Adaptive System Architectures

Klaus Waldschmidt
J. W. Goethe-University, Frankfurt

January 23, 2004

Abstract

Embedded systems, ubiquitous computing, and networked architectures are research areas in computer science where the features of organic computing become increasingly essential. Such features, like self organization and self configuration need to be combined with the still increasing requirement for computing performance. Adaptive computing systems are a promising completion to classical computer architectures. Adaptive Computing Systems (ACS) offers the opportunity to adapt the whole architecture or parts of the architecture to the changing needs of applications or changing environments. Reconfigurable Logic (RL) can itself contain Mono- or Multiprocessors or it can be a component in such systems or computer clusters. All levels of parallelism can be combined with all levels of reconfigurability. Configuration and concurrency offer a huge design space to be explored. They become tightly correlated issues in modern adaptive computer architectures. Different grains of configurability bring the flexibility into architectures. This flexibility is necessary to achieve a better exploitation of parallelism in algorithms. Architectures can thus be adapted to all the needs of problems or algorithms to turn the inherent or explicit parallelism into efficiency.

The paper addresses some of these aspects and presents some ideas for modelling and classifying adaptive computing systems (ACS) on different levels of granularity.

1 Introduction

Modern computer systems will increasingly consist of autonomous and cooperative system parts with a self-organizing behaviour. Self-organization is mainly conducted by adaptive and context-oriented behaviour. Therefore such systems are called organic computer architectures [1].

The following features of organic computing will have to be integrated into all types of adaptive systems.

- self configuration: The system should adapt itself to different environments. Sign-in and sign-off at runtime should be available for all machines in the cluster
- adaptivity: The system should cope with heterogeneous computers, so computers with different speeds and even different operating systems can participate.
- self distribution: Intelligent distributed scheduling and migration schemes should offer that data and instruction code are automatically distributed among the participating computing devices at a minimum cost of machine time.
- self organization: The system should automatically reorganize and rebalance itself if an imbalance, e.g. due to preceding growing and shrinking, leads to a costly overhead in runtime or communication latencies.
- self healing: Crashes should be detected, handled and corrected by the system automatically.
- automatic parallelization: Hidden parallelism in programs should be detected and exploited.
- self protection and protection of others: For a cluster of computing devices working within an open network it is important to consider safety aspects. The authenticity of users and data has to be ensured, and sabotage, spying and corruption has to be prevented.
- accounting: Mechanisms and cost functions for accounting of provided and used computing time should be offered.

Besides monolithic processors and dedicated parallel computers with a fixed topology large clusters of processing nodes as parallel systems become increasingly important. These clusters show a huge dynamic and heterogeneity with a structure that is usually unknown at the compile time of the application. Furthermore, the CPUs of the cluster have immense changing loads, cluster nodes are mostly specific and networks can be spontaneous with vanishing and appearing resources.

2 Classification

In adaptive computing systems concurrency and configurability thus have to be combined for synergy effects. An open question still remains: What is the best programming model and control structure for these architectures to support all or some of the above mentioned aspects? Hartenstein [1] proposes the Anti Machine [2] which basically combines the dataflow principle with the dimension of re(configuration). Other approaches provide classical von Neumann architectures or ensembles of von Neumann like structures with (re)configuration.

In any case a taxonomy would be very helpful for a discussion of these questions. A taxonomy in general defines classes of heterogeneous objects with respect to their attributes. It structures a design space and therefore supports the evaluation of new solutions on the basis of the object classes.

In computer architectures the best known and most frequently used model for classification is the taxonomy of Flynn [3]. Computer architectures are classified with respect to their parallelism in the instruction and the data streams. To cover the aspects of configuration Flynns classification scheme could be extented with the new dimension of configurability.

In contrast to Flynns original classification scheme an architecture is now defined as a triple of *(d, i, c)*, where d denotes the number of data streams, i the number of instruction streams and c the total of configuration states.

The described classification scheme is a very rough model to describe adaptive computing systems. A lot of aspects (like granularity) are lost in the model but they could easily be included by further dimensions.

3 Conclusions

In modern adaptive computing systems configuration and concurrency become tightly correlated issues. To study and to compare new architectures coming from recent research projects some extensions to Flynns classification scheme are proposed.

References

[1] VDE/ITG/GI-Arbeitsgruppe Organic Computing, Organic Computing, Computer- und Systemarchitektur im Jahr 2010. Technical report, VDE/ITG/GI, 2003

[2] R. Hartenstein, Reconfigurable Computing: urging a revision of basic CS curricula, Proceedings of the workshop on logic and synthesis for programmable devices WLSPD2002, Las Vegas, USA, 2002, August 2002.

[3] M. J. Flynn, Some computer organizations and their effectiveness, IEEE TC, Vol. C21, No. 9, Sept. 1972, pages 948 - 960.