INVITED PAPER

Dynamic and Partial FPGA Exploitation

Circuit components that can be reconfigured by software commands may be able to reduce the rapidly-growing cost of electronics in future cars.

By Jürgen Becker, *Senior Member IEEE*, Michael Hübner, *Student Member IEEE*, Gerhard Hettich, Rainer Constapel, Joachim Eisenmann, and Jürgen Luka

ABSTRACT | Today's field programmable gate array (FPGA) architectures, like Xilinx's Virtex-II series, enable partial and dynamic run-time self-reconfiguration. This feature allows the substitution of parts of a hardware design implemented on this reconfigurable hardware, and therefore, a system can be adapted to the actual demands of applications running on the chip. Exploiting this possibility enables the development of adaptive hardware for a huge variety of applications. A novel method for communication interfaces using look up table (LUT)-based communication primitives enables an exact separation of reconfigurable parts and a fast and intelligent bus-system. A new adaptive software/hardware reconfigurable system is presented in this paper, using a real application in the automotive domain implemented on a Xilinx Virtex-II 3000 FPGA to present results.

KEYWORDS | Automotive electronic systems; dynamic partial reconfiguration; high-level FPGA design flow; on-demand adaptivity

I. INTRODUCTION

Self-reconfiguration and adaptivity are the keywords of a new design methodology for reconfigurable hardware. One of the benefits provided by using this feature is a reduction of power dissipation by storing functionality to external memory. This means that a smaller field programmable gate array (FPGA) can be used to run an application by configuring parts of the chip as idle applications, which are substituted on demand by the functions that are actually needed at any given point. For this to be achieved, a system that controls these processes, for example saving actual variables and states of a hardware function or initiating self-reconfiguration, is necessary [18]. The advantage gained from this is the exploitation of parallel working tasks (functions) on the hardware in comparison to the sequential workflow of microprocessors. The high performance of reconfigurable hardware and the possibility of real hardware parallelism help to overcome increasing problems regarding data processing using traditional microcontrollers and microprocessors [2].

Reconfigurable architectures, such as Xilinx Virtex-II FPGAs, can be used to build systems with adaptive network-on-chip architectures [9]. Changing demands for data processing during run-time requires an adaption of both bandwidth for communication and suitable network topology. Thus for filter-based applications, such as an Moving Picture Experts Group (MPEG) application, a torus topology for dataflow is preferred. While the system is in process, a controller application requires a star topology with high bandwidth for data transfer, so that the system can provide the necessary topology to maintain the required performance for communication [12]. This adaptivity is not only important for high-performance data transfer and communication, but also for innovative systems, making use of adaptive reconfigurable systems in real-time applications a relevant issue. Meeting the real-time constraints in critical applications, for example in automotive applications, is a crucial factor for their encouragement in industrial systems.

Another advantage of using reconfigurable hardware is the use of soft-core processors to build up a mixed software and hardware reconfigurable system. Dataflow-oriented applications can be implemented effectively in optimized RISC processors, such as Xilinx MicroBlaze or PicoBlaze soft-core processors, and configured on the FPGA [20]. A new parallel software-reconfigurable processor system on

Manuscript received June 14, 2006; revised August 14, 2006.
J. Becker and **M. Hübner** are with the Institut für Technik der Informationsverarbeitung (ITIV), Universität Karlsruhe (TH), 76128 Karlsruhe, Germany (e-mail: becker@itiv.uni-karlsruhe.de; huebner@itiv.uni-karlsruhe.de).
G. Hettich, **R. Constapel**, **J. Eisenmann**, and **J. Luka** are with DaimlerChrysler AG, 71034 Boeblingen, Germany (e-mail: gerhard.hettich@daimlerchrysler.com; rainer.r.constapel@daimlerchrysler.com; joachim.eisenmann@daimlerchrysler.com; juergen.luka@daimlerchrysler.com).
Digital Object Identifier: 10.1109/JPROC.2006.888404

the FPGA enables hardware/software partitioning on chip during run-time. A software/hardware reconfigurable system likewise provides the possibility of task migration from software to hardware and vice versa. One example application for such a system is found in the automotive domain. A user task running on a high-performance hardware resource can be substituted by a high-priority task (e.g., brake control) and extruded as a software task with lower performance. A run-time system negotiates the qality of service with the application and provides the available alternatives for execution.

The industry has already reacted to these demands and scenarios of application [13]; for example, new FPGAs like Xilinx Virtex-Pro contain up to four Power PC hard-wired cores to build up these hardware/software (HW/SW)-systems. This architecture leads to the keyword system-on-chip which is the final stage of such HW/SW-reconfigurable systems [3].

In the following sections, a dynamic and partially reconfigurable system will be described. Section II starts with a motivation for introducing reconfigurable systems in automotive applications. Section III begins with architectural information concerning FPGA and the basics of dynamic and partial reconfiguration. Section IV gives an overview of an implemented hardware reconfigurable system. Section V describes the reconfigurable system in the automotive domain. In Section VI, the area–time dependency of the reconfigurable system will be introduced. The paper is closed in Section VII with conclusions and future work.

II. MOTIVATION

In the automotive industry, the percentage of mechanical parts in the vehicle added value is constantly decreasing. Currently, about 35% of the value of a fully equipped luxury car is determined by electric, electronic, and mechatronic systems. This percentage will increase to about 50% in the coming years. Even in low- and mid-range cars, electronic systems like power window control, air conditioning, and navigation are mostly standard. Fig. 1 shows that the market volume for electronics in cars in Europe will be about 100 billion Euros in the year 2010.

This dramatically increasing importance of electronic systems can no longer be managed by the use of conventional technologies. In the past, the growing functionality has directly led to a growing number of electronic control units (ECUs) in the cars. But the total number of ECUs in a car is on one hand limited by the complexity of the networked system and on the other hand by the power consumption of the whole electrics system including mechatronic devices like sensors and actuators. This leads to a strong demand for reducing the number of ECUs in future cars by combining different application functions, which are currently developed by different suppliers, on a smaller number of ECUs. These

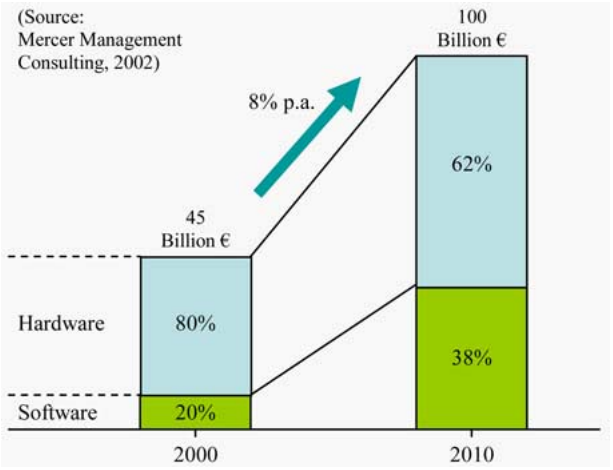

Fig. 1. ***Importance of electronics on vehicle added value.***

ECUs have to be more powerful, and, as a result of this, they are becoming more complex than today. One attempt to manage this complexity is currently being made by the AUTOSAR consortium [22]. Leading car manufacturers and their suppliers are jointly designing a common system architecture based on state-of-the-art microcontrollers. In this paper, a different approach is presented, where the microcontroller as the central component of an ECU is replaced by an FPGA which is dynamically and partially reconfigured.

III. ARCHITECTURE OF XILINX VIRTEX-II FPGAs

The SRAM-based Xilinx FPGAs of the Virtex-II series consist of configurable components, such as configurable logic blocks (CLBs), block RAM, hardware multiplier, input–output (I/O) elements, and switch matrixes for the connection of routing resources. In comparison to the coarse grained reconfigurable architecture described in [8], [15], and [17], this fine grained architecture allows path width down to one bit. Changing the content of the SRAM implies changes to the architecture, which represents the actual configuration (function) of the hardware. A detailed overview of reconfigurable hardware and the related methodologies is described in [25] and [26]. Design tools, such as Xilinx Integrated System Environment, enable the generation of the actual data needed to program such components, for example, from a Very High Speed Integrated Circuit Hardware Description Language (VHDL) description after synthesis, translation, mapping, and place and route processes. These programming data can be downloaded as bitstream to the FPGA which is then configured. After configuration, the designed architecture starts working immediately. All configurable elements of a Virtex-II FPGA are organized in columns. Fig. 2 shows a schematic view of several components of the FPGA. Xilinx

Virtex-II FPGAs allow for the reconfiguring of parts of the configuration memory at run-time. The idea of dynamic and partial reconfiguration describes the possibility to change parts of hardware on the configurable device while all other parts stay unaffected and operative. This is done by programming the respective cells of the SRAM while other memory areas stay unaffected. The memory of these FPGAs is also organized into columns, meaning that writing is therefore only possible in complete columns. The smallest changeable unit is a column with a width of one bit, a so-called frame, which therefore contains the configuration information of a complete column. The complete configuration information, for example, for one CLB column, consists of 22 frames. Segmentation of resources within the FPGA leads to the architecture that is described in the following sections. The transmission of reconfiguration data can be achieved, courtesy of a Joint Test Action Group (JTAG) interface, an external parallel interface (SelectMap), or an internal reconfiguration access port (ICAP). The first named JTAG interface, in comparison to SelectMap and ICAP, is bit-serial, while the faster eight-bit parallel interfaces are used for fast reconfiguration. The ICAP interface, in particular, is very interesting for dynamic and partial self-reconfiguration. This interface is accessible from the components within the FPGA and instantiating it as a component in VHDL enables access to the SRAM for configuration. More information can be found in [4].

The introduced term of self-reconfiguration has its meaning from the fact that the architecture integrated on the FPGA is able to control the configuration port during run-time. The system decides to load configuration data from external memory on demand and initiates the reconfiguration process by using the ICAP port. This feature is used to transmit partial reconfiguration data in the reconfigurable hardware system, which is described in Section IV.

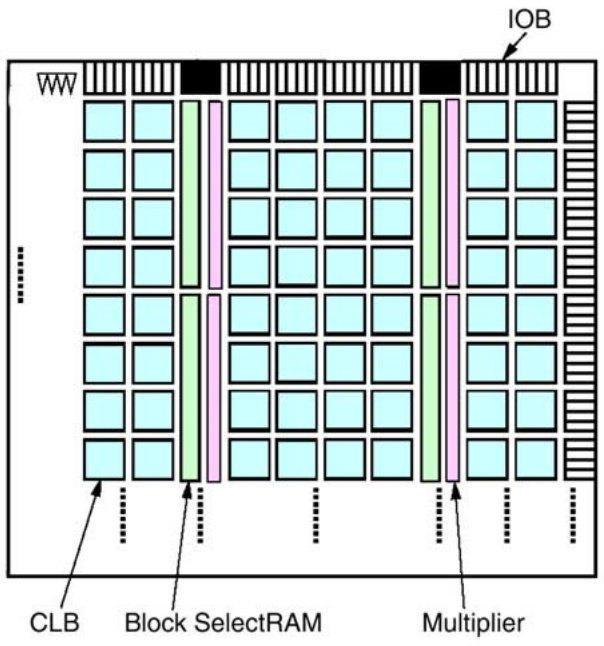

Fig. 2. ***Column-based organization of Virtex-II FPGA.***

IV. HARDWARE RECONFIGURABLE SYSTEM

During the past three years, a cooperative research project with the automotive industry, concerning the usage of dynamic and partially reconfigurable hardware for automotive applications, has been in progress at Karlsruhe, Germany. The motivation is to reduce the high number of control systems necessary for cabin functions. In a modern car, this number reaches up to 70 microprocessor systems in one car. In Fig. 3, only a fraction of all available functionalities within a car are presented. A dynamic and partially reconfigurable system was presented in March 2005 [21], which is able to provide cabin functions on demand (Fig. 4).

This system consists of a run-time module controller, implemented on a MicroBlaze soft-core processor from Xilinx. The controller is connected to an arbiter that controls the data communication on a bus macro, connecting all four reconfigurable module slots. The controller is also connected to a decompressor unit for loading configuration data from an external flash-memory. The system is connected to its environment via a controller area network (CAN) bus, which is a well established method in the automotive domain [5]. During run-time, a request (command) for an application is received by the CAN controller within the FPGA. The CAN controller is connected directly to the MicroBlaze controller as IP core and the command is processed by the run-time module controller and its run-time system. If this function is already configured to one of the four module slots, the command is transferred via the arbiter to the corresponding module. Response from the modules is also transferred via the bus system and provided to the run-time controller by the arbiter. Messages can now be sent via the CAN controller to the peripheral devices. It is possible for an application to be requested, but for it not to be configured to one module slot. In this case, the run-time controller sends a command to the decompressor system.

Within the external flash-memory, all necessary cabin functions are stored as compressed bitstream and after sending the decompressor module the start address for the function needed, a start command runs the decompression process within this module. The decompression runs parallel to all other hardware on the chip. Therefore, other functions will not stop their execution while decompression and reconfiguration takes place. During

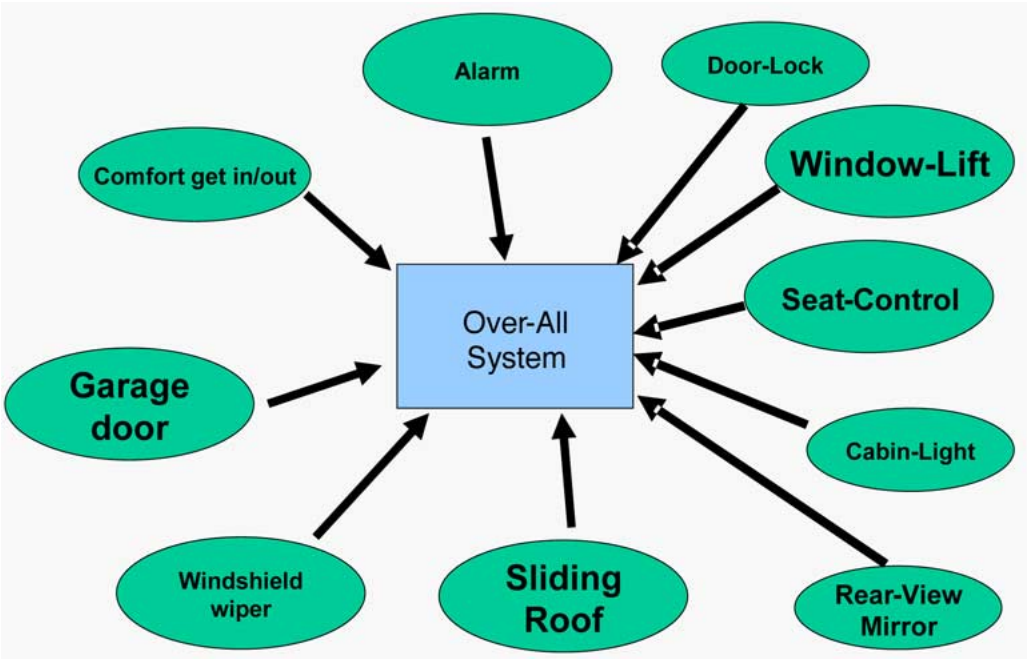

Fig. 3. *Different cabin functions of a modern car.*

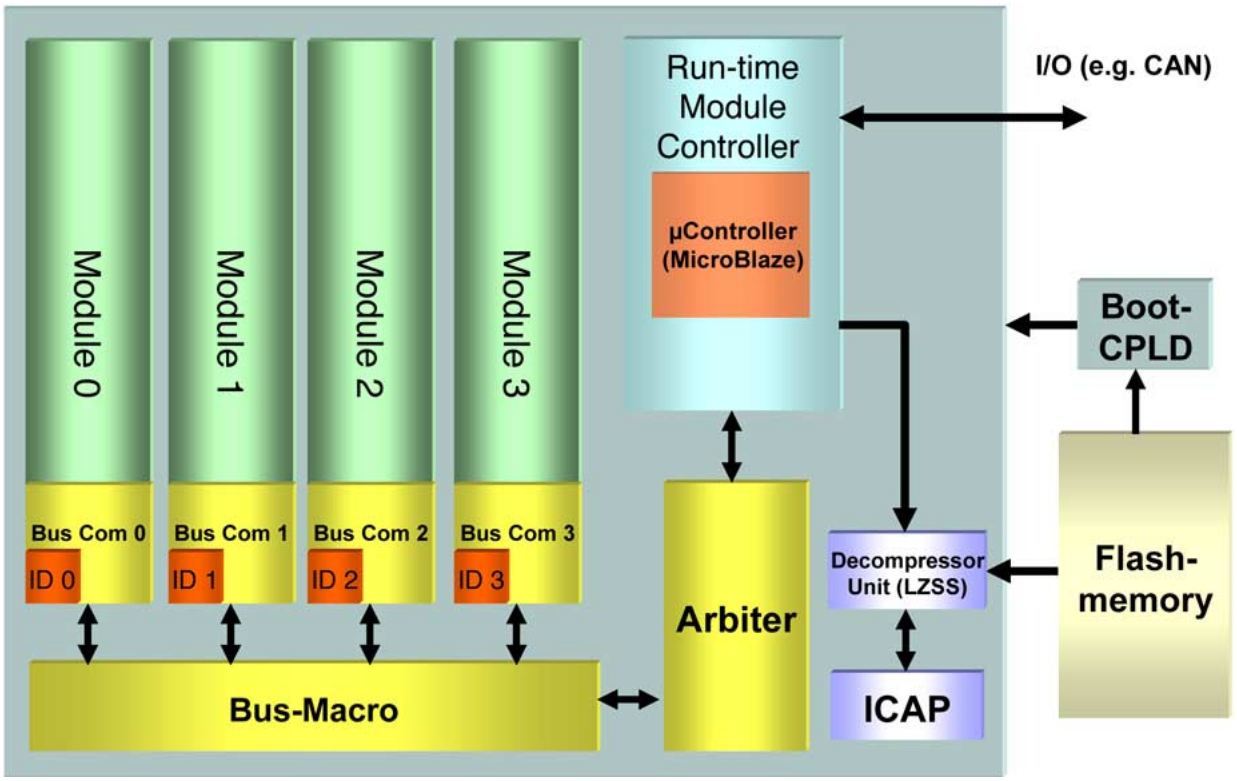

Fig. 4. *Reconfigurable system for automotive applications.*

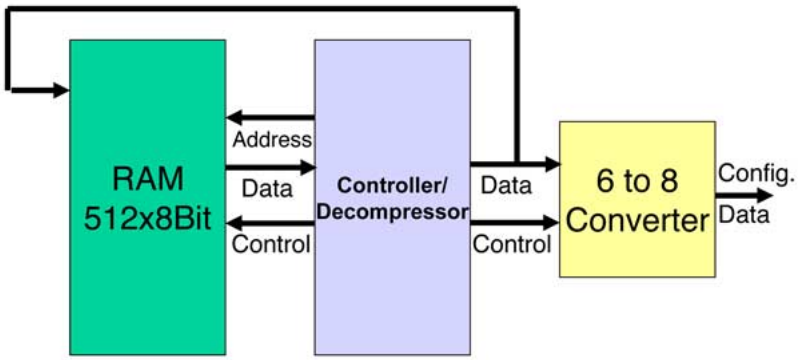

Fig. 5. *Schematic of decompressor unit.*

decompression, the configuration data are written directly to the ICAP of the FPGA. Other functions and the bus communication are not disturbed by this process.

Fig. 5 shows a schematic view of the decompressor implementation. The decompression algorithm used is based on LZSS from Lempel and Ziv. More detailed information can be found in [10]. The use of this decompressor saves up to 50% of external memory so that costs can be reduced. After successful decompression, an interrupt signals to the run-time controller that the function now is ready for use. Before this module can process data, status information and variables have to be restored to enable a restart at the state of last use. This so called context load process has to be done on every new start of a function. The run-time module controller allocates the memory for these context data and after reloading the context data to the module, messages (commands) can be processed by the function within the module slot.

If all module slots are occupied by functions, the run-time system within the controller has to check whether a module is in an idle, unused state. These modules are detected and the eldest, unused, idle module will be substituted. Before substitution, the system requests the context data for storage on the FPGA internal memory. These data are then transferred back, if the function is reused later.

The tasks of the run-time system are as follows.

- Communication Management:
  - delivering messages to the functions;
  - delivering messages from the functions;
  - storing messages;
- Reconfiguration Management:
  - finding an available module slot;
  - saving current state;
  - sending instruction block;
  - restoring state (by sending data);
- Resource Management:
  - keeping record of busy/idle modules;
  - managing the message buffer of each function;
  - storing the variables defining the states.

The run-time system (complete system) has to work within the borders of given real-time constraints, which can be fully met by this system. Response times are smaller than 10 ms and have their lower limits caused by the external memory access time. Furthermore, messages must be transferred in the right order to the respective module and to the external devices via the CAN bus. This has to be ensured by the scheduling of the run-time system within the controlling element.

The usage of such a system enables the saving of power dissipation by using a smaller FPGA device and by outsourcing configuration data [1]. Fig. 6 shows an example of the reduction in power dissipation by outsourcing functions. The measurement data of Fig. 6 includes the power dissipation of the FPGA. Power consumption of external memory is not included. With a special measurement system, the amount of current for the FPGA during reconfiguration was measured. The used XCV2000E FPGA needs two power supplies: 1.8 V for core and 3.3 V supply for I/O elements. Fig. 7 shows the measurement system. The measurement system consists of a PC with the control software, a Tektronix oscilloscope (Type TDS 220) connected to the PC's RS232 interface. The core and 3.3-V supply current is measured with a shunt resistor connected to an opened jumper bridge on the rapid prototyping board. By starting the measurement, the control software initializes the oscilloscope and preconfigures the FPGA. The next step is to start the measurement and transmit the second configuration to the board. The oscilloscope samples the voltage over the shunt resistor and stores the data into its memory. Then the measure cycle is stopped and the data is read back from the oscilloscope memory into the PC and saved into a file. This file is the basis for further calculations. This cycle is iterated n-times (n is given by the user). With the measurement system, the core voltage and the 3.3-V power supply were sampled to calculate the complete power dissipation.

The total power thus can be fragmented in P_{CORE} and P_{VCC}. The power was calculated with the formulas described below (Formula 1 and 2). The measurement results

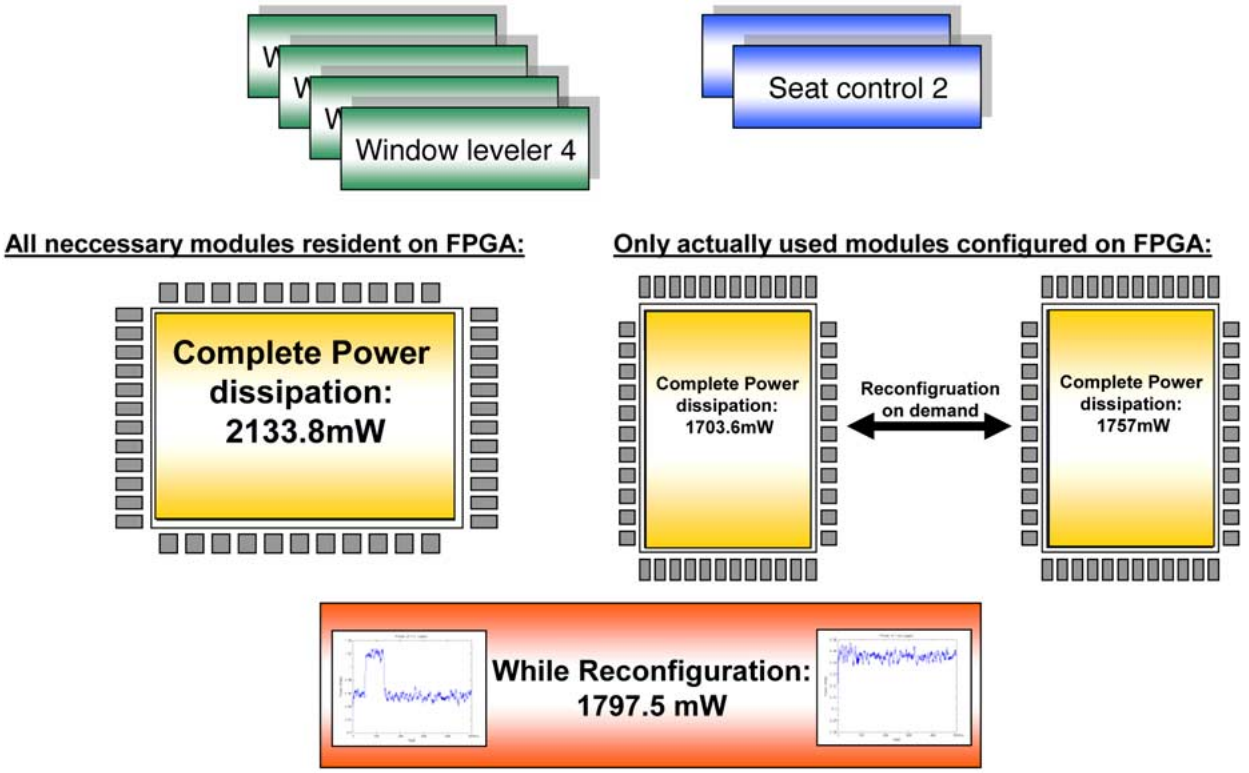

Fig. 6. ***Reduction of power dissipation by outsourcing of functions.***

show that the power consumption for processing the partial configuration data leads to an additional power consumption of 95 mW, as described in Fig. 6. The value of 1797.5 mW is below the power consumption of 2133.8 mW with all automotive functions configured on the chip. This small example shows the high potential for reducing power consumption if configuration on demand exploiting dynamic and partial reconfiguration is used in adaptive systems.

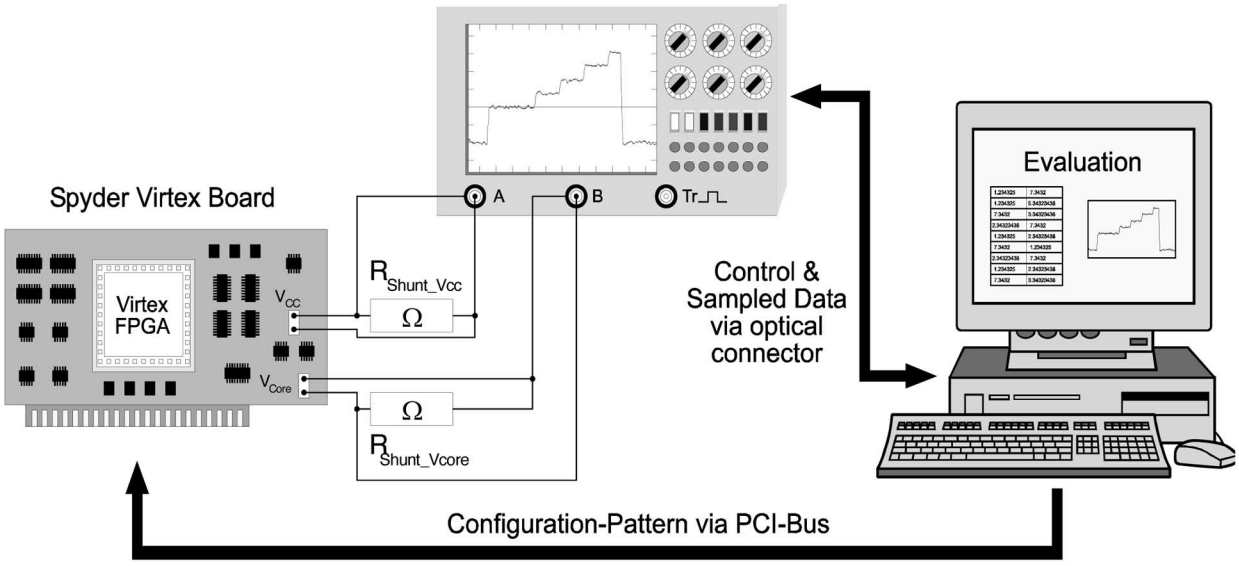

Fig. 7. ***Power measurement system.***

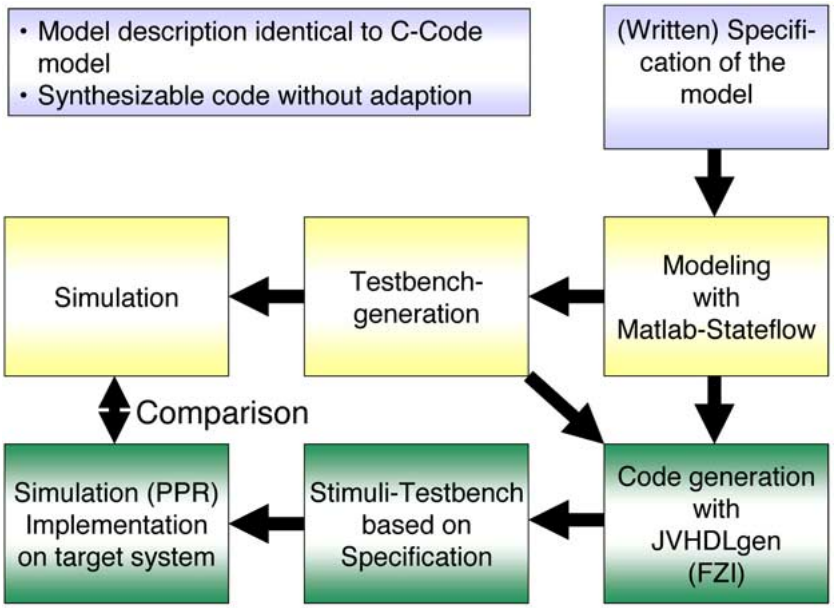

Fig. 8. ***Designflow for generating VHDL code from Stateflow description with Matlab.***

Formula 1: Average core power dissipation during reconfiguration:

$$\overline{P_{\mathrm{CORE}}} = \frac{\int_{T_{\mathrm{start}}}^{T_{\mathrm{start}}+T_{\mathrm{REC}}} P_{\mathrm{CORE}}(t)\cdot dt}{T_{\mathrm{REC}}}.$$

Formula 2: Average I/O power dissipation during reconfiguration:

$$\overline{P_{\mathrm{VCC}}} = \frac{\int_{T_{\mathrm{start}}}^{T_{\mathrm{start}}+T_{\mathrm{REC}}} P_{\mathrm{VCC}}(t)\cdot dt}{T_{\mathrm{REC}}}.$$

The high performance of an FPGA exploits the real-time parallel processing of tasks and enables the substitution of more than 12 control units for cabin functions with one system, when using the new reconfigurable method. Extrapolation and real tests show that a high-performance conventional processor has to be used to run all of these functions in quasi-parallel execution mode. However, these chips are expensive and have high-power dissipation. In addition, the increased portability of the functions, which are described as hardware implementation (VHDL), has advantages for industrial usage since functions can be treated as IP core and reused with FPGAs of a different size and even from other manufacturers.

In cooperation with the Forschungszentrum Informatik,[1] a tool called JVHDLgen has been developed. This tool enables the development of applications with Matlab Simulink-Stateflow from The Mathworks which is well established in the automotive industry. More information about the Simulink tool and Control Design can be found in [14]. The automotive cabin functions are modeled with Matlab Simulink-Stateflow, which allows for the simulation of a function and therefore a fast and safe inspection of the correct behavior is possible, before implementation occurs. At this point, a software generation in *C* with Matlab Staflow Coder is possible. The tool JVHDLgen allows for the generation of a VHDL code from the Stateflow description in Matlab. Fig. 8 shows the designflow with the tool JVHDLgen. The basis for the designflow is a State-Chart which is shown in Fig. 9 exemplarily. The diagram shows the counter for a window-lift for checking the position of the window.

It is important that the model description for VHDL is equal to the model for *C* code generation in the abstraction layer of Matlab Stateflow. This allows the generation of *C*— and VHDL codes for the functionalities for integrating either in software or hardware. In later steps, this will be used for HW/SW reconfigurable systems, which are able to run tasks in both software and hardware. This enables a hardware/software partitioning at design time and in new approaches, with task-migration techniques during runtime. The tool JVHDLgen is a further development of "sf2vhd" from Kevin Camera [6], which has now been

[1]Available: http://www.fzi.de/

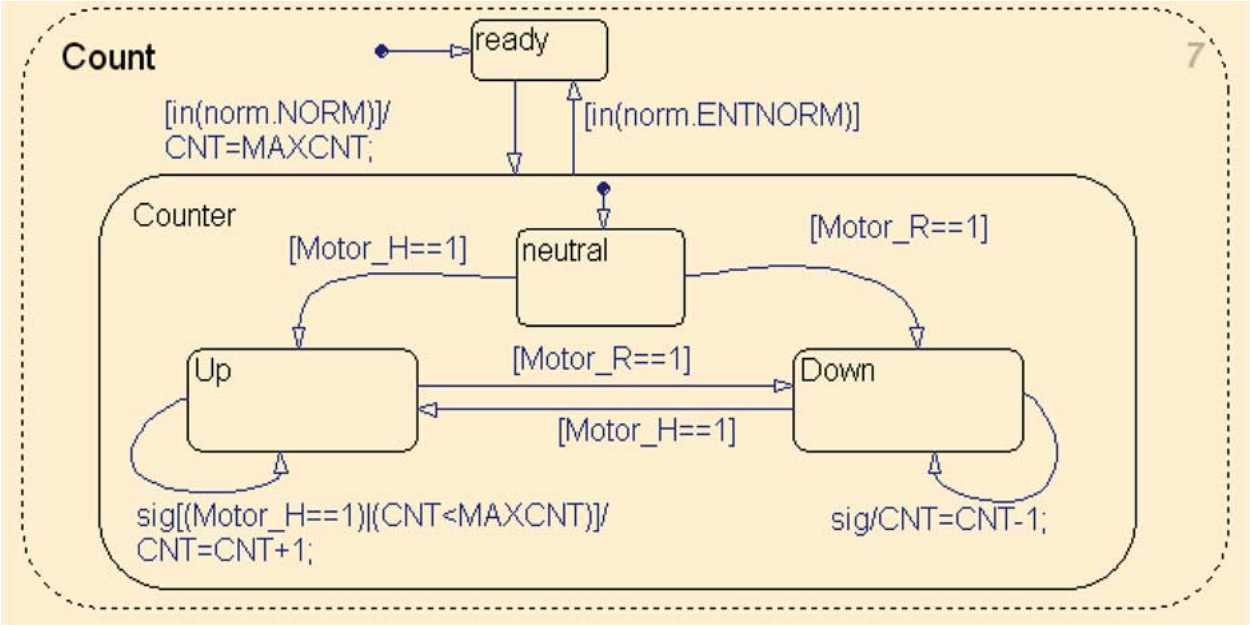

Fig. 9. ***State diagram modeled with Matlab-Stateflow.***

developed as a Java tool and is, therefore, easy to adapt to changing versions of Matlab Stateflow. Establishing a description in a high abstraction level is important for bringing new design methodologies and architectures into industrial processes and development sections, to ensure their acceptance by developers. Programming complex functions in VHDL is complicated and has a high risk of hidden failures. However, this risk can be reduced by using established tools, such as Matlab Stateflow and the possibility of simulation at an early stage of design.

Fig. 10 shows the physical implementation of the described system in a Virtex-II FPGA (XC2V3000).

The rapid prototyping platform used and the physics of the chip force the segmentation into a static and, for this example application, four reconfigurable blocks. On the right side, the static block with the run-time controller, the decompression module, the arbiter, and the CAN controller can be seen. This block will not be changed during run-time. The ICAP interface can also be found in the lower right area of this block. During run-time, these parts of the system must not be substituted or overwritten by other logic. In Fig. 10, four reconfigurable areas (module slots) can be found. These slots are connected via a bus macro which is static implemented on a fixed position for each module. This is the basis for an undisturbed communication during reconfiguration. One approach for optimized communication macros has been developed. More information can be found in [11]. These macros allow the fast and safe development of reconfigurable systems. To test this system in a real-world scenario, the tool Canoe from Vector Informatik is used [19]. Fig. 11 shows a test workbench. The Canoe system is connected to the CAN interface of the reconfigurable system. All implemented cabin functions were implemented as virtual devices with real behavior. This test equipment makes tests without a real car or part of an automotive system possible. However, the reconfigurable system is currently being implanted in a real car for running the system in a real-world scenario with real functional devices.

A. Bus Macro Structure

For faultless data communication, it is necessary to implement a structure with fixed signal lines. Fig. 12 shows the schematic of an implementation of several modules without any interfaces between the reconfigurable areas. Signal lines may be open or two output ports of a module may even be connected together. This causes damage to the chip, and therefore, communication elements with fixed connection points and wiring are necessary. Fig. 13 shows a system with generalized communication interfaces. As mentioned, fixed connection points for each module are now established. The router connects while the system is processed, all signals on a defined position. Generalizing these interfaces also enables the interchange of modules between their positions. For an undisturbed data transfer, it is not enough to simply fix the connection points in a reconfigurable system. While this is important, it is also necessary to fix the physical signal lines that are used in each reconfigurable area. During reconfiguration, other modules run in parallel and may communicate via the bus.

A change of routing can cause a glitch or even a loss of information during reconfiguration. A variety of tests were

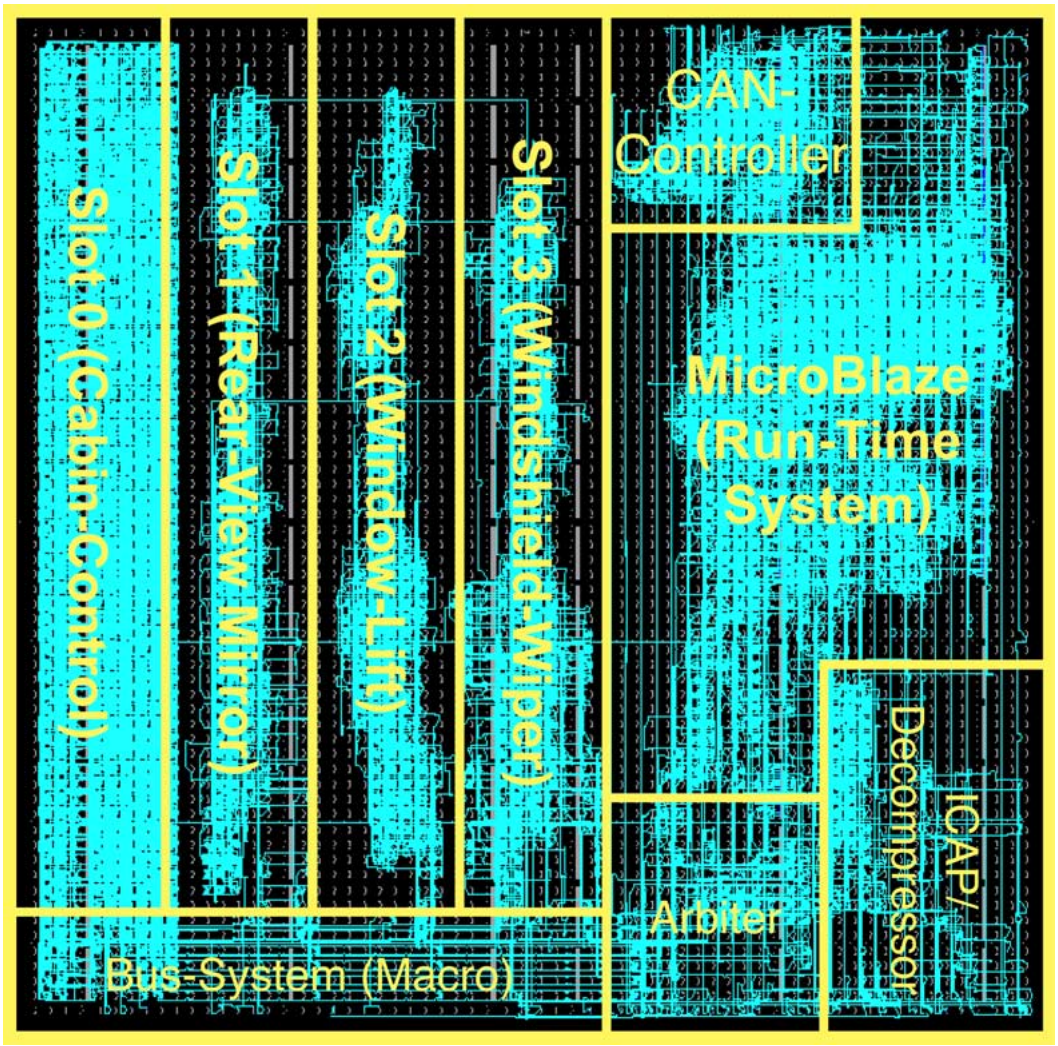

Fig. 10. ***Implemented system on XC2V3000 Xilinx Virtex-II FPGA (displayed with Xilinx FPGA Editor).***

realized replacing functionality with reconfiguration. Bus-macros are a combination of communication interfaces and fixed signal lines. Fig. 14 shows a macro which is used for the hardware reconfigurable system. Reference [11] describes the functionality of this interface/signal wiring macro and its method of development.

V. REAL-TIME RUN-TIME SYSTEM FOR AUTOMOTIVE DOMAIN

A real-time run-time system for the automotive domain, using new reconfigurable technologies and their benefits, combined with intelligent mechanisms for scheduling tasks on the hardware, is the goal of this work. The reduction of power dissipation by adapting the hardware to the actual demand of the application is possible with reconfigurable hardware. The outsourcing of functionality to external memory, and therefore, the possibility of using a smaller FPGA enables savings in power dissipation. In the presented system for automotive inner-cabin functions, the power dissipation changes with the value of around 6.2 mW within the time frame of 3 ms while the reconfiguration process. The integration of all inner-cabin functions of the demonstration system to a suitable FPGA, would require a Virtex-II 4000 FPGA which has an increased power dissipation of around 10 mW. These values where calculated by real measurement and the tool XPower which is provided by the Xilinx toolset. Additionally, dynamic power consumption by the usage of the increased chip size also leads to an increased power dissipation which is obvious in this scenario. An intelligent scheduling mechanism within the run-time system loading functions on demand to the reconfigurable platform (FPGA) helps reduce the number of different control systems within a car. The need and benefit of using FPGAs is that the number of parallel tasks increases by

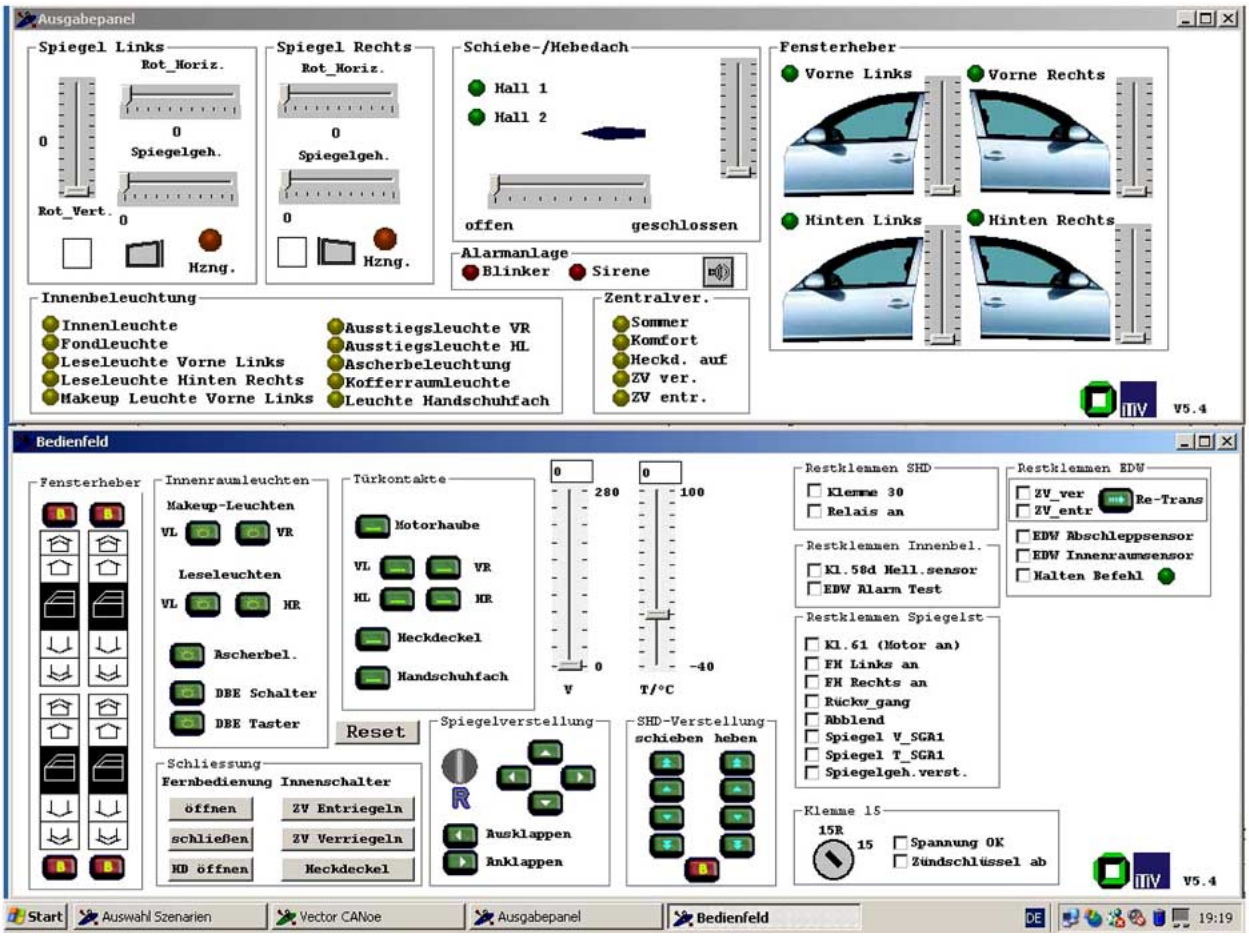

Fig. 11. ***Testbench for automotive functions.***

implementing a variety of functions in one chip. State-of-the-art processors in the automotive domain do not have the performance to run these tasks quasi-parallel. FPGAs are able to run all these tasks of scheduling, communication interfaces, and data transfer between the functions on high-performance hardware in parallel. The system

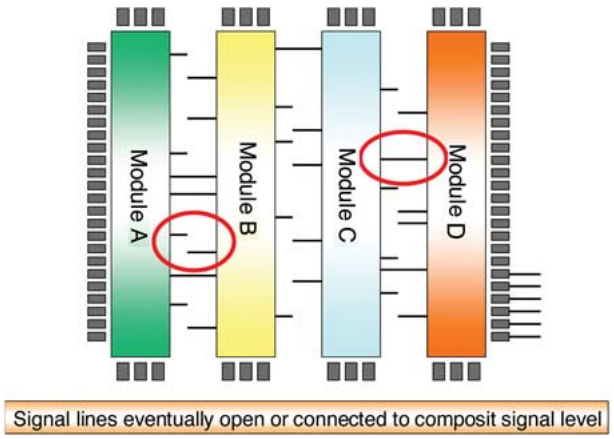

Fig. 12. ***Necessity of Bus Macro.***

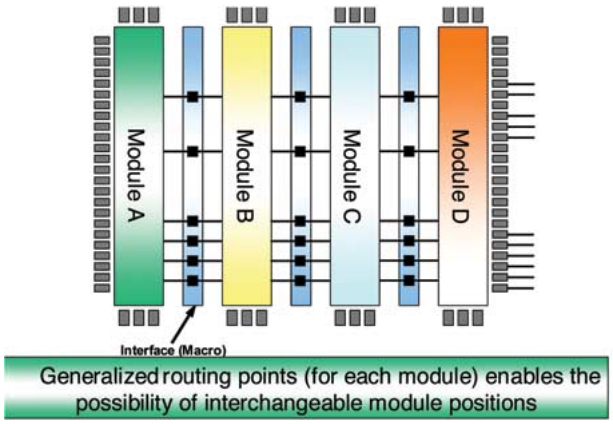

Fig. 13. ***System with communication interfaces.***

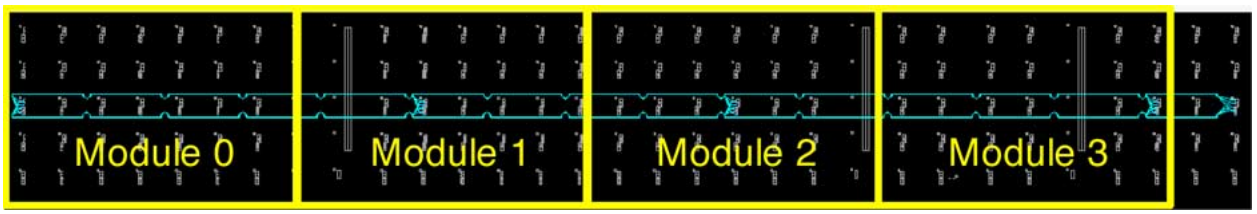

Fig. 14. ***Bus Macro connecting four modules.***

described here is able to fulfill all real-time constraints for the given applications and has the capacity for further fields of application in the automotive domain.

The further possibilities for adaptation, such as changing topologies for data transfer, clock gating techniques for optimizing power dissipation, and the scheduling of tasks which can be analyzed during run-time by the run-time system, enables the optimization of the complete system to the actual status and demand.

Fig. 15 shows an example of a reconfigurable system designed with Xilinx Embedded Development Kit (EDK, [20]). Via the On-Chip-Peripheral Bus (OPB-Bus), the parameterizable IP-Cores are connected to the microprocessor (MicroBlaze). Using the graphical abstraction layer of a hardware/software development system, like EDK, allows the system designer the combination of specific components on an abstraction level without time consuming VHDL coding. As an example for automotive applications, here the different car models and series will be named; e.g., a convertible has as the special function of the folding roof included in the ECU. By drag and drop, the functionality is integrated into the design and can be handled during run-time. An important issue is the parameterization of the IP-Cores during design time. Different car models have for example different parameters within their window lifter. Because of differing construction parameters, the way between the upper and lower bound of the windows also differs. Through parameterization, the functionalities of the window-lifter can be adapted to the requirements for this car model by adjusting the drive-way of the window. This easy method for adjusting different parameters is also a benefit during the testing phase of newly developed functionalities and allows an easy and time-saving turn-around time for rapid prototyping. A first approach for this design-flow is described in [23].

VI. AREA-TIME DEPENDENCY AND SYSTEM MODEL

Fig. 16 shows the dependency between reconfiguration, area, and time. During run-time, reconfiguration might be initiated if a request via the bus interface arrives at the run-time system. In the given example, the user starts with the request of a window-lift which is immediately configured into module-slot 3. Later, the seat control is requested and integrated on-demand into module-slot 3. This example shows the dependency of the available configuration area and functionality. More and more the configuration area is occupied with tasks during run-time until all module-slots are utilized. Different from a fixed configuration dependency graph, e.g., for a fixed schedule of tasks to be integrated to the reconfigurable system, the requested function during operation time cannot be foreseen. The requested functionality on chip depends on the actual requirements of the user and has no fixed schedule. Therefore, idle tasks need to be substituted by actual recommended functions. One criterion for substitution is certainly the available reconfigurable area and, therefore, one of the most important factors for modeling the reconfigurable system. Tasks were not substituted until all reconfigurable areas are utilized. Idle tasks, which are not used at the moment, are candidates for substitution with an actual requested function if all module-slots are occupied with tasks. By introduction of priorities related to different functionalities, another criterion for the system model is given; e.g., a windshield-wiper can be more important than a window-lift. It has to be considered that these priorities also have run-time related values. While driving, controlling a rear-mirror is very important for the secureness of the passengers and has a higher priority. When the car is in standby, this functionality gets a lower priority and so forth. These examples show that the configuration process and the decision adapting the system are a multidimensional optimization process during run-time, handled and finally decided by the run-time system. Additionally the criteria quality of service, real-time constraints, and power consumption needs to be considered by analysis of the dependencies between the different functionalities while run-time. This can be achieved by introducing of a dynamic configuration dependency graph (DCDG) which can be used as a basis for further optimization algorithms.

As described above, the sequence for configuring the module-slots cannot be foreseen. Internal parameters, e.g., power-consumption or resource allocation, as well as external requests influencing the sequence and utilization of the hardware have to be faced. As a basis for the reconfiguration and substitution process during run-time, the example of the substitution graph for automotive function will be introduced.

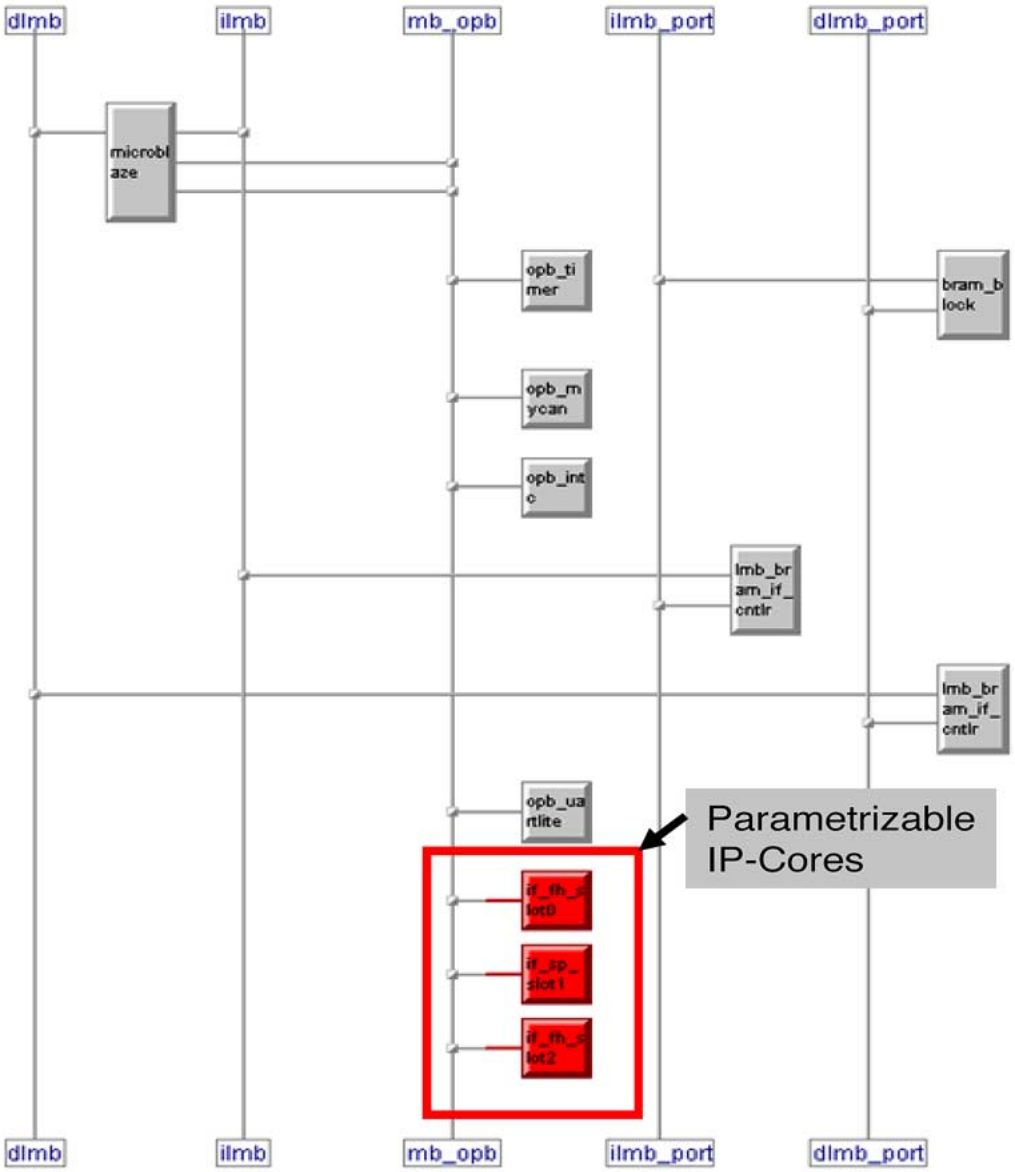

Fig. 15. ***Graphical view to embedded system in Xilinx EDK.***

Fig. 17 shows an example of a substitution graph for automotive inner cabin functions. The arrows between the different functions assign that these functions can substitute each other. The easiest example can be shown with the alarm function. The alarm function is not used if the driver requires the window-lift function. Certainly the door-lock function needs to be available if the alarm is active to open the car from the outside by the user, therefore, no arrow connects these functions within the graph.

It is clear that this graph also is run-time-dependent since, e.g., the alarm function can definitely be substituted while the car is in motion. The complexity of the relationship described in the substitution graph depends on the parameters which have to be processed within the reconfigurable system. The substitution graph which is designed statically, influences directly the configuration process and leads, therefore, to the introduction of the DCDG which describes the dependencies and influencing factors for the run-time adaptive system.

VII. CONCLUSION AND FUTURE WORK

This paper presents the system and design methodology of dynamic and partially reconfigurable systems for usage in the automotive domain. The proof of concept has shown that FPGAs, with the feature of dynamic and partial reconfiguration, can be exploited to reduce power

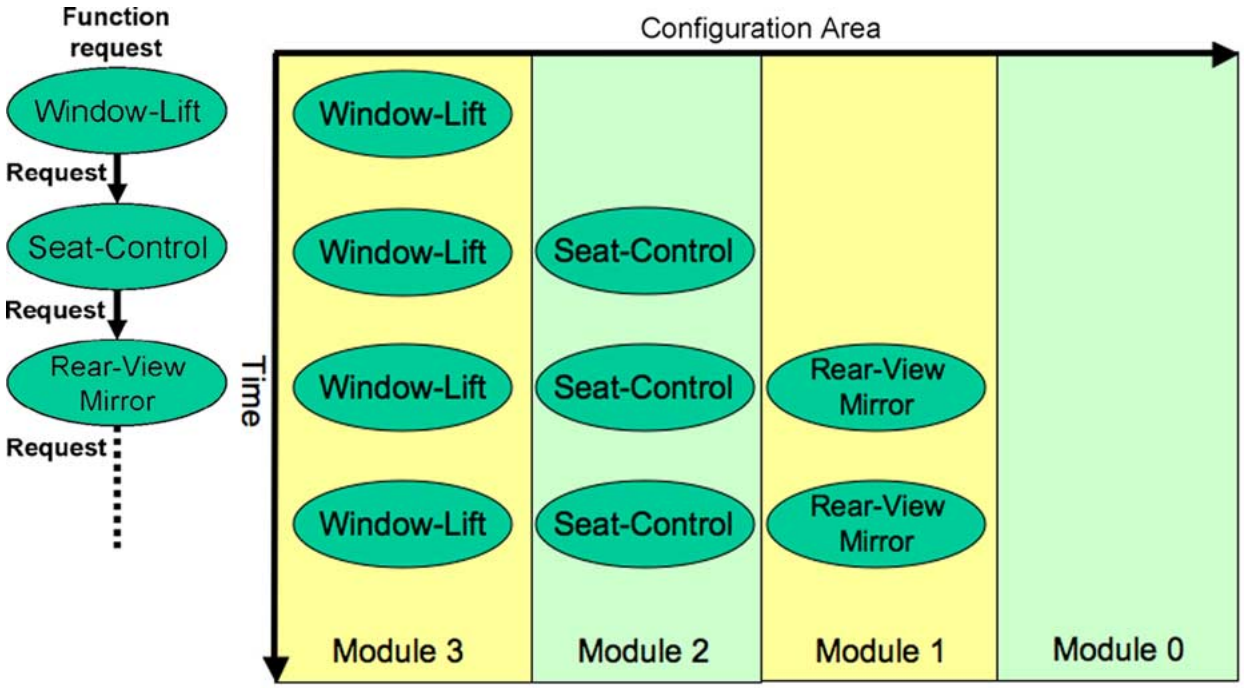

Fig. 16. ***Time-area dependency while run-time.***

dissipation and increase the adaptivity of embedded systems. Intelligent mechanisms within a run-time system can optimize the performance, and therefore, the power dissipation depending on the actual status of a system. To overcome performance problems for systems with high data throughput, reconfigurable computing will be the trend for future high-performance systems [2]. Increased data throughput caused by novel high-performance communication systems (e.g., FlexRay [7]) can be handled efficiently by parallel working reconfigurable hardware where traditional microprocessor solutions need to work with a high clocking rate. The benefit here is to reduce power consumption by parallelization as described in [16].

Future work is to develop systems, which can be used in other areas of application using the benefits of dynamic and partial reconfiguration. The goal is to use intelligent

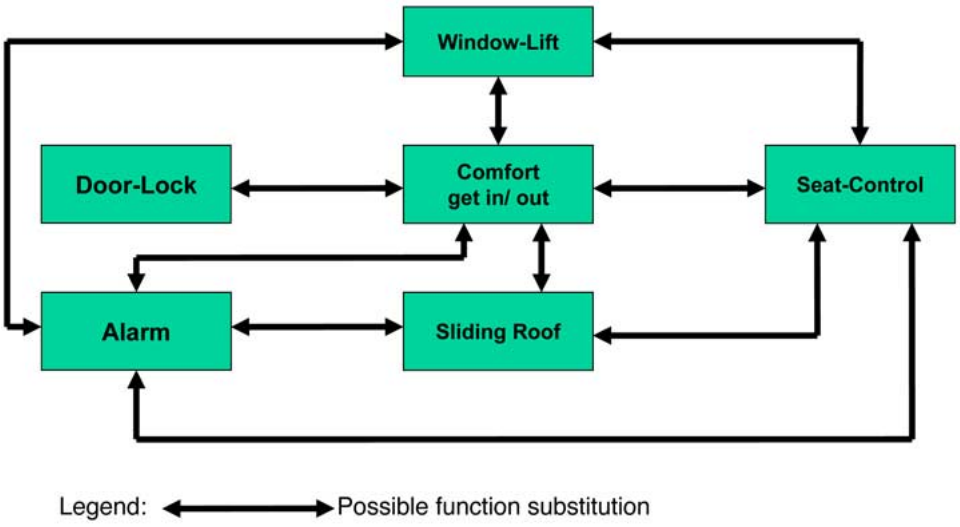

Fig. 17. ***Example substitution graph for automotive inner cabin functions.***

run-time systems, which access all parameters given by the hardware and the architecture for optimizing systems during run time. Additionally, new methodologies for accessing the reconfiguration area will be developed. In future it will be possible to use algorithms for dynamic synthesis, mapping, placement, and routing on chip, to utilize rectangular areas on the chip area during run-time. This enables the saving of external memory and adaptation of the shape of areas for functions on the FPGA during run-time. This increases the cost effectiveness for such systems by moving closer to an ideal utilization of configurable elements [24]. The development of such systems is only possible using the presented LUT-based communication elements, since dedicated connection points avoid routing across the borders of a rectangular shaped reconfigurable area. In addition, the new Xilinx FPGA, like Spartan-III does not include TBUF (Tristate Buffer) elements which were used in previous approaches to establish a macro as a communication interface. Therefore, using the presented approach, other Xilinx FPGA series can be used. Also new low-cost, low-power architectures for integration as automotive FPGA-based ECUs have to be evaluated since costs and power consumption are tremendous factors in this field of application. ■

REFERENCES

[1] J. Becker, M. Huebner, and M. Ullmann, "Real-time dynamically run-time reconfiguration for power-/cost-optimized Virtex FPGA realizations," presented at the IFIP Int. Conf. Very Large Scale Integration 2003, Darmstadt, Germany.

[2] J. Becker and R. Hartenstein, "Configware and morphware going mainstream," *J. Syst. Architecture (Special Issue on Reconfigurable Systems)*, vol. 49, pp. 127–142, Oct. 2003.

[3] J. Becker, "Configurable systems-on-chip (CSoC)," in *9th Proc. of XV Brazilian Symp. Integrated Circuit Design (SBCCI 2002)*, Porto Alegre, Brazil, Sep. 5–9, 2002, (Invited Tutorial).

[4] B. Blodget, S. McMillan, and P. Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs," in *Proc. Design, Automation and Test Eur. Conf. and Exhibition (DATE'03)*, vol. 1, p. 10 399.

[5] K. Etschberger, *Controller Area Network—Basics, Protocols, Chips and Applications*. Weingarten: IXXAT Automation, 2001, 3-0000-7376-0.

[6] K. Camera, "SF2VHD: A stateflow to VHDL translator," Masters thesis, Univ. California, Berkeley, CA, Spring 2001.

[7] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, no. 1, pp. 88–93, Jan. 2002.

[8] R. Hartenstein, "A decade of reconfigurable computing: A visionary retrospective," in *Design, Automation and Test in Eur. Conf. and Exhibition (DATE 2001)*, Munich, Germany.

[9] M. Huebner, M. Ullmann, L. Braun, A. Klausmann, and J. Becker, "Scalable application-dependent network on chip adaptivity for dynamical reconfigurable real-time systems," in *14th Int. Conf. Field Programmable Logic and Applications (FPL 2004)*, Antwerp, Belgium.

[10] M. Huebner, M. Ullmann, F. Weissel, and J. Becker, "Real-time configuration code decompression for dynamic FPGA self-reconfiguration," in *11th Reconfigurable Architectures Workshop (RAW '04)*, Santa Fé, NM.

[11] M. Huebner, T. Becker, and J. Becker, "Real-time LUT-based network topologies for dynamic and partial FPGA self-reconfiguration," in *17th Brazilian Symp. Integrated Circuit Design (SBCCI 2004)*, Porto de Galhinas, Pernambuco, Brazil.

[12] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins, "Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs," in *12th Int. Conf. Field Programmable Logic and Applications (FPL 2002)*, Montpellier, France.

[13] P. Lysaght, "Future design tools for platform FPGAs," in *Proc. 16th Symp. Integrated Circuits and Systems Design (SBCCI '03)*, Sao Paulo, Brazil.

[14] A. Tewari, *Modern Control Design With MATLAB and SIMULINK*. New York: Wiley, Feb. 2002, 0-471-49679-0.

[15] V. Baumgarten, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP—A self-reconfigurable data processing architecture," in *Int. Conf. Engineering of Reconfigurable Systems and Algorithms (ERSA 2001)*, Monte Carlo Resort, Las Vegas, NV.

[16] Rabaey, "Reconfigurable processing: The solution to low-power programmable DSP," in *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP 1997)*, Munich, Germany.

[17] A. Thomas and J. Becker, "Dynamic adaptive routing techniques in multigrain dynamic reconfigurable hardware architectures," in *14th Int. Conf. Field Programmable Logic and Applications (FPL 2004)*, Antwerp, Belgium.

[18] M. Ullmann, M. Huebner, B. Grimm, and J. Becker, "An FPGA run-time system for dynamical on-demand reconfiguration," in *11th Reconfigurable Architectures Workshop (RAW 2004)*, Santa Fé, NM.

[19] O. Falkner, "Die ideale Werkzeugkette für LIN und CAN—Vom Entwurf bis zur hard-/software-integration," *Elektronik Automotive, Sonderheft LIN*, 2004.

[20] R. Pragasam, "Design embedded programmable systems without compromise," *Xilinx XCell J.*, no. 45, Spring 2003.

[21] J. Becker, M. Hübner, K. D. Müller-Glaser, R. Constapel, J. Luka, and J. Eisenmann, "Automotive control unit optimization perspectives: Body functions on-demand by dynamic reconfiguration," in *Design, Automation and Test Eur. Conf. Exhibition (DATE 2005)*, Munich, Germany.

[22] H. Heinecke, K. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Mate, K. Nishikawa, and T. Scharnhorst, "AUTomotive Open System ARchitecture—An industry-wide initiative to manage the complexity of emerging automotive E/E-Architectures," Convergence Transportation Electronics Association, 2004.

[23] M. Hübner, K. Paulsson, M. Stitz, and J. Becker, "Novel seamless design-flow for partial and dynamic reconfigurable systems with customized communication structures based on Xilinx Virtex-II FPGAs," in *18th Int. Conf. Architecture of Computing Systems (ARCS 2005)*, Innsbruck, Austria.

[24] J. Becker, M. Hübner, K. Paulsson, and A. Thomas, "Dynamic reconfiguration on-demand: Real-time adaptivity in next generation microelectronics," in *Reconfigurable Communication Centric-SoCs Conf. (ReCoSoc 2005)*, Montpellier, France.

[25] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, Jun. 2002.

[26] S. Brown and J. Rose, "FPGA and CPLD architectures: A tutorial," *IEEE Des. Test Comput.*, vol. 13, no. 2, pp. 42–57, Summer 1996.

ABOUT THE AUTHORS

Jürgen Becker (Senior Member, IEEE) is a Full Professor of Electrical Engineering and Information Technology at the Universität Karlsruhe (TH), Germany. He is Chair of the GI/ITG Technical Committee of Architectures for VLSI Circuits and codirector of the International Department, Universität Karlsruhe (TH). He is Managing Director of the International Department, Universität Karlsruhe (TH) and Department Director of Electronic Systems and Microsystems (ESM), Computer Science Research Center (FZI). In October 2005, he was elected Vice-President ("Prorektor") of the Universität Karlsruhe (TH). He is the author or coauthor of more than 150 scientific papers, and is active as the general and technical program chairman of national/international conferences and workshops. His actual research is focused on industrial-driven system-on-chip (SoC) integration with an emphasis on adaptivity, e.g., dynamically reconfigurable hardware architecture development and application in automotive and communication systems.

Prof. Becker is a member of several program committees of international conferences, and Associate Editor of IEEE TRANSACTIONS ON COMPUTERS. He is an executive board member of the German IEEE section.

Michael Hübner (Student Member, IEEE) is working toward the Ph.D. degree in electrical engineering and information technology at the Institute for Information Processing Technology (ITIV), Universität Karlsruhe (TH), Germany, where he has been since 2003. He received the diploma degree in electrical engineering and information technology in 2003 from the same University.

His research interests are in reconfigurable computing, and particularly new technologies for adaptive FPGA run-time reconfiguration and on-chip network structures with application in automotive systems, including their integration into high-level design and programming environments.

Gerhard Hettich received the degree in electronics from the University of Applied Science, Esslingen, Germany, in 1970, and the Masters degree in physics and the Ph.D. degree in solid state physics, both from the University of Stuttgart, Germany, in 1979.

He is Director of Electric/Electronic Systems and Components at DaimlerChrysler AG Research and Technology, Germany, where he has been since 1999. From 1980 to 1989 he was Manager of Development at the Robert Bosch GmbH, and before joining the DaimlerChrysler AG, he was Chief Engineer for the Temic Telefunken Microelectronic GmbH.

Rainer Constapel received the diploma degree in electrical engineering from the University of Duisburg, Germany, in 1986, and the Dr.-Ing. degree in the field of numerical simulation techniques from the Fraunhofer Institute of Microelectronic Circuits and Systems, Duisburg, Germany, in 1991.

He is Head of the Electric/Electronic Comfort and Systems Department within Mercedes-Benz passenger car development. His research interests are electronic control unit (ECU) design, vehicle networking, and communication systems.

Joachim Eisenmann received the electrical engineering degree in 1985 at the Technical University of Stuttgart, Germany.

He is a Manager in the Laboratory of Electric/Electronic Components and Systems in Daimler-Chrysler Research and Technology, Germany. In 1991, he joined the Research Department of the former Daimler-Benz AG. After working in several projects, he is has been responsible for Electric/Electronic Topologies in research and predevelopment since 2001.

Jürgen Luka studied mechatronics at the University of Heilbronn.

He is a Manager at DaimlerChrysler Research and Technology and is responsible for new electric/electronic systems within the cabin area. He joined DaimlerChrysler in 1985, where he developed Electronic Control Units for passenger cars and commercial vehicles. In 1992, he entered the Research and Technology Division as a project leader for on-off and mobile diagnosis based on expert systems. Since 2001, he has been responsible for research projects which aim to produce scalable electronic control units, utilizing reconfigurable hardware elements. He is a member of the program committee of the international conference Reconfigurable Architectures Workshop RAW.