Hardware Implementations of Evolvable Systems
A critical analysis on self-adaptive autonomic systems on reconfigurable architectures

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 a. lobortis ultrices eget ac metus. In tempus hendrerit rhoncus. Mauris dignissim turpis id sollicitudin lacinia. Praesent libero tellus, fringilla nec ullamcorper at, ultrices i nulla. Phasellus placerat a tellus a malesuada.

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

1 Introduction

Introduction [3] blabla [?]

2 Prior knowledge

[still need to give an introduction to all the matters spoken. what is evolvable hardware?] [firstly introduce the self* properties as they are mentioned right into the discussion]

3 Discussion of the Different Papers

[introduction to discussions and seperation of the topics, maybe seperate hardware from software solution from these papers?] [Give some small introduction to each paper in 3 sentences and then maximum of 15 for every paper]

Considerable research has already been done in order to efficiently accelerate hardware while still maintaining virtually unlimited adaptability. Software techniques in autonomic computing systems such as hot-swapping and data clustering are discussed in [1]. These self-aware systems can adapt their behavior on FPGA-based system as discussed in [4].

Inspired by life on Earth and the natural processes of living organisms, *bio-inspired* hardware systems have evolved. [6] 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 the two underlying themes. Evolutionary algorithms are common nowadays, and strive for optimization and automatic programming. Programmable circuits are integrated circuits that are to be configured by the user. Back in the days, the PAL (Programmable Array Logic) was the most popular PLD (Programmable Logic Devices). Nowadays, FPGAs (Field Programmable Gate Arrays) 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 [6]. By dreaming about POE hardware systems that are endowed with evolutionary, reproductive, regenerative learning capabilities this article is often consulted in research considering evolvable systems.

[There has to be some images added or else it doesn't come across, or just less info]

The Erlangen Slot Machine from [5] 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.

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 configurations. 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. Hot-swapping is a technique to inject monitoring and diagnostic code into live code using inter-positioning 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 [2] 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 ([2]). 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 ([2]).

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 to 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.

During the design space exploration phase of system design, overheads associated with reconfiguration and hardware/software interfaces need to evaluated carefully to harvest full potential. The context where different applications demand ever increasing adaptability and performance has already be answered by introducing reconfigurable SoC employing different multiprocessor cores. The increasing prominence of reconfigurable devices within such systems require hardware/software co-design for SoC to address the trade-off between software execution and reconfigurable hardware.

Right now this co-design emphasizes on identifying intensive kernel tasks and implementing these tasks on the reconfigurable hardware. The performance model of the hardware depends on the degree of parallelism while the performance model for software execution is static and does not become affected by external factors. There are several modes and interfaces to configure a specific FPGA family: the JTAG cable, the selectMAP interface for daisy chaining the configuration on multiple FPGAs, configuration loading from PROMs or compact ash cards, micro controller based configuration and internal configuration access port (ICAP).

Partial dynamic reconfiguration is a key feature that makes FPGAs unique. This addresses the lack or resources to implement an application and its adaptability needs. It could also be achieved by having a larger resource array, but this is not always viable for non-trivial designs. Reconfigurable hardware taking advantage of partial dynamic reconfiguration is the perfect trade-off between the speed of HW and the flexibility of SW. Other aspects that motivate the use of online self-adaptable systems are the QoS and the reliability and continuity of the service.

Important problems are the lack of availability of software tool chains that take into account partial dynamic reconfiguration, the time overhead the reconfiguration process introduces and the two-dimensional partitioning strategy reconfigurable devices need: spatial and temporal.

Scenario's where self-awareness is useful include: mobile technologies, cloud-computing systems, adaptive and dynamic compilation, multi-core micro-architectures and novel operating systems. Existing adaptive systems often fail because of they are largely ad hoc and fail to incorporate true goals.

Since the introduction of reconfigurable hardware platforms such as FPGAs, the hardware domain shifted into the software domain: the possibility of implementing a reconfigurable architecture increased the flexibility of the hardware.

In online task management the operating system requests for a hardware module by configuring the IP core on the FPGA and creating a communication channel between the module and the software application in a transparent way. This process is managed by two kernel extensions called the reconfiguration support and the centralized reconfiguration manager. This centralized approach allows for module caching or module allocation and to enhance parallel execution of hardware IP-cores.

The results of the system however shows the negative impact of reconfiguration latency in the execution of a functionality, therefor there is a big need to evaluate this overhead carefully, it is not always an improvement. This is basically the result of the paper.

Self-aware adaptive computing systems are capable or adapting their behavior and resources thousands of times based on changing environmental conditions and demands. This allows them to automatically accomplish their goals in the best way possible. The results of the papers presents a system built on top of a set of enabling technology that proves the effectiveness of using self-aware adaptive computing systems.

Because of the increasing system complexities, it is unfeasible for the average programmer to weight all the constraints and optimize systems for a wide range of machines. Self-aware adaptive computing focuses on creating a balance of resources to improve performance, utilization, reliability and programmability. A programmer will ideally only have to provide the system its goal, rather than a description of tasks, provided with some constraints.

A self-aware adaptive computing system becomes an active system where the hardware, application and the operating system has to be seen as unique entities to autonomously adapt itself. The underlying architecture has cognitive hardware mechanisms In its core to observe and affect the execution. The self-aware adaptive computing system also implements learning and decision making engines to determine appropriate actions. A key challenge is to identify what parts of a computer need to be adapted and to quantify the degree of which adaption can afford savings in metrics of interest of us.

The operating systems chooses at run-time among a set of possible implementations according to the criteria. This decision is based on the Observe-decide-act. The need for this dynamic choice between available implementations is give by the fact that the system is live and lives in an unpredictable environment.

When the throughput of the heart monitor over a certain window of time drops under a certain limit, the heuristics for hot-swapping are activated. A thing to keep in mind is that the heartbeats application contains an overhead of 3,52

The results of a first static analysis showed that hardware implementations are generally faster than software applications. However, when considering dynamic scenario's, execution times might change radically.

They propose an evolvable system that runs self-adaptive applications on top of a heterogeneous system consisting of a general purpose processor and a reconfigurable device. The operating system running on top of the heterogeneous system is responsible for providing self-adaptive capabilities. Specifications are running a customized version of GNU/Linux and a set of self-adaptive applications on top of a heterogeneous system featuring a multi-core processor, an Intel Core i7 and a reconfigurable device: a Xilinx Vertex 5 FPGA based board.

There is the increasing importance of non-functional requirements such as power consumption and reliability as well as many potentials that lie at the border of functional such as efficiency and accuracy. Meeting such constraints is getting more and more difficult , mainly because of the exponential increase of interactions among systems and the environments in which the systems are required to work. They also enable self-adaptiveness through a monitoring framework, which tackles the primitive layer of self-adaptiveness that it self-awareness and through an adapting framework which deals with self-configuration and self-optimization, the major layer of self-adaptiveness.

The monitoring process is central to self-adaptive systems. The application Heartbeats is particularly effective for implementing these applications. Using machine learning, one can enhance synchronization techniques. K32 is a modern object-oriented operating system that responds to changing and challenging environments adapting its operations. It supports on line reconfiguration of functionalities by interposition, hot-swap and dynamic update.

Evolvable systems exploiting self-adaptive techniques are self-configuring and self-optimizing systems capable of changing their operations to meet the given performance goals by modifying either the underlying heterogeneous architecture, the operating system and the self-adaptive applications.

Design choices are important when developing the observe-decide-act loop. Their vision is that the underlying hardware architecture is made up of static area and a dynamic area. The reconfigurable device can be configured to implement different functionality through dynamic reconfiguration support provided by the operating system. This is provided through standard libraries and the OS implements parts of the loop. The OS is therefor capable of choosing at run-time the best implementation for the required functionality among the available.

Heartbeat makes it possible to assert performance goals as heart-rate windows each of each is delimited by a minimum and maximum heart-rate. It updates the progress of the execution calling the function that signifies a heartbeat. It monitors the progress of the execution through either a windows heart rate and a global heart-rate.

A decisioning framework can designed using analytical models and empiric models. They choose to implement empiric models because of the generality and for the reduced effort they place on the designer. The hot-swap mechanism then provides the ability of switching among different implementations of the same functionality in a transparent function while executing. The switchable units were identified as the libraries that export an implementation of a certain functionality, a self-adaptive library or Dynamic-link library (DLL).

They created 2 libraries, one for the multicore processor and one for the FPGA. The former is programmed using C and where necessary assembly to further refine the performance.

4 Comparison

Comparison

5 Conclusion

Comparison

References

[1] M. Khan A. Khalid, M. Haye and S. Shamail. Survey of frameworks, architectures and techniques in autonomic computing. In *Fifth International Conference on Autonomic and Autonomous Systems (ICAS).*

[2] 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).*

[3] 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).*

[4] H. Hoffmann M. Maggio M.D. Santambrogio F. Sironi, M. Triverio. Self-aware adaptation in fpga-based systems. In *2010 International Conference on Field Programmable Logic and Applications.*

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

[6] 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.