Digital On-Demand Computing Organism for Real-Time Systems

Jürgen Becker[1], Kurt Brändle[1], Uwe Brinkschulte[2], Jörg Henkel[3], Wolfgang Karl[3], Thorsten Köster[1], Michael Wenz[2], Heinz Wörn[2]

[1] Institut für Technik der Informationsverarbeitung (ITIV)
Universität Karlsruhe (TH)
76131 Karlsruhe, Germany
{becker, koester}@itiv.uni-karlsruhe.de

[2] Institut für Prozessrechentechnik, Automation und Robotik (IPR)
Universität Karlsruhe (TH)
76131 Karlsruhe, Germany
{brinks, mwenz, woern}@ira.uka.de

[3] Institut für Technische Informatik (ITEC)
Universität Karlsruhe (TH)
76131 Karlsruhe, Germany
henkel@informatik.uni-karlsruhe.de, karl@ira.uka.de

Abstract. This paper presents the goals and the research approach of the project DodOrg (Digital On-demand Computing Organism for Real-Time Systems) within the Priority Programme "1183 - Organic Computing". The main goal of the project is the development of a biologically motivated computing organism with the three levels "brain", "organ" and "cell". On the brain level an organic robot control will be developed with emphasis on self-x features. The robot control forms together with the middleware on the organ level and the reconfigurable hardware on the cell level an organic computing architecture.

1 Introduction

The project DodOrg focuses on the "self-x" properties that are essential to "Organic Computing" (e.g. self-organization, self-configuration, self-healing and self-protection). Similar properties can be found in various fields of biology, suggesting a comparison between technical systems and living organisms. This observation led us to an interdisciplinary formation of authors' working groups, tightly integrating knowledge from various areas such as robotics, computer science, electrical engineering and –most important– biology. The goal of the project, however, is not to copy nature, but rather to study the strategies organisms have developed to achieve their aims, in order to find out whether similar strategies can be build in technical systems.

We start with the consideration of the "self-x"-properties from a biological point of view before we discuss possible technical solutions.

During evolution, living organisms have found an immense number of solutions for different problems. Moreover, in various groups of plants and animals different solutions for the same problem were developed, e.g. respiration in Insects, Fishes, Amphibians and Mammals. Unfortunately many of these solutions are either impracticable or irrelevant for hard- and software development. Therefore the main task of "Organic Computing" consists in finding close resemblance of related problems in nature and computer systems.

For our application we selected the mammalian heart (Fig. 1) as a model comparable to our envisaged organic computing architecture, because it fits in many aspects very well to the project: In both cases the subsystems are organized in a hierarchical order showing similar characteristics. Some features overlap all levels, such as low power consumption or fail safe features. Nevertheless the selection of the heart as a model is arbitrary, because other examples, such as the coordination of limb movements in Amphibians or the processing of visual information in lower vertebrates, would have been just as good for comparison.

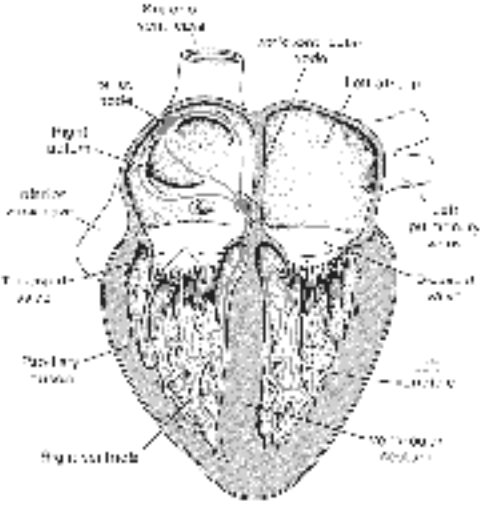

Fig. 1: The Mammalian heart as model for DodOrg research

In the mammalian heart the muscle cells form the lowest level of the system. Although their contraction mechanism is most complex, their answer to electrical stimulation is remarkably rigid: When an electrical impulse reaches the outer cell membrane, it causes an inversion of the membrane potential (action potential), resulting in a contraction of the muscle fibre. The action potentials are different from those in skeletal muscles: their plateau persists for a prolonged time to make sure that all muscle cells of a large area contracts almost simultaneously. Similar the pause after a depolarization is prolonged to prevent tetanic contractions. At the same time the depolarization spreads on neighbour cells.

The next higher level is the contraction control of the whole heart. It coordinates the contraction sequence in time. Activity initiated in a pacemaker region in the wall of the atrium (sinus node) spreads over the muscles of both atria causing their simultaneous contraction. The atria are connected electrically to the ventricles only by a second pacemaker, the atrio-ventricular node. After a short delay the wave of excitation is conducted very fast via transitional fibers of HIS to the tip of both ventricles. Starting there the contraction of the ventricles spreads back to the border between atria and ventricles. The functional significance of the electrical organization of the heart muscle is its ability to generate separate, synchronous contractions of the atria and ventricles, following one another. Because the basic heart control resides in the heart itself, the heart beat continues even in case the heart is isolated from the body.

Nevertheless, the heart function can be modified to meet altered demands of the body by two antagonistic parts of the vegetative nervous system: The sympathetic system increases heart rate and stroke volume, the parasympathetic system decreases the cardiac output. The activity of the vegetative nervous system is influenced by the metabolic activity of the body as well as by psychic factors as excitement or fear.

2 State of the Art

This project covers topics from the fields of hardware architectures, low power design, monitoring, self-organization of distributed real-time systems and robotics.

2.1 Reconfigurable Hardware Architectures

Since application algorithms can be in principle implemented in software or in hardware, i.e. either completely in time or completely in space, reconfigurable computing means structural programming and reconfiguration concurrently in time and in space. Data stream schedules have to be optimized within and at the I/O borders of such kind of array architectures [BH03]. Dependent on the granularity and bitwidth of the reconfigurable data manipulating logic and arithmetic units, the corresponding hardware architectures are called fine-grain, coarse-grain, and multi-grain [BBU05]:

- Fine-grain FPGA: Complex data manipulations are decomposed into single-bit logic operations that match the available configurable logic blocks. The drawback is that fine-grain FPGAs lack mainly area/power-efficiency, since their physical integration density is roughly two orders of magnitude worse than the Gordon Moore Curve, whereas the logical integration density is about 4 orders of magnitude behind Gordon Moore [D00].
- Coarse-grain FPFA (Field-Programmable Function Arrays): Complex data manipulations are decomposed into multiple-bit arithmetic operations that match the available reconfigurable datapath units (rDPUs). Coarse-grain reconfigurable hardware solutions are about one order of magnitude more energy-efficient than fine-grain alternatives [BHP05].
- Multi-grain FPMA (Field Programmable Digital-Analog Mixed Arrays): Complex data manipulations are decomposed into single- and multiple-bit logic,

arithmetic, and bitlevel operations that match the available mixed processing elements (MPEs). This "hybrid" architecture variation reflects combination of fine- and coarse-grain operation types found in many application algorithms.

2.2 Ultra Low Power Design

Dynamic power management (DPM) is a design methodology that dynamically reconfigures an electronic system to provide the requested services and performance levels with a minimum of active components or a minimum load on such components as stated in [BB00]. The basic concepts of DPM i.e. identification and definition of power states and (power consuming) transitions between these states and first heuristic optimization strategies with application to embedded systems have been proposed in [BM97][P01]. Early work has focused on known stationary effects [BBP99] i.e. the transition matrix representing state transitions is time-invariant for a stationary workload. More relevant for embedded systems, however, are unknown stationary environments i.e. an offline analysis is not sufficient. Control theory [H89] has been applied to address these scenarios. Stochastic learning with application to ACPI compliant devices has been proposed. From an application point of view, power management has also been applied to distributed computing systems. In [CR03] the paging of a wireless system is combined with power management strategies.

2.3 Monitoring

An important issue with self-organization is system monitoring. Common used techniques for monitoring include hardware probes and software instrumentation. The former is non-intrusive, while the latter is flexible, easy-to-implement, but with overheads. In the hardware area, monitors are usually deployed either at processor level or at system level. In the first case, they occur in the form of performance counters and are integrated in the processor core. Actually, most modern processors provide such counters. Well-known examples are the IBM POWER2[W94] that has 5 performance counters enabling the concurrent monitoring of five events and Intel Itanium Architecture [INT02] which supplies 8 counters to allow the collection of more information. At system level, usually specific hardware has to be developed in order to meet individual requirements and purposes. Examples include the Princeton hardware monitor for SHRIMP multiprocessor [KC99] and the Stanford hardware performance monitor for the DASH multiprocessor [L93].

2.4 Self-organization of distributed real-time systems

Self-organization has been a research focus for several years. Publications like [J89] deal with basic principles of self-organizing systems, like e.g. emergent behaviour, reproduction etc. Regarding self-organization in computer science, several projects and initiatives can be listed. IBM's Autonomic Computing project [KC03] deals with self-organization of IT servers in networks. Several so called self-X properties

like self-optimization, self-stabilization, self-configuration, self-protection and self-healing have been postulated. The German Organic Computing Initiative has been founded in 2003. Its basic intention is to improve the controllability of complex embedded systems by using principles found in organic entities [VDE03]. Organization principles successful in biology shall be adapted to embedded computing systems.

Regarding self-organization of distributed embedded real-time systems, not much work has been done yet. On one side, the relationship between self-organization and real-time to our knowledge is poorly researched. On the other side, in the field of self-organization for distributed systems, more work has been done. Middleware is a key component in organizing distributed systems. For standard applications middleware architectures like CORBA, DCOM, .NET, JMS or RMI are state of technology. Some of these systems are suitable for real-time applications, e.g. RT-CORBA [OM03]. The resource needs of the middleware architectures mentioned above is quite high. So it is not possible to use them on small computation nodes like microcontrollers due to memory and computing power limitations. To reduce these needs and to make middleware applicable to embedded systems, several work has been done. Regarding self-organization, current middleware approaches provide features for load balancing. Middleware architectures fulfilling organic computing principles are rare. In [BFL03], the use of middleware for self-healing is investigated. In [TBP03], a self-organizing middleware to manage smart doorplates is described.

2.5 Application domain for organic computing: Robot-based manufacturing

The increasing degree of functionality of production systems with a concomitant increase in degree of complexity of these systems, leads to the requirement for highly efficient production systems. Thereby robots play a dominating role. In the area of robot based manufacturing, the production processes and equipment are very complex with a hierarchical multi-level structure, especially in the automotive industry. Changes in production cause time and cost expensive reconfiguration and adaptation of control parameters. Faults during automatic production can result in considerable losses of the product quality or additional production costs e.g. due to maintenance or standstill periods. Moreover, faults in the robot environment of a cell can considerably endanger human health. Thus, around this field of application a very important issue will be in future the overall multi-sensoric supervision and control of the complete process to achieve goals of higher accuracy and more efficiency as well as to guarantee safety even in the case of component malfunctions or faults of the human operator.

3 The DodOrg Architecture - from the Organic Model to a Technical System

In this section the overall architecture of the project and the relations between the different sub areas are described. Starting with basic biological mechanisms, we derive a digital, on-demand computing organism representing three levels.

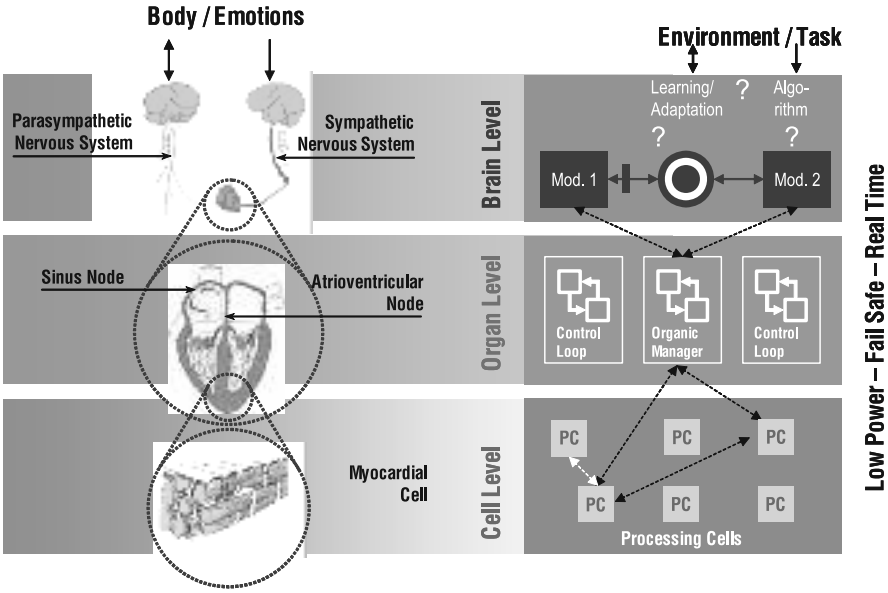

Fig. 2: From Biology towards an Organic Computing System

Fig. 2 puts the concept of the mammalian heart on the left side in relation to the computing system architecture we envision. Three levels can be distinguished: Cell level, organ level and brain level. These levels and their interrelation correspond to a natural partitioning of this project:

- A robot control architecture is an example of a complex real-time system. It can be regarded as the brain of a self-organizing robot. It controls the way in which the robot performs its tasks and reacts to his sensor inputs. The robot controller comprises the basic control algorithms within the robot system itself, e.g. control of drives, sensors, technology, digital IOs, etc. The robot components are integrated via high-performance industrial networks, e.g. field buses or Industrial Ethernet. The architecture for self-organisation and self-configuration in the robot system is strongly demanding, because reconfiguration of components and control policies has to be done during the run-time of the system.

- In biological organisms, cells specialize and group together to form organs. In our organic architecture, this will be achieved by a middleware. The task of this middleware is to dynamically group processing cells and to assign to them parts of the application resulting in something like “virtual organs”. Furthermore, the middleware is responsible for several self-X features on the organ level. The challenge is to research and develop structures to realize these features. In order to allow self-organization and self-optimization, closed control loops and organic management must be present. To enable self-healing, these closed control loops and organic management must be totally decentralized and reconfigurable, so they do not break when processing cells fail.

- To facilitate closed control loops, flexible monitors are necessary on each level to observe vital system parameters. These monitors are responsible to gather state information of the entire system, which is needed by the middleware to dominate the cells and to configure them on-demand. As the monitors are connected to various system levels, including hardware, middleware, and application, specific requirement arises, especially in flexibility and reconfigurability. Hence, we propose monitors that operate independently of the target modules, much as the autonomic nervous system operates independently of the human consciousness. Each monitor will work as an independent device, which snoops on a bus or communication wire, or which observes system resources in the middleware and application level. Each monitor itself is reconfigurable, allowing to dynamically adapt its activity to changing system conditions. These monitors will be developed both in the form of software and hardware, and distributed throughout the complete system. At the cell level, hardware probes will be applied to snoop buses and information about e.g. energy consumption, cell utility, malfunctions, and abnormal operations will be delivered. At the middleware level, software probes will be used to extract information from the components included in the control loops and information about the efficiency of the middleware in terms of various target functions will be gathered. At the application level, the monitors are capable of acquiring, filtering, and processing any kind of state information and events delivered by the applications, and then generating essential information for application-specific adaptation, for example, fault-tolerance control. Additionally, standardized API will be provided to control the monitors and to query the performance data uniformly across all monitors in the system.

- Since we envision, again comparable to the cells of an organism, to include a large number of processing cells in our organic architecture, power consumption is a key issue. Low power design is a key issue for designing future embedded systems generations. Design for ultra low power consumption will allow applying embedded systems virtually everywhere as it increases the operation time of mobile devices between re-charge cycles. In our proposed organic computing mechanism, we will deploy what we call a swap-on-the-fly system architecture in order to minimize the power consumption of our organic computing mechanism. It is a technique that dynamically interchanges processing resources at the cell and organ level in order to ensure a minimum of power consumption under the momentarily valid constraints (e.g. performance). The swap-on-the-fly system architecture provides a swap power manager, a swap scheduler and a swap execution unit as basic components. It receives monitoring data, aligns the schedule with the coarse-grained system schedule and utilizes the adaptive hardware as one possible option for swapping-on-the-fly.

- On the level of processing cells, comparable to the cells of an organism, flexible, adaptive, dependable and high performance hardware is necessary. Reconfigurable hardware architectures, based on a common infrastructure, promise to be an appropriate solution to achieve this goal. Various types of hardware cells allow for self-induced grouping via an extensible network, providing various “ser-

vices" to the whole of the system, such as (but not limited to) calculation, memory, interaction with the environment and monitoring of other cells. Seamless shifting of tasks and responsibilities from one cell to another in case of failure and self-monitoring of cells are needed for true de-centralization of the system and provide challenges for research.

The overall architecture is organized in five categories addressing particular areas of research. There are two hardware-centric, two software-centric, and one mixed hardware/software category. These categories are described in detail in the following.

3.1 Organic Processing Cells

The organic self-x hardware architecture proposed in this project represents a new kind of hardware and processor architecture family, which is conceptually derived from today's dynamically reconfigurable technologies. The architecture topology and hardware data path structures are continuously adapted and dynamically (self-) customized during runtime, in order to always have an optimally dimensioned hardware solution available with respect to minimal power consumption and resource availability. Monitoring, fault-detecting and -handling and reconfiguration must occur at regular intervals. An adaptive cell-based digital computing "organism" would functionally behave like a real organic being and fulfill these needs. To effectively realize the properties of future "organic" systems, a node-based approach to implementation seems viable (Fig. 3).

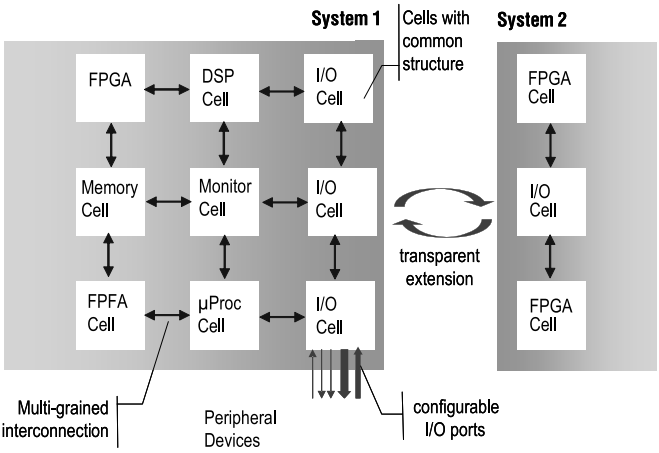

Fig. 3: Conceptual Hardware Architecture Overview

Modularity will be gained by having all cells of the "organic" hardware system share one of a set of common "blue prints". This corresponds to the notion of cells being derived from one basic type, only differentiated during development of the organism by special "programming" or interaction with neighbouring cells. True

"organic" behaviour could be achieved by only using FPGA cells, configured to the needs of the respective processes running on it. As pointed out above, this would render efforts of achieving low power consumption quite fruitless, though. In this project, this strict principle should therefore be mellowed. Cells will come "pre-packaged" with a certain special function, such as DSP or micro-processor functionality, FPGA functionality, FPFA functionality or I/O functionality, but the overall number of these variations should be kept to a minimum. In addition, this constraint will also make it easier for the system to find cells that can take over tasks from defunct neighbours. The necessary complexity of these functionality blocks will need to be established, as it depends mainly on the overhead for the common cell infrastructure. The trade-off that must be considered here lies between few but powerful cells (i.e. full-fledged processors) and many cells with only limited capabilities (e.g. reconfigurable data path units with sequencers). A system of mixed complexity would be possible as well.

All cells need to be able to do a minimum of housekeeping themselves. More complex tasks are taken over by the distributed organic middleware, but the basic interfacing to the hardware needs to be in place. We envisage a common structure of all cells that comprises a router, clock and power management, functionality monitoring, state and configuration control and cache memory, or possibly a subset thereof (Fig. 4).

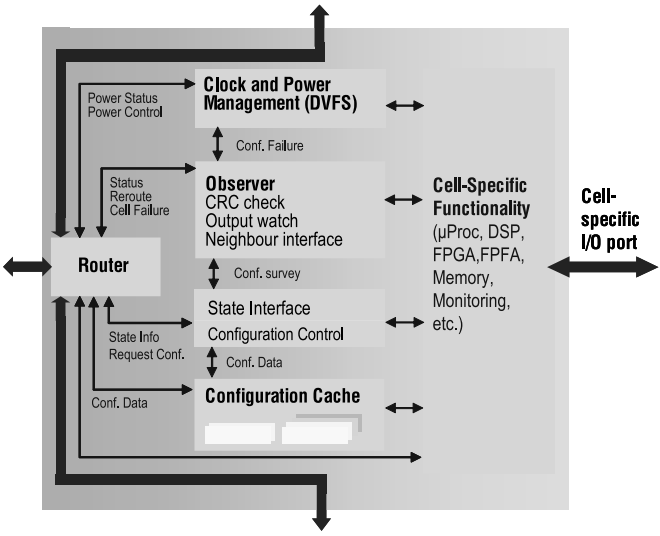

Fig. 4: Hardware structures common to all cells

The router is the basic building block of the whole cell network. It must circumvent defunct cells (self-rerouting), it needs to forward data to suitable successive cells as defined by the target application, thereby implementing self-establishing datapaths, and it needs to communicate the cell's status to neighboring cells and react to similar messages coming in from other cells (basis for self-reconfiguring behavior).

To carry out the decisions made by the Power Management layer, hardware support is needed on cell level. The clock and power management functionality shall be responsible for powering down cells when their functionality is not needed, and for applying dynamic frequency and voltage scaling techniques to the cell's content.

The observer circuit is the hardware part of the monitoring system. It checks for errors in program code or configuration data, employing checksum or hash functions, and it provides basic control of the cell's behavior, e.g. monitoring its output and rejecting possibly dangerous output events. The configuration control unit is responsible for reconfiguration of the cell specific block. For FPGA cells it will determine the slices that are to be reprogrammed dynamically [4], for processor cells it is responsible for loading application code to the proper memory location. It interfaces to the router for obtaining the current configuration data or program code.

To facilitate effective and efficient adaptivity with respect to cell failures (self-healing) and with respect to a reconfiguration of data-paths, a mechanism for transferring state information among the various cells is envisaged.

3.2 Low power through swap-based system architecture

A novel system architecture is proposed in order to minimize the all-over power consumption of future MPSoCs (Multi-Processor-System-on-Chip). According to our model (Fig. 5), this sub-area covers both the cell and organ level and requires monitoring input, utilizes the reconfigurable hardware and exchanges system information with the middleware.

In the past, energy savings at system-level have been achieved through switching between inherent power modes and adding additional modes to further extend the leeway for optimization. However, if a system has only a small amount of components with unfavorable power characteristics, there is only that much optimization potential that can be managed by the power management scheme.

Our Swapping-on-the-fly proposal will vastly increase the leeway for power optimization by offering a set of alternatives for implementations of a task. At any point in time, exactly one implementation of a certain task will be instantiated though. Here, the meaning of "implementation" is twofold: it may either denote the physical implementation ("software", "custom hardware", "reconfigurable hardware" etc.) or it may denote the algorithmic implementation.

In the first case, variations in power consumption result from the differing characteristics of the physical fabric the task is implemented on. It is obvious that a task implemented as a software program residing in a RAM-based memory has a different power consumption and performance compared to an ASIC-style implementation. In the second case, variations result from differences in computational complexity of algorithms, additional features of an algorithm etc. Eventually, further variations result from combining physical and algorithmic implementations.

Hence, we assume that each task may potentially be available on a SoC in one or more implementations resulting from combining physical and algorithmic options. Though this hardware/software architecture will result in a larger SoC area (i.e. gate equivalents) compared to the case of only a single implementation for each task, it

will add a new dimension for significantly reducing the power consumption of a SoC compared to an implement-once-and-fix-forever style of implementation.

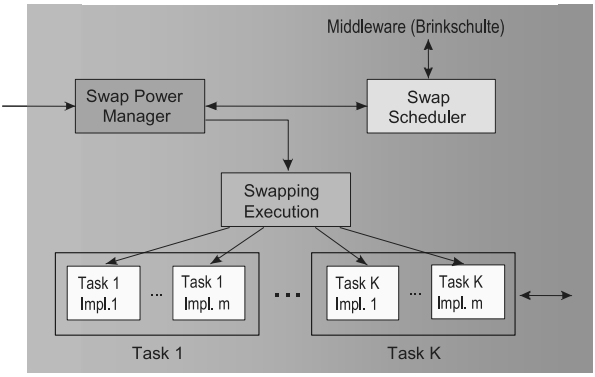

Fig. 5: Three building blocks of self-adapting SoC for ultra-low power consumption

Fig. 5 shows an overview of the building blocks for the proposed sub-area. It comprises the swap power manager, the swap scheduler and the swapping unit. The swap power manager receives monitoring data on the system's status, like remaining battery energy, changes in constraints imposed to the systems (the system might need to run temporarily at higher performance etc.) and so on. According to the swap power manager's policy, a swapping is initiated and the swap scheduler is instructed to provide the lowest possible power schedule under the momentarily valid performance constraints. If a schedule is not possible the swap power manager is informed in order provide a different swap scheme etc. Once, a valid schedule is found, the swapping execution unit is eventually conducting all necessary actions to perform the swapping. At any time, new monitoring data may trigger a re-swap.

3.3 Monitoring

An organic architecture requires a comprehensive, flexible, and adaptive monitoring approach for self-organization. Current monitoring systems, however, do not meet these features due to their limited monitoring capabilities and single source monitoring. In this case, we propose to develop a flexible and unifying framework for monitoring of virtually any system component. To acquire parameters for middleware to perform the loop controls, multilevel monitors are needed. These monitors are distributed across the system, connected to different components and layers, responsible for a variety of performance metrics, and therefore must be themselves reconfigurable and adaptable, multifunctional and efficient. For this, we develop a flexible and unifying framework for monitoring of virtually any system component. The central parts of this framework are a generic and standardized Monitor Cell capable of holding an adaptable, reconfigurable monitoring device, domain-specific Monitor Analysis Modules for extracting, processing, and storing system states, and a uniform software infrastructure for correlation of monitoring information. Depending on the target

object, a Monitor Cell can be in the form of a hardware facility or a software component, where the former deploys the same structure as the processing cells and the corresponding Analysis Modules will be implemented using Hardware Description Language, while the later relies on dynamically downloadable kernel modules to hold the Analysis Modules which are also implemented using software.

3.4 Organic Middleware

To interconnect the processing cells and to form the virtual organs envisaged in our project, an organic middleware architecture is needed. This middleware architecture should be strongly influenced by successful biologic strategies and mechanisms.

We envision using decentralized closed control loops to operate the system and to control the virtual organs. Like in biology, these control loops must be decentralized to prevent a system failure in case of the failure of a control loop component. Fig. 6 gives an overall view of the processing cells, middleware to form the virtual organs and the applications induced by the brain level.

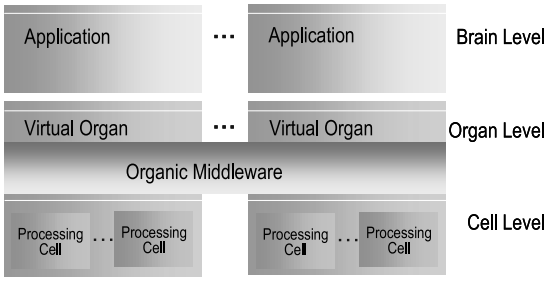

Fig. 6: Organic system architecture with middleware

To realize these decentralized control loops, the biological principle of suppressing and accelerating messengers is used. Fig. 7 shows the basic concept:

Each processing cell initially contains all or at least a big subset of all system tasks. This idea is adapted from the biological cells, which contain the full genome. It is not a contradiction to low energy consumption, because the program code for all tasks can be stored in read-only memory. Memory sections of inactive tasks can be turned off thus not consuming energy.

For each task, a processing cell sends a messenger containing a value how eager the cell is to execute this task. Let's call this the Eager Value. E.g. if a cell is not able to execute a specific task at all (e.g. due to missing periphery or computation power), it sends a 0. If a cell is quite able to perform the task, it sends a medium value. If the cell is eager to execute the task (e.g. due to special hardware very suitable to perform the task), it sends a high value.

Each cell compares for each task its Eager Value to the Eager Value of the other cells. If its Eager Value is higher, the cell starts executing the task.

In the moment a cell starts executing a task, it sends a suppressor messenger to all the other cells. This suppressor is subtracted from the Eager Value thus preventing others from executing the same task.

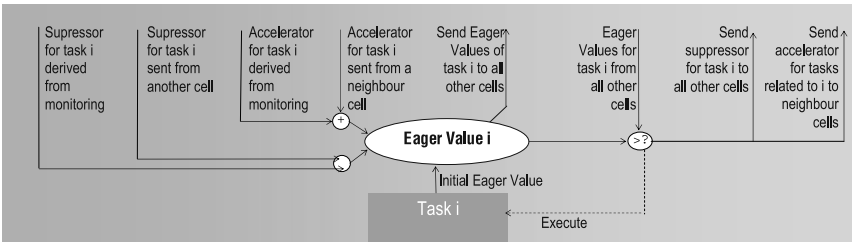

Fig. 7: Principles of a decentralized biologically motivated control loop using messengers

For the virtual organ idea, it is advantageous when cooperating tasks are executed on neighbor cells to reduce the communication overhead in the virtual organ. This can be realized by accelerator messengers. If a cell has started executing a task, it not only sends suppressors for the same task to all other cells, but as well accelerators for the related tasks to the neighbor cells. These accelerators are added to the Eager Value thus increasing the probability for a related task to be executed on a neighbor cell.

To realize closed control loop, vital system parameters like performance, network load, energy consumption, task response times etc. are monitored. This monitoring allows conclusions on how well a certain task performs on a certain processing cell. Based on the monitoring results accelerators or suppressors for this task are calculated and added or subtracted from the Eager Value thus encouraging or discouraging the continuation of task execution on this cell. If due to the accelerators and suppressors the Eager Value of task i currently executed on cell m becomes lower than the Eager Value of the same task on cell m, this cell will take over task execution.

These principles enable main organic computing features like self-organization. self-optimization and self-healing.

3.5 Organic Robot Control

Today's fast changing industrial markets force manufacturers to reduce costs and production time and to increase productivity whilst guaranteeing quality standards. Manufacturing systems are becoming more and more automated. However, due to application specific control technologies and fixed configured machines and robots, limited flexibility of manufacturing systems is given. In order to handle small and fast changing production series, future manufacturing systems must be build of highly flexible and reconfigurable components.

In DodOrg a more flexible way of robot based manufacturing is examined. Two levels of production are investigated. At the first level there are elementary autonomous production units, i.e. an autonomous robot which has knowledge over itself and which can offer its capabilities to a higher manufacturing level, i.e. a production cell. At the second level there is an autonomous production cell which consists of elementary autonomous production units, i.e. robots, transport and storage units. The production planning system plans an order on a high level. According to this order the

autonomous production units are able to configure themselves in a decentralized way over a local area network to an autonomous production cell. Each robot offers his capabilities over the network to the cell control which is able to self-organize the production cell consisting of different robots according to the order.

3.5.1 Self-organizing Robot System

There are two levels of self-organization and self-configuration in this manufacturing scenario. At the first level there is the autonomous robot which consists of an automation system. The automation system consists of different kind of field networks for coupling the cell level, the drives, the IO's and the sensors. According to the order, the tasks, the role and occurring failures of the autonomous robot an autonomous self-configuration and self-reconfiguration of its software and hardware system is performed. Monolithically structured robot controls can only be adapted and enhanced with high efforts. Therefore a component based software architecture for the organic robot control is developed. Self-configuration of the robot control for a given manufacturing task is done based on knowledge and on rules. The feasibility of the approach is shown in a simulation environment. The transfer to a real robot system is aspired in a later phase of the project.

3.5.2 Self-organizing Robot Swarm

At the second level of self-organization according to a given order a free robot offers its capability to a cell control which is able to distribute individual tasks autonomously to the robots according to their offered capabilities and according to the required capabilities of the order (Fig. 8).

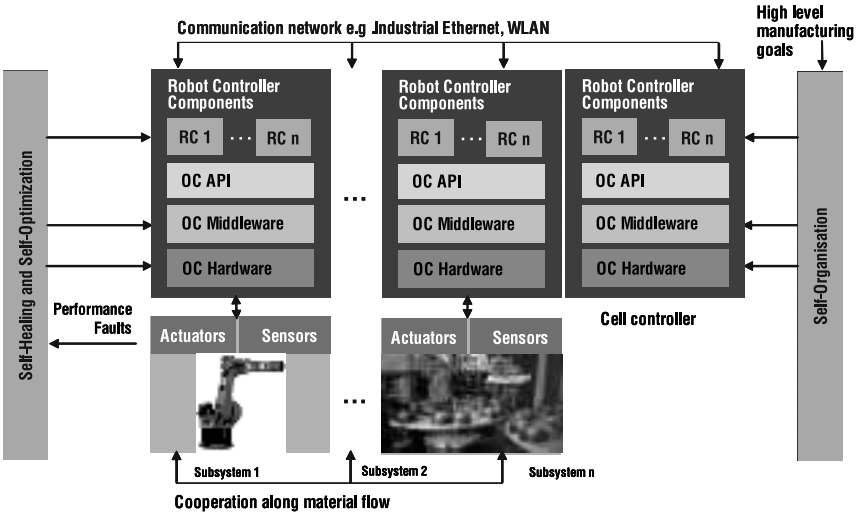

Fig. 8: Self-organization of an autonomous robot-based production cell

Thereby a main task is to configure and to reconfigure autonomously the hardware and the software of the production cell (different robots, transport systems, storages, fixtures, tools and local hardware and software of the automation system of the elementary production units).

To realize robots that cooperate intelligent with one another, the exploration of decentralized control strategies is done. Safe interaction including dynamic path planning and collision avoidance are also addressed, because the robots work together in a shared workspace. Mechanisms to resolve competitive situations and to avoid stagnations are examined as well. In order to cope with unexpected events and failures in dynamic environments, robots are given more autonomy. Key components, such as programs and data, are interchangeable among different robots. Furthermore it is possible that if one robot fails a task, another one can take it over. Thus, the robot group introduces some kind of redundancy into the manufacturing process. Through communication various information, e. g. positions, current status, future actions, etc. are exchanged. Robots tell each other what they are doing. Thereby the knowledge on how to react when a dynamic change occurs is distributed over the cooperating robots. Communication is done both between robots and robot to human operator. In these highly mobile environments beside PDA's and other hand-held devices also direct human-robot interaction over tactile or vision sensors are investigated.

4 Conclusion

In this paper an organic computing system that is especially suited for real-time applications is proposed. Starting with investigating basic biological mechanisms, we eventually derive a digital, on-demand computing organism representing the three levels, "brain", "organ" and "cell". The "on-demand" characteristic thereby emphasizes its responsiveness to environmental requests / changes as well as to changes resulting from the dynamics of the computing organism itself.

Beginning with the brain level, a software architecture for a robot controller with emphasis on self-x features is proposed. It closely interacts with an organic middleware at the organ level, featuring a decentralized control loop using messengers. At the cell level, a novel adaptive and dynamically reconfigurable hardware architecture is capable to implement the self-x features in an efficient way. In between, a power management system's architecture co-ordinates brain level and cell level for ultralow power system efficiency. All levels are supplied with monitoring techniques and architectures as a prerequisite for enabling self-x features. We believe that our comprehensive approach to organic computing will represent the first step towards more adaptive, more power efficient and more flexible future embedded real-time systems.

Acknowledgments. The work presented in this paper is supported by a grant from the German Research Foundation (DFG) within the scope of the priority programme ''Organic Computing'', SPP 1183.

References

[BH03] Becker, J.; Hartenstein, R.W.: Configware and morphware going mainstream, J. of Systems Architecture 49, pp. 127-142, 2003.

[BBU05] Becker, J.; Brändle, K.; Ullmann; M.: Rekonfigurierbare Hardware und intelligente Laufzeitsysteme für adaptives Rechnen. "it-information technology journal", Oldenburg, 2005.

[D00] DeHon, A.: The Density Advantage of Configurable Computing; IEEE Computer, 2000.

[BHP05] Becker, J.; Hübner, M.; Paulsson, K.; Thomas, A.: Dynamic Reconfiguration On-Demand: Real-time Adaptivity in Next Generation Microelectronics, ReCoSoc2005, Montpellier, France, 2005.

[BB00] Benini, L.; Bogliolo, A.; De Micheli, A.G.: A survey of design techniques for system level dynamic power management; Very Large Scale Integration (VLSI) Systems, IEEE Tran actions on, Volume: 8, Issue: 3, June 2000, Pages: 299 – 316, 2000.

[BM97] Benini, L.; De Micheli, G.: Dynamic Power management: Design Techniques and CAD Tools, Kluwer, 1997.

[P01] Pedram, M.: Power management and optimization in embedded systems systems, Proc of Asia and South Pacific Design Automation Conference, pp.239-244, 2001.

[BBP99] Benini, L.; Bogliolo, A.; Paleologo, A.; De Micheli, G.: Policy optimization for dynamic power management; Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Volume: 18, Issue: 6, Pages: 813 – 833, 1999.

[H89] Hernandez-Lerma, O.: Adaptive Markov Control Process, Springer Verlag 1989.

[CR03] Chiasserini, C.F; Rao, R.R.: Improving energy saving in wireless systems by using dynamic power management; Wireless Communications, IEEE Transactions on, Volume: 2, Issue: 5, Sept. 2003, Pages: 1090 – 1100, 2003.

[W94] Welbon, E. et al.: The POWER2 Performance Monitor. IBM Journal of Research and Development, vol. 38, no. 5, 1994.

[INT02] Intel Corporation: Intel Itanium Architecture Software Developer's Manual, Volume 1-3. 2002.

[KC99] Karlin, S.C.; Clark, D.W.; Martonosi, M.: SurfBoard - A Hardware Performance Monitor for SHRIMP. Technical report of Princeton University, Computer Science Department, 1999.

[L93] Lenoski, D. et al.: The DASH Prototype: Logic Overhead and Performance. IEEE Transactions on Parallel and Distributed Systems, vol. 4, no.1, 1993.

[J89] Jetschke, G.: Mathematik der Selbstorganisation, Harry Deutsch Verlag, Frankfurt, 1989.

[KC03] Kephart, J.O., Chess, D.M.:. The Vision of Autonomic Computing, IEEE Computer, 2003.

[VDE03] VDE/ITG (Hrsg.), „VDE/ITG/GI-Positionspapier Organic Computing: Computer und Systemarchitektur im Jahr 2010“, GI, ITG, VDE, 2003.

[OM03] Object Management Group. RealTime - CORBA Specification (Dynamic Scheduling) 2.0, OMG Document formal/03-11-01 edition, 2003.

[BFL03] Buschmann, C.; Fischer, S.; Luttenberger, N; Reuter, F.: Middleware for Swarm-like Collections of Devices, IEEE Pervasive Computing Magazine, Vol. 2, No. 4, 2003.

[TBP03] Trumler, W.; Bagci, F.; Petzold, J.; Ungerer, T.: Smart Doorplate - Toward an Autonomic Computing System. The Fifth Annual Internatioanl Worshop on Active Middleware Services (AMS2003), Seattle USA, 2003.