Essay Embedded Computer Architecture
Hardware Implementations of Evolvable Hardware

IMARA SPEEK
AIMEE FEROUGE
Delft University of Technology
October 28, 2013

Abstract

Morbi tempor congue porta. Proin semper, leo vitae faucibus dictum, metus mauris lacinia lorem, ac congue leo felis eu turpis. Sed nec nunc pellentesque, gravida eros at, porttitor ipsum. Praesent consequat urna a lacus lobortis ultrices eget ac metus. In tempus hendrerit rhoncus. Mauris dignissim turpis id sollicitudin lacinia. Praesent libero tellus, fringilla nec ullamcorper at, ultrices id nulla. Phasellus placerat a tellus a malesuada.

Keywords: bio-inspired , self-aware , computing systems, machine learning

Introduction

Introduction [2] blabla [**?**]

Discussion of the Different Papers

Discussion

In artikel [4] a new kind of engineering is being introduced. Inspired by life on Earth and the natural processes of living organisms, *bio-inspired* hardware systems have evolved. The paper introduces the POE model, that classifies bio-inspired systems according to three axes:

- Phylogeny, which concerns the evolution of a species over time, aiming for optimization of the genome
- Ontogeny, which is about a second biological organization within multicellular organisms, being celluar reproduction
- Epigenesis, concerning the learning systems of organisms such as the nervous system, the immune system and the endocrine system.

Along the axis of phylogeny, evolvable hardware can be found. Artificial evolution and large-scale programmable circuits are two underlying themes. Evolutionary algorithms are common nowadays, as they strive for optimization and automatic programming. Programmable circuits are integrated circuits that are to configured by the user. Back in the days, the PAL (programmable array logic) was the most popular PLD (programmable logic devices). Nowadays, field programmable gate arrays (FPGAs) have become widely used, providing high flexibility and the possibility of being reconfigurable.

Ontogenetic systems have the ability to self-repair, as the main goal of ontogeny is growth or construction. Whereas phylogeny is about reproduction of the system by use of crossover and mutation, ontogeny is about replication by creating daughter cells that are exact duplicates of the mother cell.

Epigenetic systems are more software-based, as they contain error detection and immune systems for computers. It becomes interesting when multiple axes are being combined, as stated at the end of [4]. By dreaming about POE hardware systems that are endowed with evolutionary, reproductive, regenerative learning capabilities this article is often *geraadpleegd* in research considering evolvable systems.

The Erlangen Slot Machine from [3] tackles the four major limitations of the Virtex-II FPGA produced by Xilinx. First, the I/O dilemma caused by fixed pins spread around the device is solved by connecting all bottom pins from the FPGA to an interface controller realizing a crossbar. It connects FPGA pins to peripherals automatically based on the slot position of a placed module. This I/O rerouting principle is done without reconfiguration of the crossbar FPGA.

Second, the memory dilemma has been solved. In normal Virtex-II FPGAs, a module can only occupy the memory inside its physical slot boundary. Storing data in off-chip memories is therefore the only solution. In the ESM, six SRAM banks are connected to the FPGA. Since these banks are placed at the opposite side as the crossbar, a module will connect to peripherals from one side, while the other side will be used for temporally storing computational data. In order to use a SRAM bank (called a slot), the module must have at least a width of three micro-slots, in which the total device is divided. This organization simplifies relocation, enabling a partially reconfigurable computing system. Also, equal resources will be available for each module.

Finally, the inter-module communication dilemma is dealt with. Dynamically routing signal lines on the hardware is a very difficult task. The ESM uses a combination of bus-macros, shared memory, RMB (Reconfigurable Multiple Bus) and a crossbar to take away the limiting factor for the wide use of partial dynamic reconfiguration.

In order to initialize executable application modules and their run-time supervision the ESM requires an operating system. This is being implemented by means of a Reconfiguration Manager that uses a parallel port interface to download and store bitstreams into the flash memory. A pipelined data flow architecture has been used, replacing the finite state machine by a MicroBlaze microcontroller and employing a data crossbar between plug-ins. Providing a new architecture to avoid the current physical problems of reconfigurable FPGAs, a new inter-module communication concept, as well as an intelligent module reconfiguration management has made the ESM an alternative for the Xilinx FPGAs.

A self-managed software architecture is one in which component automatically configure their interaction in a way that is compatible with an overall architectural specification to achieve the goals of the system. Dynamic change which occurs while the system is operational, is far more demanding and requires the system to evolve dynamically and that adaption occurs at run-time. There is a community called SEAMS (Software engineering for adaptive and self-managing systems).

They focus on the use of ADLs for software design and implementation from components, including limited language support for dynamic change, a general model for dynamic change and evolution, associated analysis techniques and initial steps towards self-management.

The three-layer Gat model is presented. The bottom control layer consists of sensors, actuators and control loops and includes facilities to report the status to the middle layer. The change management layer, or middle layer reacts to state changes accordingly and can introduce new components. The upper goal layer consists of time consuming computations such as planning. They propose a component design that implements the set of services that it provides and the set that it may need.

Self-configuration is the ability of the system to perform configurations according to the pre-defined high level policies and seamlessly adapt to change caused by automatic configu-

rations. Self-optimization is the ability of the system to continuously monitor and control resources to improve performance and efficiency. Self-healing is the ability of the system to automatically detect, diagnose and repair faults. Self-protection is the ability of the system to pro-actively identify and protect itself from malicious attacks or cascading failures that are not corrected by self-healing measures.

This paper presents a wide variety of techniques in autonomic computing. Hotswapping is a technique to inject monitoring and diagnostic code into live code using interpositioning and replacement. Data clustering is an unsupervised learning algorithm, used to identify configuration classes and determines the degree of similarity between clusters using convex average metric.

In [1] a Virtex4 FPGA implementation is introduced for evolvable systems. Other then earlier versions of Virtex (e.g. the Virtex-II Pro), Virtex4 devices enable two-dimensional dynamic reconfiguration, a feature which considerably reduces the reconfiguration time an thus the evoltuion time ([1]). By using both VRCs (Virtual Reconfigurable Circuits) and direct bitstream manipulation, this new architecture eliminates the biggest limitation of Virtex FPGAs, which is an almost unknown and undocumented bitstream data format and an unsafe configuration schema.

By implementing a Cartesian Genetic Programming schema and a new cell structure (two 4-input LUTs (Lookup Table) and a multiplexer), the total speed up compared to the Virtex-II has a 16x factor. In addition, the proposed candidate solution can perform hierarchical evolution. This way, it is possible to preserve the fine-grained evolution typical of the direct bitstream manipulation systems. Also, it is possible to cope with problems requiring a high number of basic blocks.

This Virtex4-based device, which takes advantage of 2D reconfiguration capabilities and direct manipulation of the bitstreams is the first one of its kind. it enables the parallellism between the evaluation and the reconfiguration phase and by speeding-up the reconfiguration process ([1]).

Another FPGA-based architecture is proposed in [**?**]. A highly regular and modular architecture is being integrated with a widely known 2D systolic processing architecture with an optimized DPR control engine. The engine allows the implementation of adaptive (evolvable) prosessing-hardware with native reconfiguration support. In the design, the processing elements (PEs) of the reconfigurable core are structured as a 2D systolic array, known for its high performance and restrained use of resources (for collection of processed data only the lowest and rightmost PE has to be considered).

As for the reconfiguration engine, three enhancements are included in order achieve fast reconfiguration. First, only the body of the bitstream is stored. The header and tail info are eliminated and are added at run time. This leads to reduction of the bitstream size (and thus, data transference time from the external memory) and raster relocation possibilities. Second, internal memories have been included, avoiding the same element to be reallocated at different positions int he architectue. This greatly reduces the reconfiguation overhead, which is a limitation when using VRCs. Another technique that has been applied is overclocking the Virtex-4 Internal Configuration Access Port (ICAP) to 2,5 times the maximum frequency report by the manufacturer. This was no problem and also reduced the reconfiguration overhead.

Validation alone does not always guarantee trustworthiness as each individual decision could be correct, but the overall system might not be consistent or dependable. These aspects should be integrated at architectural level and not be seen as add ons. Autonomic computing is mostly based on the architecture's basic MAPE (monitor, analyze, plan, execute) control loop. Another inspiration for autonomic systems is Intelligent Machine Design (IMD), based on the human autonomic nervous system.

In large systems with a wide behavioral space it is highly complex to determine whether all autonomic decision were in overall interest of the system. There is a vital need to dynamically validate the run-time decisions of the autonomic manager. The higher goal is not to just reach self-management, but to achieve consistency and reliability of results through self-management.

Current research has looked into a fifth state of the self* properties: self-regulation. It tests itself integral in the architectural, however it assumes that the other states perform optimally and they do not ensure trustworthiness. Another believe is that trustworthiness is achieved when keeping an account of its behavior. This requires the user to intervene if necessary. A dead-zone can be introduced to prevent unnecessary inefficient and ineffective control brevity when the system is sufficiently close to its target value.

The proposed trustworthy architecture exists out of Autonomic Controller, an Validation Check, a Dependability Check and the managed sub-system. The AC doesn't matter about the content of the unit, but only provides an interface to the designers to express rules that govern the goal. The VC is a higher level mechanism that keeps track of the goal. It is important to also consider the possibility of overall inconsistency in the behavior of the system (the AM could erratically be changing its mind, causing oscillation). The DC only allows the AM to change its mind when its necessary and safe to do so. Dead-zone logic is implemented to account for this, as well as prediction and learning. A system, no matter the context of deployment, is truly trustworthy when its actions are continuously validated to satisfy set requirements.

Comparison

Comparison

Conclusion

Comparison

References

[1] M. Santambrogio F. Cancare and D. Sciuto. A direct bitstream manipulation approach for virtex4-based evolvable systems. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*.

[2] H. Hoffmann et al. F. Sironi, M. Triverio. Self-aware adaptation in fpga-based systems. In *International Conference on Field Programmable Logic and Applications (FPL)*.

[3] A. Ahmadinia M. Majer, J. Teich and C. Bobda. The erlangen slot machine: A dynamically reconfigurable fpga-based computer. *Journal of VSLI Signal Processing*.

[4] D. Mange et al. M. Sipper, E. Sanchez. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, August 2002.