engine Control	168 mW	1 ms
Brake Control	168 mW	1 ms
Steering Control	168 mW	1 ms
Wheel Speed Sensor	120 mW	1 ms
Steering Angle Sensor	75 mW	1 ms
Yaw Rate Sensor	125 mW	5 ms

TABLE III. ENERGY RELEVANT PARAMETERS OF COMMUNICATION

Component	CAN TRX	CAN CC	Eth. TRX	Eth. CC
TX	152 mW	1.1 mW	35.5 mW	8.2 mW
TX#Nodes	13.1 mW	0.02 mW	-	-
RX	2.5 mW	0.3 mW	16.5 mW	2.1 mW
RX#Nodes	0.01 mW	0.01 mW	-	-
Offset	22 mW	23 mW	44 mW	154.7 mW
Sleep	50 µW	20 µW	560 µW	330 µW

allocation of the specific software components are identical within the different system states, i.e., a reallocation of software components during runtime is not considered within this case study. (Simulation and the energy model are able to cope with a reallocation during runtime of the system, but this is not considered within this work.)

Equation (1) is used within the implemented simulation to estimate the energy demand of the later system. Some assumptions which specify the behavior of the system are necessary to made to determine the equation input data.

The execution times of software components vary from ECU to ECU, where the processor frequency has a large influence. In accordance with Walla et al. [42], the execution time is assumed to be inversely proportional to the processor frequency. On processors the software components are scheduled by the Round Robin algorithm, which is preemptive and based on fixed time slices for each process. It is assumed that the time slices are much shorter than the execution times of the software components, i.e., execution times are enlarged, if more than one software component is active at the same time. Furthermore, it is assumed, that software component specific sensors and actuators are only active, when the affiliated software component is active. This results in an energy demand to (de)activate sensors and actuators, however, as described within Subsection IV-B this energy is less than the energy needed to stay active. Further power management techniques like Dynamic Voltage Scaling are not used, because executing software components at maximum processor speed and deactivation afterwards is more energy efficient, if the entire system is considered [43]. Additionally, it is assumed that the communication data rate is used efficiently, i.e., splitting payload is just done, if necessary. Through that the necessary overhead is determinable, even if no further details concerning the kind of communication is available.

To simulate the system context the given speed profile of the NEDC is analyzed to divide the time sequence into slices with non-changing system states. Equation (1) enables to determine the energy demand of every system states. The transitions between system configurations are considered using the necessary energy demands for activation and deactivation of software and hardware components as described within Subsection IV-B.

In addition, different constraints are checked to verify the validity of the allocation. This includes obtaining deadlines of software components, maximum data rate of a network and workload of CPUs. The CPU workload is determined using the resulting workloads of the software components. The workload is not allowed to exceed 100 % or a given maximum value. The data rate is verified using the transmitted data per network, if bus based communication is used. Using full-duplex Ethernet the data rate per interface and direction is analyzed. It is also possible to extend the constraint checks, however, this is not focus of this work. The deadline of software components are examined by comparing the execution time after the scheduling process. Within this case study it is assumed that the deadline of the software components is equal to the cycle time.

In the following section, the examined allocations of software components within the system are detailed and the results of the energy demand estimation are presented and discussed.

VI. Energy Demand Estimation

The energy demand of the presented system (Section IV) is estimated using the introduced simulation (Section V), which is based on the given energy estimation model (Section II). The speed profile provided by the NEDC (cf. Subsection IV-C) is used as system context, which results 48 configuration changes during the NDEC according to the thresholds shown in Table I.

Within the following, the two considered allocations of software components are presented. Afterwards, the estimated energy demand of the system is presented and discussed.

A. Allocation of Software Components

Two possible allocations of software components within the hardware architecture are considered. A so-called "Function-based Allocation", which represents the current usual kind of allocation. That means software components associated to a specific functionality are grouped together on one or just a few hardware components. This is due to the fact that suppliers had previously provided hardware and software components as an integrated system. This paradigm changes nowadays and software components become independent from hardware components, e.g., using standardized software architectures such as AUTOSAR [44]. The other allocation is called "Energy-focused Allocation" where the software components are placed to reduce the energy demand of the system. Within an adaptive system it is relevant to take the usage of the system into consideration, e.g., it is advantageous to group software components which are needed at the same time. This reduces for example the energy demand to activate ECUs. Table IV shows the allocation of the software components to the hardware components of the presented hardware architecture (Section IV-B).

In the following subsection, the estimated energy demands using these allocations are presented.

B. Energy Demand Estimation

Table V shows the resulted energy demand estimation of the two previously presented allocations of software components within the hardware architecture. Beside the energy demand of the entire system, the percentage of energy demand for CPU, ECU offset, sensors/actuators, communication and adaptivity is

TABLE IV. ALLOCATION OF SOFTWARE COMPONENTS, SENSORS AND ACTUATORS ON ECUS (ECU NUMBERS ACCORDING TO FIGURE 4)

Software Component, Sensor, Actuator	Allocation "Func.-based"	Allocation "En.-focused"
Long-Range Radar	#16	#2
Signal Processing (LRR)	#16	#1
Object Recognition (LRR)	#16	#1
Short-Range Radar (Front)	#2	#2
Signal Processing (SRR-F)	#2	#1
Object Recognition (SRR-F)	#4	#1
Short-Range Radar (Rear)	#2	#2
Signal Processing (SRR-R)	#2	#1
Object Recognition (SRR-R)	#4	#1
Path & Curve Calculation	#16	#17
Course Prediction	#16	#17
ACC Control	#16	#17
Parking Assistant Control	#4	#17
Engine Control	#18	#16
Brake Control	#16	#16
Steering Control	#12	#16
Wheel Speed Sensor	#17	#18
Steering Angle Sensor	#12	#18
Yaw Rate Sensor	#17	#18

shown. (This work focuses on the electronic system, e.g., the components like the engine, which is controlled by "Engine Control" is excluded.) Table VI further details the energy demand for communication (cf. [10]) and adaptivity (cf. Subsection IV-B).

In the following subsection, the presented energy demand estimations are discussed.

C. Discussion

As shown within Table V, the energy demand of the "Energy-focused Allocation" is about 18 % less than of the "Function-based Allocation". The column "Difference" shows that the energy for CPU, ECU offset, sensors/actuators and communication is decreased and the energy for adaptivity is increased. As shown, the increased energy demand is less than the decreased energy, which results an energy saving in total. However, the increase of energy of adaptivity is resulted by the possibility to deactivate software components and ECUs. As shown in Table VI most of the energy is necessary for (de)activation of ECUs, which enables the saving of the ECU offset energy demand. It is mentionable that just the system context "Vehicle Speed" is evaluated in this case study and just two applications are available. Within real vehicles a lot more applications are executed (cf. Section III) and more context information is interpretable. Through that a lot of system states are possible, which may result a lot of energy to enable adaptivity. Even within the 1180 seconds of the NDEC the system has to switch the states 48-times. Through that, it is necessary to model and evaluate the energy demands of different allocations to evaluate and reach a trade-off. However, a deactivation of components to save energy is only useful, if more energy is saved than needed for the deactivation and activation process [2].

TABLE V. ESTIMATED ENERGY DEMAND OF THE SOFTWARE COMPONENT ALLOCATIONS "FUNCTION-BASED ALLOCATION" AND "ENERGY-FOCUSED ALLOCATION"

	Function-based		Energy-focused			
	Energy Demand	Percentage	Energy Demand	Percentage	Difference	
System	20725.11 Ws		16989.03 Ws		-3736.08 Ws	-18.03 %
CPU	3441.16 Ws	16.61 %	3330.85 Ws	19.61 %	-110.31 Ws	-3.21 %
ECU Offset	9372.00 Ws	45.22 %	8229.00 Ws	48.44 %	-1143.00 Ws	-12.20 %
Sensors/Actuators	4711.41 Ws	22.73 %	3570.02 Ws	21.01 %	-1141.39 Ws	-24.23 %
Communication	2933.12 Ws	14.15 %	1526.24 Ws	8.98 %	-1406.88 Ws	-47.97 %
Adaptivity	267.42 Ws	1.29 %	332.92 Ws	1.96 %	+65.50 Ws	+24.49 %

TABLE VI. ENERGY DEMAND IN DETAIL FOR COMMUNICATION AND ADAPTIVITY OF THE "FUNCTION-BASED ALLOCATION" AND THE "ENERGY-FOCUSED ALLOCATION"

	Function-based		Energy-focused	
	Energy Demand	Percentage	Energy Demand	Percentage
Communication	2933.12 Ws		1526.24 Ws	
Comm. Connections (incl. Transfer)	2091.21 Ws	71.30 %	1052.30 Ws	68.95 %
Transfer	11.50 Ws	0.39 %	13.37 Ws	0.88 %
Listener	838.84 Ws	28.60 %	469.02 Ws	30.73 %
Energy Saving Mode	0.10 Ws	0.10 %	4.93 Ws	0.32 %
Adaptivity	267.42 Ws		332.93 Ws	
(De)Activation of Software Components	2.42 Ws	0.91 %	2.92 Ws	0.88 %
(De)Activation of ECUs	265.00 Ws	99.09 %	330.00 Ws	99.12 %

The resulted energy demand of the case study also shows that the energy savings are realized by the ECU, sensor/actuators and communication instead of the CPU. Today's CPUs are highly optimized and further energy saving potential is difficult to reach. In addition, the percentage of different energy demands are more or less equal between the two allocations. This means that it is necessary to consider the entire system, instead of focusing a specific part of the system. The case study shows the relevance of modeling the energy demand of adaptive, networked, embedded systems as a whole. The accuracy of the estimation depends on the considered system. Using the accuracy ranges presented within the previous work concerning the energy model (cf. Section II) the accuracy is theoretically between -11.4 % and +7.1 %. (The estimation accuracy of ECU offset and adaptivity energy demand is estimated to be ±10 %.)

In the following section, the results are concluded.

VII. CONCLUSION

This paper presented a case study, which shows the applicability of the energy model, which was presented within previous work, and demonstrates the possible energy saving potential within adaptive, networked, embedded systems. The energy model is part of previous work and was used within this case study to estimate the energy demand of two allocations of software components.

Within the case study of this paper two advanced driver assistant systems were presented in detail including the resulting software architecture and the necessary software components. The considered system is adaptive and deactivates temporarily unnecessary software and hardware components which are not needed during the current system context. Within this case study the vehicle speed was used as the system context, which was derived by the New European Driving Cycle. Different speed thresholds result the activation or deactivation of (un)necessary components. The considered hardware architecture and the energy relevant parameters were presented. Afterwards, a simulation was briefly explained, which enables the estimation of the energy demand of an adaptive, networked, embedded system during a specific system context. Finally, the estimated energy demand of the two software component allocations were presented and the results discussed. The simulation results showed an energy saving potential of about 18 %. The energy demand of the adaptivity process itself increases significantly, nevertheless this is still less than the enabled energy saving of the other components resulted by adaptivity.

The results of this paper showed the relevance of modeling the energy demand of adaptive, networked, embedded systems and a considerable energy saving potential. This kind of estimation at the network level needs an appropriate energy model to estimate the energy demand of adaptive, networked, embedded systems during the development of the system, which was presented within previous work.

ACKNOWLEDGMENT

The research leading to these results has received funding from the German Federal Ministry for Economic Affairs and Energy (BMWi).

REFERENCES

[1] A. Barthels, J. Fröschl, H.-U. Michel, and U. Baumgarten, "An Architecture for Power Management in Automotive Systems," in Proceedings of the International Conference on Architecture of Computing Systems, vol. 7179, Springer, 2012, pp. 63–73, ISBN: 978-3-642-28292-8.

[2] C. Schmutzler, A. Krüger, F. Schuster, and M. Simons, "Energy efficient automotive networks: state of the art and challenges ahead," in International Journal of Communication Networks and Distributed Systems, vol. 9, no. 3/4, 2012, pp. 266–285, ISSN: 1754-3916.

[3] J. Weber, "Automotive Development Processes: Processes for Successful Customer Oriented Vehicle Development," Springer-Verlag Berlin Heidelberg, 2009, ISBN: 978-3-642-01253-2.

[4] A. Monetti, T. Otter, and N. Ulshöfer, "Spritverbrauch senken, Reichweite erhöhen: System-Basis-Chip für den Teilnetzbetrieb am CAN-Bus," Elektronik Automotive, no. 11, 2011, pp. 24–27, ISSN: 1614-0125.

[5] A. D. Little, "Market and Technology Study Automotive Power Electronics 2015," 2006, URL: www.adlittle.com/downloads/tx_adlreports/ADL_Study_Power_Electronics_2015.pdf [accessed: 2016-05-17].

[6] P. Heinrich and C. Prehofer, "Network-Wide Energy Optimization for Adaptive Embedded Systems," ACM SIGBED Review, vol. 10, no. 1, 2013, pp. 33–36, ISSN: 1551-3688.

[7] P. Heinrich and C. Prehofer, "Early Energy Estimation in the Design Process of Networked Embedded Systems," in Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems, 2013, pp. 214–220, ISBN: 978-989-8565-43-3.

[8] P. Heinrich, H. Bergler, and D. Eilers, "Energy Consumption Estimation of Software Components based on Program Flowcharts," in Proceedings of the 11th IEEE International Conference on Embedded Software and Systems (ICESS), 2014, pp. 550-553, ISBN: 978-1-4799-6122-1.

[9] P. Heinrich, H. Bergler, and E. Oswald, "Early Energy Estimation of Networked Embedded Systems Executing Concurrent Software Components," in International Journal of Modeling and Optimization, vol. 5, no. 2, pp. 119–127, 2015, ISSN: 2010-3697.

[10] P. Heinrich, D. Gossen, E. Oswald, and R. Knorr, "Early Energy Estimation of Heterogeneous Embedded Networks within Adaptive Systems," in Energy Efficient Vehicles 2015, B. Bäker and L. Morawietz, Eds., Dresden: TUDpress, 2015, pp. 64–76, ISBN: 978-395-9080-08-8.

[11] P. Marwedel, "Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems," 2nd ed., Springer, 2011, ISBN: 978-940-0702-56-1.

[12] A. Metzner, "Effizienter Entwurf verteilter eingebetteter Echtzeit-Systeme," Dissertation, Universität Oldenburg, 2007.

[13] AUDI AG, "Adaptive Cruise Control with Stop & Go Function," 2011, URL: www.audi-technology-portal.de/en/electrics-electronics/driver-assistant-systems/adaptive-cruise-control-with-stop-go-function [accessed: 2016-05-17].

[14] Robert Bosch GmbH, "Kraftfahrtechnisches Taschenbuch," 26th ed., Wiesbaden: Vieweg, 2007, ISBN: 978-383-4814-40-1.

[15] BMW AG, Active Cruise Control with Stop&Go Function, 2015, URL: www.bmw.com/com/de/insights/technology/technology_guide/ articles/active_cruise_control_stop_go.html [accessed: 2016-05-17].

[16] U.S. Software System Safety Working Group, "Adaptive Cruise Control System Overview," 2005, URL: www.sunnyday.mit.edu/safety-club/workshop5/Adaptive_Cruise_Control_Sys_Overview.pdf [accessed: 2016-05-17].

[17] H.-H. Braess and U. Seiffert, Eds., "Vieweg-Handbuch Kraftfahrzeugtechnik," 5th ed., ser. ATZ-MTZ-Fachbuch, Wiesbaden: Vieweg, 2007, ISBN: 978-383-4802-22-4.

[18] Daimler AG, Active Parking Assist, 2016, URL: www.techcenter.mercedes-benz.com/en/active_parking/detail.html [accessed: 2016-05-17].

[19] H. Hansson, M. AAkerholm, I. Crnkovic, and M. Torngren, "SaveCCM - a component model for safety-critical real-time systems," in Proceedings of the 30th EUROMICRO Conference, 2004, pp. 627–635, ISBN: 978-076-9521-99-2.

[20] A. G. Fontquerni, "Embedded Linux RADAR device: Taking advantage on Linaro tools and HTML5 AJAX real-time visualization," Embedded Linux Conference Europe, Barcelona, 2012, URL www.elinux.org/images/7/75/Embedded_Linux_RADAR_Device.pdf [accessed: 2016-05-17].

[21] Freescale Semiconductor, "MPC5674F Microcontroller Data Sheet," 2015, URL: www.cache.freescale.com/files/32bit/doc/data_sheet/MPC5674F.pdf [accessed: 2016-05-17].

[22] T. Weber, V. Lauer, D. Mann, and M. Simons, "Das umfassende Energiemanagement: Vom konventionelle Verbrenner bis zum E-Antrieb," in 4. VDI-Tagung Baden-Baden Spezial 2010, ser. VDI-Berichte, vol. 2098, VDI Verlag, 2010, ISBN: 978-3-18-0920-98-6.

[23] A. Y. Benbasat, "An Automated Framework for Power-Efficient Detection in Embedded Sensor Systems," Dissertation, 2007, URI: http://hdl.handle.net/1721.1/38524.

[24] Robert Bosch GmbH, "Fernbereichsradarsensor LRR3: Long-Range Radar, 3. Generation," 2009, URL: www.produkte.bosch-mobility-solutions.de/media/db_application/pdf_2/de/LRR3_Datenblatt_DE_2009.pdf [accessed: 2016-05-17].

[25] Continental AG, "Short Range Radar Sensor SRR 20X /-2 /-2C /-21," 2013, URL: www.conti-online.com/www/industrial_sensors_de_en/themes/srr20x_en.html [accessed: 2016-05-17].

[26] Freescale Semiconductor, "Automotive Engine Control IC: Technical Data," 2014, URL: www.cache.freescale.com/files/analog/doc/data_sheet/MC33810.pdf?pspll=1 [accessed: 2016-05-17].

[27] NXP Semiconductors, "KMI17/4 - Rotational speed sensor: Product data sheet: Rev. 1, September 2014," 2014, URL: www.nxp.com/documents/data_sheet/KMI17_4.pdf [accessed: 2016-05-17].

[28] NXP Semiconductors, "KMA220 - Dual channel programmable angle sensor: Product data sheet: Rev. 2, April 2013," 2013, URl: www.nxp.com/documents/data_sheet/KMA220.pdf [accessed: 2016-05-17].

[29] Robert Bosch GmbH, "Combined inertial sensor for vehicle dynamics control - SMI650: Automotive Electronics," 2013, URL: www.bosch- semiconductors.de/media/pdf_1/einzeldownloads/vehicle dynamics_systems/datenblatt_smi650.pdf [accessed: 2016-05-17].

[30] Robert Bosch GmbH, "CAN Specification: Version 2.0," Stuttgart, 1991, URL: www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf [accessed: 2016-05-17].

[31] Microchip Technology Inc., "High-Speed CAN Transceiver: MCP2561/2: Revision C," 2014, URL: www.microchip.com/downloads/en/DeviceDoc/20005167C.pdf [accessed: 2016-05-17].

[32] Microchip Technology Inc., "Stand-Alone CAN Controller with SPI Interface: MCP2515: Revision G," 2012, URL: www.microchip.com/downloads/en/DeviceDoc/21801G.pdf [accessed: 2016-05-17].

[33] Standard Microsystems Corp., "Small Footprint MII/RMII 10/100 Ethernet Transceiver with HP Auto-MDIX and flexPWR Technology: LAN8710A/LAN8710Ai: Revision 1.4," 2012, URL: www.microchip.com/downloads/en/DeviceDoc/8710a.pdf [accessed: 2016-05-17].

[34] Cirrus Logic Inc., "CS8900A: Product Data Sheet: Crystal LAN Ethernet Controller," 2010, URL: www.cirrus.com/jp/pubs/proDatasheet/CS8900A_F5.pdf [accessed: 2016-05-17].

[35] C. Schmutzler, "Hardwaregestützte Energieoptimierung von Elektrik/Elektronik-Architekturen durch adaptive Abschaltung von verteilten, eingebetteten Systemen," Dissertation, 2012, ISBN: 978-86-6448-75-9.

[36] European Parliament, Council of the European Union, "Directive 98/69/EC of the European Parliament and of the Council," 13.10.1998, URL: www.eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1998L0069:19981228:EN:PDF [accessed: 2016-05-17].

[37] S. Schmidt, "EcoTest: Testing and Assessment Protocol: Release 2.0," 2015, URL: www.ecotest.eu/html/testing%20and%20assessment%20protocol.pdf [accessed: 2016-05-17].

[38] J. Banks, J. S. Carson, II, B. L. Nelson, and D. M. Nicol, "Discrete-Event System Simulation," 4th ed., ser. Prentice-Hall international series in industrial and systems engineering, Pearson Prentice Hall, 2005, ISBN: 978-013-6062-12-7.

[39] P. Bastian, "Grundlagen der Modellbildung und Simulation," Vorlesungsskript, Universität Stuttgart, Stuttgart, 2008, URL: www.conan.iwr.uni-heidelberg.de/teaching/scripts/msarticle.pdf [accessed: 2016-05-17].

[40] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger, "Modeling and Simulation: An Application-Oriented Introduction," Berlin: Springer, 2013, ISBN 978-364-2395-24-6.

[41] B. P. Zeigler, H. Praehofer, and T. G. Kim, "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems," 2nd ed., Academic Press, 2000, ISBN: 978-012-7784-55-7.

[42] G. Walla, A. Herkersdorf, A. S. Enger, A. Barthels, and H.-U. Michel, "An automotive specific MILP model targeting power-aware function partitioning," in Proceedings of the 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), pp. 299–306, 2014, ISBN: 978-147-9937-69-1.

[43] R. Jejurikar and R. Gupta, "Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems," in Proceedings of the 2004 International Symposium on Low Power Electronics and Design, New York, pp. 78–81, 2004, ISBN: 978-158-1139-29-7.

[44] AUTOSAR Development Cooperation, "Automotive Open System Architecture," URL: www.autosar.org [accessed: 2016-05-17].