Development and Evaluation of a Self-Adaptive Organic Middleware for Highly Dependable System-on-Chips

Benjamin Betting, Mathias Pacher, and Uwe Brinkschulte
Institute for Computer Science
Johann Wolfgang Goethe-University
Frankfurt am Main, Germany
Email: {betting, pacher, brinks}@es.cs.uni-frankfurt.de

***Abstract*—This article presents our concept of an artificial hormone system for realizing a completely decentralized self-organizing task allocation using self-X properties. Besides the basics of the prior hormone concept and possible realizations in soft- as well as hardware, we present latest results of our research: evaluation of a completely AHS-controlled SoC implementing the different approaches, verifying the work and stability criteria, analysis of upper timing boundaries and showing the improvement of the reliability of the system.**

Keywords-Reliability; Artificial Hormone System; Heterogeneous System-on-Chips.

I. INTRODUCTION

Because the performance of nowadays computational systems is still increasing rapidly within each generation, the complexity to handle and manage such systems has grown in a similar way, too. Today's systems offer a high bandwidth of functionality, considering an integration of large numbers of distributed heterogeneous processing resources, showing a highly dynamic behavior in time. Although the architectural design of distributed systems differs strongly, a common layer is still provided by Middleware, managing the coordination of tasks on the corresponding resources and also hiding the distribution from the application. To be precise, Middleware is responsible for seamless task interaction on distributed hardware. All tasks are controlled by the Middleware layer and are able to operate beyond processing element boundaries as if they would reside on a single hardware platform. Besides complexity, other system criteria like reliability have become important, too. In consequence of the increasing integration density of today's SoCs, systems got likely open to system failure even during the early stages of operation. Crashing of resources can be caused, e.g., by radiation, aging or temperature hot spots. Hence, in order to handle the complex task management as well as the reliability problems of today's and even more future distributed systems, self- organizing and adapting techniques are necessary. As the term 'self' denotes, these techniques must be achieved autonomously by the system itself without any further external intervention (introduced in [7][8]). In fact, a system should be able to find a suitable initial configuration of task assignment by itself, to adapt or optimize itself to changing environmental and internal conditions, to heal itself in case of system failures or to protect itself against attacks. In this paper, we present the solution of an organic Middleware - implemented by an Artificial Hormone System (AHS) - providing self-configuration, self-optimization and self-healing for an autonomous decentralized task assignment. Furthermore, the proactive task behavior to prevent the system from arising failure conditions are implemented. In Section II, we introduce the basic principles of the AHS. Sections III and IV show theoretical constraints and implementations of the approach, which are evaluated in Section V. Topics of related work are presented in Section VI. Finally, we conclude the paper with Section VII.

II. THE ARTIFICIAL HORMONE SYSTEM

According to organic endocrine systems, the AHS considers elemental exchange of different hormone types for special communication and controlling interaction. In fact, it is the main function of the AHS to assign tasks to resources without any further external intervention. The proper assignment is handled in a self-organizing way, implemented via simple resource competition upon tasks using three major types of hormones:

- ***Eager value*** This hormone type represents the suitability of a resource to execute a task. The higher the hormonal value, the better the ability of the resource to execute the task.
- ***Suppressor*** This hormone type lowers the suitability of a task execution on a resource. Suppressors are subtracted from eager values.
- ***Accelerator*** This hormone type favors the execution of a task on a resource. Accelerators are added to eager values.

These basic hormone types are divided in further subtypes. Detailed information about these subtypes is presented when needed because they are used for fine tuning of the AHS and do not affect its basic understanding.

Keeping consistent formalism we introduce special notation, which distinguishes between received hormones, hormones to be sent and also between tasks and processing resources (like, e.g., processing cores). Therefore, tasks and

 ISBN: 978-1-61208-187-8

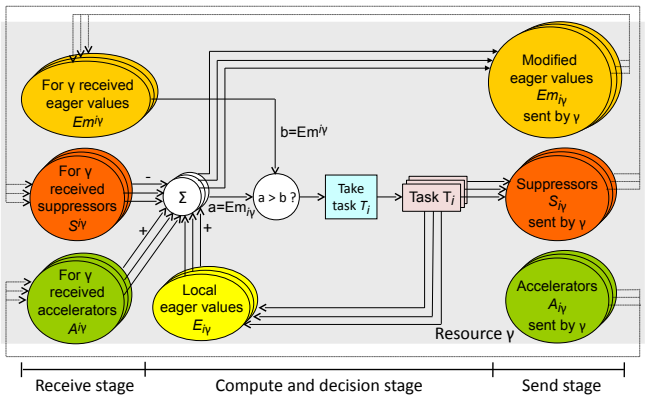

Figure 1. Hormone based control loop

resources are referenced by different indices using Latin letters such as i for tasks and Greek letters such as γ for resources. The notation of raised $H^{i\gamma}$ indents a hormone, which will be received by resource γ, dedicated and effecting task T_i. In turn subscript representation of $H_{i\gamma}$ declares resource γ and task T_i as emitter of the hormone, which is send to other resources. Each resource periodically executes the hormone based control loop presented in Figure 1. Each iteration consists of three stages.

- *Receive stage*: Resource γ receives the modified eager values $Em^{i\gamma}$, suppressors $S^{i\gamma}$ and accelerators $A^{i\gamma}$ for each task T_i from each resource inside the network. The communication between the different resources is depicted by the dashed lines.
- *Compute and decision stage*: Resource γ computes the modified eager values $Em_{i\gamma}$ for all of its tasks by the following rule. The local static eager value $E_{i\gamma}$ indicates how suited γ is to execute task T_i. From this value, all suppressors $S^{i\gamma}$ received by task T_i are subtracted, and all accelerators received by task T_i are added:

$$Em_{i\gamma} = E_{i\gamma} - \sum S^{i\gamma} + \sum A^{i\gamma} \qquad (1)$$

  The modified eager value $Em_{i\gamma}$ of each task T_i is finally broadcasted to the same task T_i on the other resources in the send stage. In each iteration a single task T_i is selected and the resource decides on allocation. For this purpose, it compares its own modified eager value $Em_{i\gamma}$ with the received modified eager values $Em^{i\gamma}$ (from all other resources) for this task. If $Em_{i\gamma} > Em^{i\gamma}$ is true for all received modified eager values $Em^{i\gamma}$, it decides to take the task. In case of equality, a second criterion, e.g., the unique identifier of the resources, is used to get an unambiguous decision. Otherwise another resource has the highest modified eager value for T_i and γ decides to not take it.
  In the next iteration step the resource selects another task and decides whether it will be taken. A resource selects the tasks in a cyclic way, i.e., each task will be selected in each m^{th} iteration, if m tasks have to be assigned. By selecting only one task at each iteration, the suppressors and accelerators can take effect. Otherwise, the decision of taking a task would happen instantaneously and the hormones would have no effect.
- *Send stage*: As already mentioned above, resource γ broadcasts the modified eager values $Em_{i\gamma}$ to each task T_i on the other resources. The strength of these values depends on the results of the computation in the last phase.
  If a task T_i is taken on resource γ, it also broadcasts suppressors $S_{i\gamma}$ dedicated to the same task on all other resources. On one hand sending the suppressors indicates the resource has taken the task, and on the other hand, it limits the number of further allocations of the same task somewhere else.
  Furthermore, the resource multicasts accelerators $A_{i\gamma}$ to its neighbored resources to attract tasks cooperating with task T_i to neighbored resources, thus forming clusters of tasks.

Our approach is completely decentralized, each resource is responsible for its own tasks and the communication to other resources is realized by a unified hormone concept. The AHS offers the following so called self-X properties:

- The approach is self-organizing, because no external influence controls the task allocation.
- It is self-configuring, an initial task allocation is found by exchanging hormones. The self-configuration is finished as soon as all modified eager values become zero, meaning no more tasks have to be taken.
- The self-optimization is done by re-offering tasks for allocation. The point in time for such an offer is determined by the task or by the resource.
- The approach is self-healing: In case of a task or resource failure, the emission of related hormones stops. This results in an automatic reassignment of the task to the same resource (if it is still active) or to another resource.

In addition, the self-configuration, self-optimization and self-healing is real-time capable. Tight upper time bounds are given for self-configuration, these are presented in detail in [2][4][12].

III. Implementation Constraints

Additional to the theoretical concept of Section II, the real implementation of AHS has to consider further aspects of the environmental system surrounding. Based on the primary decentralized control loop mechanism of Section II, every resource or processing element holds its own local instance of the AHS during runtime. In fact today's SoCs offer the significant opportunity of an dual realization of the AHS in

 ISBN: 978-1-61208-187-8

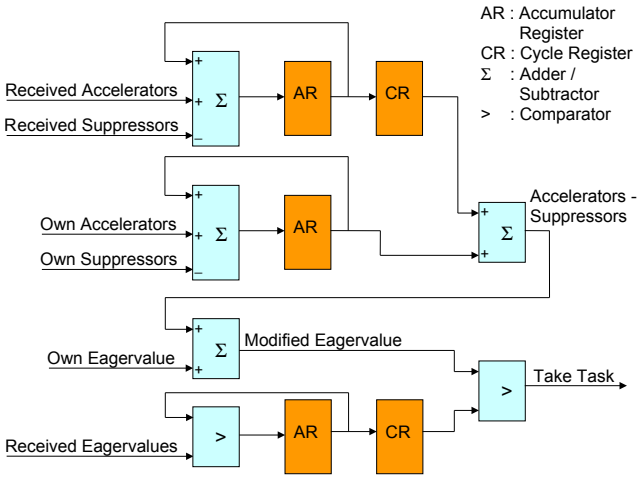

Figure 2. Simply buffered solution of the hormone cycle implementation using self-synchronization of cycles

both domains of soft- as well in hardware. Obviously this is reducible to the likely natural representation of hormone signals in terms of simple coded messages easily delivered between tasks and resources.

First a total hormone buffered solution, using a local memory backing up all received hormones for each task within each cycle, is traced. This approach keeps the advantage of no further required modifications of the origin hormone cycle and its steps, retaining a still asynchronous behavior during run time. Due to the hormone memory, jitter effects can be compensated. Duplicate hormones can be identified and eliminated while missing hormones can be bridged by previous values. But the approach also considers a high effort of required management, controlling those hormone memories, for, e.g., keeping them consistent. In cause of this, the real implementation uses a simplified second approach discarding the complex hormone memory and exploits the accumulation character of the hormones as described in Section II (adding accelerators and subtracting suppressors from eager values). As shown in Figure 2, hormone values of the current cycle are just accumulated and the overall sum of the previous cycle is stored in a register. Instead of storing many hormone values for each task within each cycle, just the last recent accumulations for accelerators and suppressors as well as the highest modified eager value are kept. Considering this, the approach takes a high benefit and requires a less level of buffering complexity than the first approach. But due to the asynchronous processing resources the risk of evaluating inconsistent hormones remains.

Therefore, this approach considers a modification of the origin cycle scheme tracing an active self-synchronization of the hormone cycles. This feature is simply achieved by using the received hormones for synchronization. To be precise, each cycle holds a waiting period right after startup,

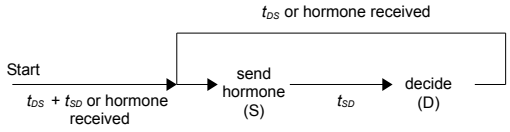

Figure 3. Self-synchronized approach of the hormone cycle using waiting period after startup

before emitting hormones to other resources. This waiting period ends after the total hormone cycle time or the receipt of a hormone from another resource, whatever happens first. So, one resource will be the first and the others will follow when receiving the hormones from the first resource. The same happens after each loop period. This keeps all resources synchronized within the maximum communication time t_k between the resources. Figure 3 shows a detailed timing schedule of the hormone cycle, where t_{DS} denotes the maximum time span between the decision and sending stage and vice versa t_{SD} between the stage of sending and decision. Hence the maximum total cycle time is set up by accumulation of t_{SD} + t_{DS}. To specify both spans as constraints, the maximum time displacement between the earliest cycle evaluating resource P_γ and the latest one P_δ, which is the already mentioned maximum communication time t_k between P_γ and P_δ, must be considered. In order to satisfy the receipt of a hormone sent by the later P_δ to the earlier P_γ, t_{SD} must be at least $2 * t_k$ (t_k needed for the communication + t_k as maximum time displacement between P_δ and P_γ). The definition of t_{DS} is rather simple. To guarantee a synchronized restart of both the early P_γ and the late P_δ, t_{DS} has to be at least t_k. Due to tolerances caused by timers on local resources, an additional jitter compensation factor $\triangle t_{SD}$ has to be considered. So t_{DS} has to be at least $t_K + t_{SD}$, which guarantees a synchronous start of the next hormone cycle on every resource. Finally the total time of the self-synchronized hormone cycle is set up by the accumulation of both constraints to at least $3 * t_K + \triangle t_{SD}$.

In summary, the major advantage of the self-synchronized approach is the feature of hormone consistency. Within each cycle, resources are capable to take correct decisions right after the receipt of all necessary hormones sent by other resources. Furthermore this modification does not influence the already taken assumptions about other self-X properties like healing or configuration, unless the basic processing procedure of the hormone cycle will not be modified. As a disadvantage, it causes a hormone cycle time increase by $t_K + \triangle t_{SD}$ compared to the asynchronous hormone buffered solution, which requires only $2 * t_k$ as cycle time (see [3]).

 ISBN: 978-1-61208-187-8

IV. IMPLEMENTATION

Because the AHS is intended to control a decentralized assignment of tasks in software- as well as in hardware related domain, specific implementations regarding different system environments are required. In fact, two implementations of the AHS have been developed, providing a specific realization of the prior hormone loop for the processing of software and hardware tasks.

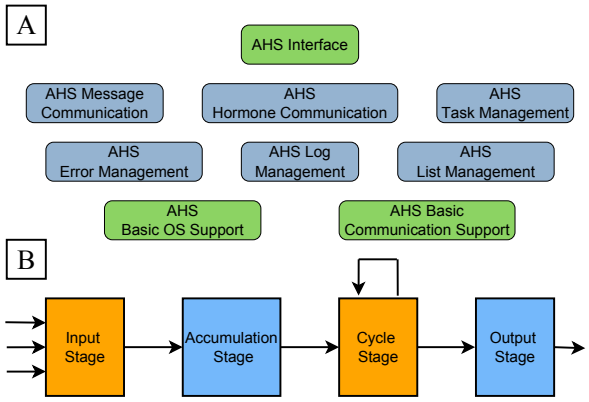

Figure 4. Structure of software (A) and hardware (B) implemented AHS, coded in ANSI-C and VHDL

A. AHS on software-level

In order of compatibility reasons and platform independence, the AHS features a pure coded ANSI-C implementation. Considering this, the implementation possesses a high level of reusability and achieves the goal of platform independent Middleware. This fact also leads to a usability in the domain of low-performance microcontroller computing. The implementation of AHS is set up by different modules and interfaces related to the AHS kernel as well as the attached Operating System (OS) and distributed application. The implementation structure is shown in Figure 4A. Starting on the lower operating system level, elementary modules for task handling and I/O communication are supported. In case of platform porting, these modules must be implemented. Regarding this abstraction, the approach takes high benefit and requires less effort in modification and maintenance. Upon OS modules, the main kernel of the AHS takes place, isolated of the attached platform depended components.

B. AHS on hardware-level

Porting the basic hormone concept to real hardware implementation takes the approach one step further. Against pure software implementation, the entire hormone loop mechanism is spread up into 4 different pipeline stages (Figure 4B). Each stage represents a single isolated hardware component, which implements a specific step of the proper hormone loop shown in Section II. Further, this pipeline has currently been fully implemented on register transfer level by hardware description using Very High Speed Integrated Circuit Hardware Description Language (VHDL). Within each single hormone cycle all stages of the pipeline are passed, issuing the decision whether task T_i is taken by the corresponding resource or not. Because the current hardware implementation realizes the self-synchronized variant of AHS only, an cycle stage is attached for buffering already accumulated hormone values, avoiding the use of heavy weighted local memories. This stage also confirms the criteria of taking correct decisions upon consistent hormone data by delaying progress of the current evaluation cycle until all necessary hormone data of all other resources is received. To avoid everlasting stall of the pipeline due to missing data, this stage is passed lately after the timing constraint of $T_{SD} \geq 2 * t_k$ (shown in Section III) is expired.

V. EVALUATION

As next step, an evaluation to show the increase of dependability using the hormone system for task assignment in a distributed system is conducted. To achieve full insights in hormone processing, a hormone cycle accurate hormone simulator for the AHS has been developed [13]. Besides the capability of simulating a dynamic processing grid, containing multiple mixed signal resources, the provision of self-X properties is ensured. We used the grid of 16 heterogeneous resources with 16 different tasks to be executed. This simulation focuses on self-healing of dynamic failures during runtime. As a reference, we first run a simulation with deactivated self-healing. This means, after the self-configuration process the hormone cycle was deactivated. Failures caused by single event upsets, aging and temperature effects have been created by a stochastic process according to corresponding failure models described. In the first simulation, transient and permanent failures leading to task or core crashes are considered. Figure 5 shows the result. Initially, all 16 tasks are allocated by self-configuration. This process is finished at hormone cycle 6, so at that time the system is operational. Already at hormone cycle 7 the first failure, a single event upset, crashed one task. More task crashes due to single event upsets followed at hormone cycles 25, 37, 47 and 51 further reducing the number of active tasks. No aging or temperature based failures occurred up to that point in time. So, starting from hormone cycle 7 the system is no longer operational as can be seen by the linearly increasing system downtime (violet line) resulting in a downtime ratio of $50/51 = 0.98 = 98\%$ at hormone cycle 51. Figure 6 shows the same scenario with self-healing activated by the hormone cycle. To be comparable, the stochastic process creating the failures was initialized with the same random seed to produce identical events. Again, the system comes operational by allocating all 16 tasks at hormone cycle 6 while at hormone cycle 7 the first single event upset occurred crashing a task. This caused the corresponding task suppressor to vanish. Due to

 ISBN: 978-1-61208-187-8

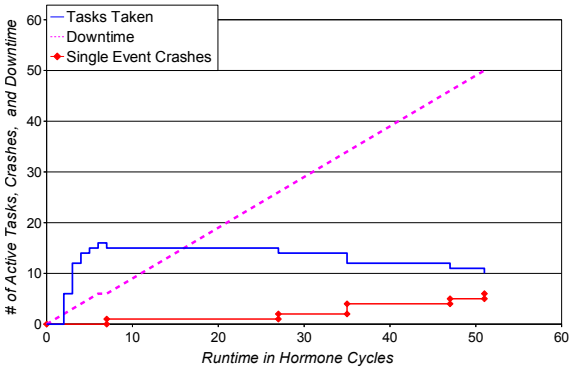

Figure 5. Behavior of 16 AHS resources with deactivated self-healing

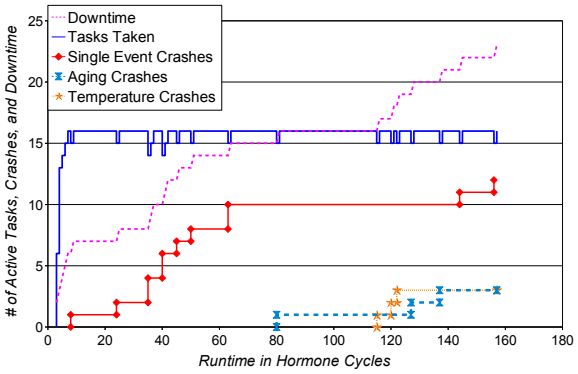

Figure 6. Behavior of 16 AHS resources with reactive self-healing

the resulting hormone imbalance, this task is reallocated at cycle 8 bringing the system back online. The same happens for the following failures. Every time a task is crashed by a failure, the hormone system compensates this event by task reallocation or reassignment*. Beginning at hormone cycle 81, aging and temperature based crashes occur as well and are compensated. Even so the system downtime still increases due to these crashes, it increase much slower and the system always comes back online, as long as enough cores are available to take all tasks (either by other cores or regeneration of the crashed cores). The downtime ratio is $23/158 = 0.14 = 14\%$ at hormone cycle 158.

The behavior shown is a pure reactive self-healing process. To allow proactive failure handling, additional suppressors can be applied. By monitoring, e.g., failures and temperature, suppressors can be emitted for resources with high temperature or failure count. This favors reliable and cool resources in comparison to unreliable and hot ones. The

*In case of a permanent failure, the task is reallocated to another core. In case of a transient failure, the task might be reallocated to another core or reasigned to the same core. This depends on the hormone balance induced by the current task distribution.

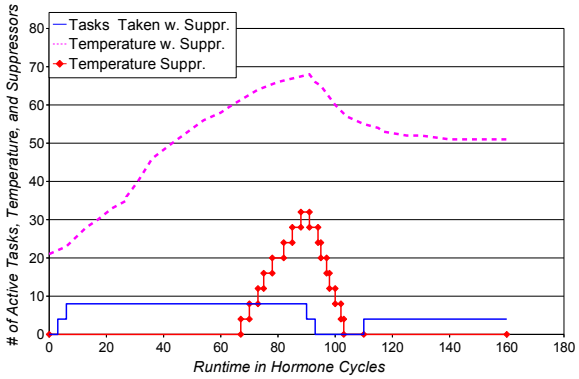

Figure 7. Temperature proactive task behavior of a local AHS resource

major effect of this proactive reallocation behavior is shown in Figure 7, where temperature suppressors are proportional emitted to the raising temperature load. This successively reduces the suitability (eagervalue) of the core until tasks get migrated to other cores. As a result of the sinking workload, the temperature and the temperature suppressor are declining. The temperature and load are balancing at a reasonable level. The proactive task assignment increases the reliability by preventing cores from total failing, using a rebalance of the workload via task distribution on different cores. In conclusion, the evaluation shows the major advantage of the hormone cycle for task assignment. Comparing with the results of the first simulation the system achieves an excellent enhancement in downtime optimization thus improving dependability in a significant way.

VI. Related Work

Currently, there exist only a few approaches of task assignment in Middleware on future CMP based mixed-signal SoCs. The approach of [11] traces the assumption of a reliable multi-layered MPSoC architecture against thermal issues due to increasing task processing. This concept internalizes a proactive task migration on cooler resources by an distributed hierarchical agent-based system. Like our approach the system is widely capable handling single point of failure within the exception of high-level agent errors, which is resisted by hardware redundancy. Furthermore, the system is restricted to thermal management domain on none intermixed circuit technology only.

Another approach using also the assumption of an agent distributed system is shown in [1]. The author presents an algorithmic schedule for task distribution on a processing grid. Against our solution, this approach uses centralized elements, so called Group Manager's (GM), responsible for the internal controlling in a clustered bunch of tasks. Unless a single GM-instance fails, there is no possibility for restoring the corresponding group information, which implies a single point of failure occurrence.

 ISBN: 978-1-61208-187-8

In [9], two algorithms for task scheduling on heterogeneous systems are presented. Within both schedulings task priorities are computed statically or dynamically. The first algorithm, Fast Critical Path (FCP) uses dynamic increase of priorities to ensure time constraints are kept. The second approach, Fast Load Balancing (FLB) uses a workload balanced task assignment to ensure every processor core will be used. In contrast to our approach, both algorithms do not regard crashing of cores and tasks.

Heiss and Schmitz [6] presented another approach for a decentralized load balanced task assignment. The authors consider a physical model where tasks are represented as particles, which are influenced by forces like, e.g., load balancing force (issued by the load potential of cores) or communication force (intensities between tasks). Depending on the resultant force action tasks are assigned to corresponding cores.

Other approaches for workload balanced assignment are presented in [5][10]. In summary, none of the concepts above covers a decentralized assignment of tasks including the spectrum of self-X properties and real-time conditions like our approach.

VII. Conclusion and Future Work

In summary, this paper presented an approach of a self-adaptive organic Middleware solution for highly dependable SoCs. The organic Middleware is represented by the AHS - an artificial hormone system providing a decentralized self-organized assignment of tasks on processing resources.

In prospection and future work, the whole project investigates further analysis of the reliability especially in a field of real SoC computing, facing timing behavior on real prototypes leaving the accomplished sector of simulation behind. Therefore, we intend the development of a highly dependable mixed signal SoC. The prior AHS is used in combination with a generalized core and task concept to assign software and hardware tasks to suitable mixed signal resources, so called processing elements (PEs). Every PE represents a specific SoC function, which can be any type of processor core like timer, memory, analog or digital PE. Since different hardware tasks and analog PEs are involved, the hormone controlled concept is extended for time continuous processing of analog hormone signals. The interconnection and communication throughout both systems is realized by a common inter-core network with redundant interfaces. Overall the interaction of the resultant reliable mixed signal SoC has to be achieved facing a real demonstrator application settled in the predestined area of automotive driven assistance control. For this, the SoC is admitted controlling a complex model helicopter.

References

[1] L. F. Bittencourt, E. R. M. Madeira, F. R. L. Cicerre, and L. E. Buzato. A path clustering heuristic for scheduling task graphs onto a grid. In *3rd International Workshop on Middleware for Grid Computing (MGC05)*, Grenoble, France, 2005.

[2] Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln. Towards an artificial hormone system for self-organizing real-time task allocation. *5th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS)*, pages 339–347, 2007.

[3] Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln. An Artificial Hormone System for Self-Organizing Real-Time Task Allocation in Organic Middleware. In *Organic Computing*. Springer, 2008.

[4] Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln. An artificial hormone system for self-organizing real-time task allocation. *Springer*, 2008.

[5] Jorge Finke, Kevin M. Passino, and Andrew Sparks. Stable task load balancing strategies for cooperative control of networked autonomous air vehicles. In *IEEE Transactions on Control Systems Technology*, volume 14, pages 789– 803, 2006.

[6] Hans-Ulrich Heiss and Michael Schmitz. Decentralized dynamic load balancing: The particles approach. In *Proc. 8th Int. Symp. on Computer and Information Sciences*, Istanbul, Turkey, November 1993.

[7] Paul (IBM Research) Horn. Autonomic computing manifesto: IBM's perspective on the state of information technology, October 2001.

[8] Christian Mueller-Schloer, Christoph von der Malsburg, and Rolf P. Wuertz. Organic computing. *Informatik Spektrum*, 27(4):332–336, 2004.

[9] Andrei Radulescu and Arjan J. C. van Gemund. Fast and effective task scheduling in heterogeneous systems. In *IEEE Computer - 9th Heterogeneous Computing Workshop*, Cancun, Mexico, 2000.

[10] Larry Rudolph, Miriam Slivkin-Allalouf, and Eli Upfal. A simple load balancing scheme for task allocation in parallel machines. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 237–245, 1991.

[11] Thomas Ebi, Holm Rauchfuss, Jörg Henkel and Andreas Herkersdorf. Agent-based Thermal Management using Real-Time I/O Communication Relocation for 3D Many-Cores. *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation (PATMOS), Madrid, Spain*, September 26-29 2006.

[12] Alexander von Renteln and Uwe Brinkschulte. Reliablity of an Artificial Hormone System with Self-X Properties. In *Parallel and Distributed Computing and Systems*, Cambridge, Massachusetts, USA, November 19 - 21 2007.

[13] Alexander von Renteln, Michael Weiss, and Uwe Brinkschulte. Examining Task Distribution by an Artificial Hormone System Based Middleware. In *11th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2008)*, Orlando, Florida, USA, May, 5-7 2008.

 ISBN: 978-1-61208-187-8