Schloß Dagstuhl Seminar Report 00261
Dynamically Reconfigurable Architectures
June 25-30 2000

Gordon Brebner, Karl-Heinz Brenner, Hossam El Gindy and Hartmut Schmeck

1 Summary

The Dagstuhl Seminar on *Dynamically Reconfigurable Architectures* brought together 42 participants from 11 different countries. As with its predecessor seminar held in 1998, the participants came from three distinct communities, concerned with:

- field programmable gate arrays for fast and flexible configurable computing, and the associated design tools; and
- computational models of, and designing efficient algorithms for, processor arrays with reconfigurable bus systems;
- optoelectronic systems for communication with very high bandwidth.

A main aim of the seminar was to encourage cross-fertilisation between these communities, to work towards a new understanding of dynamically reconfigurable architectures and their possibilities. Some of the specific questions addressed by the seminar were:

- What are shared characteristics of, and possible interfaces between, machine-based computational models and circuit-based computational models?
- How will current "ASIC replacement" device architectures evolve to be genuine "computational component" device architectures?
- What are appropriate system-level architectures for systems composed of processors, configurable logic and memory, and how might these be implemented at chip level?
- What overall computational models are best to evaluate alternative system-level architectures?
- Which aspects of current hardware and software design practice are, and are not, relevant to the design of "soft" circuitry and "hard" programs?
- What are appropriate new design processes, and supporting tools, for systems composed of a mixture of circuit and program, implemented in hard and soft manners?

- How general purpose can system design be? Is specialisation inevitable to achieve best performance?
- How are reconfigurable systems best presented to algorithm designers?
- What are realistic aspirations for the practical benefits achievable from the use of reconfigurable systems?
- How can optical communication be used best to support run-time reconfiguration?

There were 34 talks given by the participants, each shedding some light on one or more of the above issues. A lively discussion session on the Wednesday evening provided an opportunity to synthesise the differing viewpoints of the participants. Interestingly, one topic that provoked much discussion was how to define the term 'dynamically reconfigurable architecture', with differing positions emerging on how dynamic the reconfiguration should be expected to be, and on what types of architecture might be included.

The seminar emphasised that technological advances have opened up new ways of implementing complex systems, ways that blur the traditional barriers between hardware and software components. Because of this, existing design tools do not seem to be adequate for the necessary new design styles - in fact, at one extreme, it is possible to let hardware evolve by itself, learning the required functions. There are many applications, ranging from computing on spacecraft to the control of motors, that should be able to derive benefits from enabling novel types of configurable computing systems.

A recurrent theme was whether the traditional Von Neumann model of computation has now run out of steam, as sequential processors become just one type of component in parallel and distributed systems. Irrespective of opinion on this, the need for robust computational models for run-time reconfigurable systems and evolvable hardware was highlighted, possibly following the general philosophy of the RMESH model for reconfigurable meshes. A thorough understanding of the computational possibilities is of great importance in assessing the benefits of dynamic reconfiguration for real-life applications.

Based on such an understanding, it will then be possible to make progress on finding apt high-level languages and notations for expressing computations based upon dynamically reconfigurable architectures. While mature concepts from both hardware and software engineering will influence this exercise, it seems that these must be augmented by natural high-level ways of expressing features such as concurrency, topology and reconfiguration.

Bridges then have to be built between the high level descriptions of function and the physical underlying architectures, which may be based on a variety of technologies, including optoelectronics. Robust computational models will also enable the development of apt virtual machines, that mask the particular technical details of specific reconfigurable computing machines.

The seminar pointed the way to how progress might be made in these directions, and new collaborations were initiated. The pleasant atmosphere of Schloß Dagstuhl was an important incentive for the lively interaction between the participants. We thank all who contributed to the success of the seminar. Furthermore, we gratefully acknowledge financial support for the seminar from Xilinx Inc.

2 Abstracts of talks

2.1 Index

2.2 Monday

Evolvable Hardware
by Xin Yao

Evolvable hardware refers to hardware that can change its architecture and behaviour dynamically and autonomously by interacting with its environment. It attempts to make hardware softer while maintaining its speed advantage. Evolvable hardware means different things to different people. Some people regard it as the application of evolutionary algorithms to hardware design. However, this view does not capture the essence of evolvable hardware. This talk gives a state-of-the-art overview of different approaches to evolvable hardware. It argues strongly that there is an urgent need to study the theoretical foundation of evolvable hardware while working on applications. Key issues, such as scalability, generalisation, circuit verification and test, must be addressed in research before any real breakthrough can be made in this area.

Reconfigurable Computers in Space:
Problems, Solutions and Future Directions
by Neil Bergmann (joint work with Anwar Dawood)

Interest in Reconfigurable Computing (RC) has recently spread to those interested in space missions. Although RC has sparked much interest in the general computing community, it has yet to demonstrate 'killer app' status for any terrestrial applications. We believe that there are compelling arguments about why RC is an excellent match to the requirements of space missions.

- After launch, unmanned spacecraft electronics are generally unavailable for physical upgrade or repair. RC technology allows new hardware circuits to be uploaded via a radio link.

- New circuit configurations can overcome design faults, allow improved processing algorithms to be uploaded, or change system functionality in response to changing mission requirements.

- The same circuitry can be used with different configurations at different stages of a mission, reducing weight and power.

- If part of an FPGA fails, then circuitry can be reprogrammed to make use of remaining functional portions of the chips.

- Use of FPGAs allows generic circuit boards to be designed, which are customised for individual applications. This helps overcome the very high NRE costs.

- In-flight reconfiguration provides additional safety margins for missions with very short lead-times, or for those where mission requirements are not fully defined at launch.

There are however problematic aspects of operating FPGAs in space:

- Ionising radiation causes soft-errors in the static RAM cells used to hold programming information. Longer-term ionising radiation causes hard-errors in the electronic circuitry.
- Radio-links to spacecraft are often low bandwidth and high error-rate. This is not a good match to the relatively large configuration files .
- Limited on-board memory restricts the number of different configurations that can be stored.

Short-term solutions for configuration errors include readback and checking of the programming bitstream, or use of triple-redundancy voting circuits. Short-term solutions for configuration management include specialised compression techniques, and differential configuration formats (relative to an on-board default configuration). Long-term solutions require special 'space-friendly' FPGAs which provide on-chip configuration error-detection and/or correction circuitry which operates continuously and unobtrusively. Techniques will need to be developed to identify permanently faulty logic blocks within an FPGA. Techniques are needed to allow existing circuit designs to be reconfigured on-board the spacecraft to avoid faulty logic cells. This is impractical with current generation place-and-route software.

There is much scope for research into error-detecting or fault-secure logic circuit designs for FPGAs. A single chip or MCM combination of FPGA, microcontroller, flash memory configuration store, and digital and analog I/O circuitry would greatly reduce space mission weight and cost. We are building an experimental satellite payload to demonstrate the reconfiguration of FPGAs in space. The payload consist of A small (1kg) payload consisting of a microcontroller with RS422 communications, SRAM-based FPGA (Xilinx X4062), RAM, PROM, Flash Memory storage and an Adaptive Instrument Port. The payload will fly on the Australian experimental FedSat satellite, due for launch in late 2001.

In summary, we believe that Reconfigurable Computing has many advantages for space applications, and is an excellent match to new directions in low-cost, flexible space missions. Overall, there is potential for space-based computing to be a 'killer app' for reconfigurable computing technology.

Fault Tolerant Computing On Board Satellites
by Heiko Schröder

The University NTU of Singapore plans to build a satellite for the purpose of earth observation. The satellite will fly at an altitude of 800 km and could collect up to 400 Gbit of image data per orbit. But it can download only about 1/1000 of this data. Thus on-board image processing is required, which up to now has not been attempted in mini-satellites. A range of architectures are presented that can provide various degrees of fault tolerance in order to cope with damage caused by radiation. These architectures range from well established schemes like ‘shadow’ processors and ‘majority voting’ to toroidal architectures with 4-neighbourhood and 6-neighbourhood.

Systems in which processors can time-share between several instruments as well as systems with one processor per instrument with a range of spare processors catering for processor loss are presented. Finally the various systems are compared in terms of their atomic fault pattern sizes and in terms of their survival probability in the presence of a given number of faulty processors. Overall, it is shown that traditional schemes to cope with faults due to radiation can be outperformed easily if the processors are connected in regular fault tolerant networks.

Self Organisation in the μ-configurable Hardware POLYP by Uwe Tangen

DNA or RNA mediated self-replication involve highly refined enzymes or special environmental conditions and procedures. In between the biochemical experimental setup with its typical 10^{14} sequences of RNA, DNA plus enzymes, and evolving 10^3 to 10^5 computer programs — such as Tierra with a length of several tens of instructions, a large gap is apparent. The use of electronically evolvable hardware promises to be an intermediate research platform.

This lecture reports on a model situated between the biological and the artificial scenario using custom built hardware. POLYP is a second generation, massively parallel reconfigurable computer based on micro-reconfigurable Field Programmable Gate Arrays (Xilinx XC6264) with a large number of additional distributed memory circuits under local control. It is shown that self-organisation of simple state-machines indeed is possible with not only their descriptions evolving but concurrently working on these descriptions. Self-replicators emerge and evolution over long timescales shows the feasibility of online-electronic evolution.

Experiences with Reconfigurable Computing and Outlook on Further Work by Tobias Oppold

Dynamically reconfigurable architectures are getting more and more important for the future! For example, there will be a high demand for mobile applications in the upcoming years and low power consumption is therefore an important issue in future chip design. FPGAs already have this advantage over conventional microprocessors but in many cases they are too expensive to be used in industrial production. Besides many other advantages, dynamically reconfigurable architectures like NEC’s dynamically reconfigurable logic engine (DRLE) combine an improved performance/power ratio compared to processors with an improved area efficiency compared to FPGAs. The DRLE stores multiple configurations on-chip and can be reconfigured within a few nanoseconds on a fine-grained basis. Since the University of Tuebingen has much experience with high-level synthesis and mapping of object oriented descriptions into reconfigurable architectures, we are highly encouraged to investigate how our techniques can be applied to this new architecture.

The Third Way: Neither Hardware Nor Software
by Gordon Brebner

The words 'hardware' and 'software' have various connotations - of circuitry, architectures and hardness in the first case, and of programs, architectures and softness in the second case. Dynamically reconfigurable architectures do not fit into either category, so a 'third way' is needed. The bigger picture is one where classical networked von Neumann architectures are running out of steam, in the face of new technological developments, such as system-on-chip, configurable logic arrays and photonics. A future view of computing systems is as networks of diverse programmable information handling components, from the chip level upwards. Information may be digital, analogue, approximate, etc. and components may be electronic, photonic, micromechanical etc. Communication is the key concept.

Thus, the third way encompasses ways of expressing diverse programmability and diverse logical and physical connectivity, at apt levels of abstraction. Programmability includes both human-programmed control flow and data flow, and self-programming in adaptive systems, applying equally to computation and communication as these concepts merge. Connectivity is concerned with the absolute and relative positions of the communicating components — determined perhaps by external-world factors, the human programmer or by automated process. Expression of parallelism and interaction is extremely important.

In devising the third way, one should resist existing abstractions that may be too abstract and also inventing attempted complete solutions, but rather identify clear stepping stones en route, each with solid formal underpinning. One must be prepared to invent new information handling models for this.

Makimoto's Law, the 2nd Design Crisis
and the Future of Reconfigurable Computing
by Reiner Hartenstein

Panelists currently discuss the 2nd VLSI design crisis: skyrocketing design cost and shrinking product lifetime, mostly of hardwired accelerators. Two remedies are needed to cure the 2nd design crisis:

- Reconfigurable accelerators for flexibility to reduce specialization and increase product lifetime.

- A transition from net-list-driven synthesis to the application of real compilation techniques instead of 'synthesis' and so-called hardware 'compilers'.

However, the CAD industry is still mainly using methods of the Stone Age (Rubylith Age): the first wave of Silicon Application (the first 20 years after market introduction of the integrated circuit) due to the Makimoto/Tredennick wave model of silicon IC usage history. The second (20 years) wave (the Bronze Age) brought the microprocessor, microcontroller and memory to silicon IC application: the so-called von Neumann Machine Paradigm serves as the enabler of efficient compilation techniques. But the ASIC hardware still uses Stone Age CAD methods.

The third wave, the now beginning SoC age, brings FPGA usage into the mainstream. Analysts and experts claim that reconfigurable core usage is an indispensable ingredient of future SoC design methods. But we now still use stone age CAD methods, not only for ASIC hardware, but also for (re)configuration of reconfigurable circuits. It is time to come up with a (non-von Neumann) machine paradigm for reconfigurable machines, as an enabler for real compilation techniques being much more efficient than net list pushing. Compilation techniques to generate code for reconfigurable accelerator machines are a way out of the current 2nd design crisis.

2.3 Tuesday

Microprocessors and Reconfiguration
by Klaus Waldschmidt

Microprocessor usage in PCs and workstations will fade away compared to the usage in embedded systems as a system on a chip (SOC) solution. An embedded system has typically an analog/digital structure with the digital core of the microprocessor and some mixed signal glue circuits. Selecting an approach for a microprocessor implementation in embedded systems depends on the architect's ability to correctly model the effects of new technologies, new applications, new software (compilers) and CAD tools. Successful microprocessor implementation depends mainly on bringing together the advances in technology, knowledge of computer architecture and the potential of reconfiguration.

Starting with a short resume of the main technology trends and tradeoffs, a new structure model of today's microprocessor architectures is presented. Architectures have been classified in this model by a 'class box'. By this class box, all architectural and implementation alternatives at the microarchitectural level can be explained. This model serves also for an understanding of potential usage of reconfigurability within microprocessor design, which can be done either at run, load or production time.

VLIW, Superscalar, Direct Issue or Dependent Based Superscalar, TTA and ADARC are evaluated by this class box with respect to complexity and reconfigurability.

Configurable System-on-Chip
by Rainer Kress

The rapidly growing market for web-enabled consumer electronic devices introduces a paradigm shift in embedded system design. Traditionally, embedded systems have been designed to perform a fixed set of previously specified functions within a well-known operating environment. After shipment, the functionality of the embedded system remains unchanged during product lifetime. However, with shorter time-to-market windows and increasing product functionality this design philosophy has exhibited its shortcomings. The key features of next-generation embedded devices will be the capability to communicate over networks and to adapt to different operating environments. There is an emerging class of systems, which concurrently execute multiple applications, such as processing audio streams, capturing video data and web browsing. These systems need to be adaptive to changing operating conditions. For instance, in multimedia applications the video frame rate has to be adjusted depending on network congestion. Likewise, for audio streams, different compression techniques are applied depending on network load. Besides this class of multi-function systems, there are multi-mode systems, i.e. systems which know several alternative modes of operation, for example a mobile phone which is able to switch between different communication protocols, or a transmitter which can toggle between different encryption standards. This paradigm shift in functional and non-functional requirements of embedded appliances not only holds for consumer devices. In industrial automation, there is a growing demand for sensor and actuator devices which can be remotely controlled and maintained via Internet.

In the following, we briefly highlight three examples of how Infineon copes with the above mentioned requirements. A product, a design study, and a research project are presented. The CARMEL Core is the first in a family of 16-bit, fixed-point digital signal processing (DSP) cores that target advanced communications and consumer applications. Its modular design architecture allows for complete system-on-a-chip (SoC) implementations using advanced Electronic Design Automation (EDA) methodologies in an integrated development environment. The Configurable Long Instruction Word (CLIW) architecture sets the core apart by allowing Very Long Instruction Word (VLIW) performance at the low cost of traditional DSP architectures. The new CARMEL 20xx DSP allows system-on-chip (SoC) designers to extend the functionality of the built-in execution units with the addition of PowerPlug modules. This unique feature makes the Carmel DSP 20xx architecture scalable and configurable at the SoC level.

In a design study, Infineon combined a fine-grained Flash-based FPGA with a 16-bit controller on a single chip. Having the possibility to reconfigure the peripherals of the controller also after product launch opens the market for hardware and software adaptable systems.

JaCoP (Java driven codesign and prototyping environment) is targeted to suit as a complete design environment for embedded systems including also dynamically reconfigurable hardware components. JaCoP is based on Java, which is used for specification and initial profiling, as well as for the final implementation of system software. In the talk, we gave an overview of the implemented codesign flow, and we presented a tool for managing the interaction of hardware and software components.

Infrastructure of PCI Pamette by Mark Shand

PCI Pamette is a modest sized reconfigurable PCI add-in board. It is designed primarily to aid a conventional processor in I/O oriented tasks.

This talk describes the infrastructure of the PCI Pamette board, from its origins in earlier reconfigurable work at Digital, notably DECPeRLe-1, to its possible future directions. Infrastructure covers compilation aids, board control circuitry, the board's software runtime library, and a number of utility programs. Wherever possible, the discussion will be motivated by practical examples drawn from actual PCI Pamette applications.

Asynchronous Reconfigurable Computing
by Simon Taylor

As driving clocks becomes more difficult as the clock frequency increases, asynchronous computing architectures provide a promising alternative to synchronous systems. The speed-independence of asynchronous circuits allows greater exploitation of dynamic reconfiguration.

This talk describes the necessary architectural features for implementing asynchronous designs and highlights the problems with synchronous FPGAs which make them unsuitable for this task. Two early asynchronous FPGAs are described before a more detailed description of STACC, a self-timed version of the Xilinx XC6200, is given. The talk concludes with a summary of my current research.

Embedding Express Graphs into Networks
by Manfred Kunde (joint work with Heiko Schröder and Martin Fürer)

In an ATM network, an express graph is built on the top of the physical network in order to make the network faster. An express edge (a, b) is built by connecting a physical path from a to b in the original network to one edge in the modified network. By establishing several express edges, an embedded express graph is generated. The aim is to construct an express graph with a very small diameter.

We show that an arbitrary graph with N nodes and containing k directed Hamiltonian cycles has an express graph with a diameter of $O(kN^{1/k})$. The result can be extended to a class of graphs having bigger subgraphs with k Hamiltonian cycles. Furthermore, for the hypercube with N nodes, we give a new simpler construction for an express graph with diameter $O(\log N/ \log\log N)$ which asymptotically matches the lower bound.

Hardware Generation from Partitioned Regular Algorithms
with Resource Constraints
by Marcus Bednara (joint work with Jürgen Teich)

A lot of research on regular algorithms and their mapping to VLSI hardware was conducted in the 1980's, but the high degree of specialization caused extraordinary costs for real implementations. However, since reconfigurable logic devices became available, array processors can be implemented in a more cost effective way.

Our goal is to automate the design path for generating FPGA hardware from regular algorithms specified in a high level language. Several steps have to be performed: transformation into single assignment code, hardware resource specification (including hardware constraints), composition of a mixed integer linear program (MILP) which provides solutions for projection vectors and schedule vectors that are optimal, e.g. with respect to the array latency and resource usage. In order to achieve an optimal timing as well as short design cycles, the regular structure of the algorithm must be preserved when generating hardware. This requires a proper specification of placement constraints, especially for the FPGA target technology. Currently, we work on two back-ends for our design tool: one for synthesizable structural VHDL and one for the JBits API to the Xilinx configuration bitstream.

Optoelectronics in the reconfigurable environment
by John Snowdon (joint work with Keith Symington, and Ben Layet)

It is apparent that as the density of on-chip gates and ancilliary components increases that a solution must be found to the inherent communication problem that such a density creates. Optical communication has a number of advantages that can be exploited - free space, fibre, WDM, surface normal and long distance (more than 2 cm) at low cost. Several technologies exist to interconnect silicon through the optical domain, our preference being VCSELs or MQW modulators being flip-chipped (solder bumped) onto the top surface of the silicon. The bandwidth of such a system allows 10,000 channels at more than 10 Gb/s to be realised. The latency of entering the optical domain (and re-entering silicon) is also low at less than 5 ns. We have built several demonstrator experiments to illustrate these capabilites and to drive the technology further. These include pipe-lined image processors, dedicated sorters, telecom and datacom switches and neural networks. Our generic 'optical highways' scheme can in principle provide raw bandwidths of more than 10 Pb/s; the number being based on validated data extracted from our laboratory experiments.

The key to implementing the technology in a sensible way is the optical packaging. Recent developments in 'plug and play' freespace systems and in fibre optic image guides probably offer the best solutions. As feature size shrinks and inter (and intra) chip and MCM connection costs accelerate, it may also be true that optics provides a cheaper, higher performance alternative to any other interconnect technology.

Using Optoelectronic Interconnects for Run Time Reconfigurable Arrays
by Dietmar Fey

One of the major problems in current VLSI systems is limited bandwidth because of too few and too slow off-chip interconnects. The imbalance between satisfying on-chip computing power and insufficient off-chip communication performance has lead to a situation which is generally described as intra- and inter-connect crisis in microelectronics. One of the reasons for this crisis is that off-chip interconnects are located at the circuit's edge. A solution to this problem is offered by optoelectronic VLSI circuits. They exploit the whole chip area for communication by using a 2-D optoelectronic interface to eliminate pin limitation. Data rates of several 100 Gbit/s up to 1 Tbit/s are feasible in chip-to-chip communication. This will satisfy future requirements in processor-memory and processor-processor communication.

Massively optical interconnects for memory-processor communication are very attractive for reconfigurable architectures. We developed a test circuit consisting of an array of optically loadable look-up tables. Each look-up table is provided with an optical input pad. Using this 2-D optical input interface, all look-up-tables can be configured in the nanosecond range simultaneously. Parallel optical interconnects between a configuration memory and reconfigurable hardware circuits allows simple dynamic reconfiguration. This allows separation of reconfigurable logic and memory. Configuration memory must not be integrated within the reconfigurable hardware instead more area on chip is available for integration of configurable logic units. Such an approach is very beneficial to increase throughput and to speed up reconfiguration time in dynamic reconfigurable architectures like MorphoSys or the RAW machine, or smart memory systems as well. Also, the optical parallel interface can be exploited for rapid dynamically configuration of the interconnects in chunky unit architectures. Summarizing, optoelectronic interconnects are a very promising technology for run time reconfiguration in reconfigurable architectures.

eASIC, the Innovative Way to an Alternative to ASIC and FPGA
by Viorel Onofrei (concept by Zvi Or-Bach)

The only two available options up to now for a digital system designer are ASICs and FPGAs. FPGAs are preferred for prototyping and low-volume production and ASICs for high-volume production as imposed by the cost. eASIC is a concept developed by Zvi Or-Bach, founder and president of eASIC Corporation, and its central idea is merging the best of FPGAs (LUTs for ease of use) with the best of ASICs (high density achieved by fixed interconnect).

The structure of the basic cell comprises combinational logic (two three-input LUTs feeding a multiplexer) and a flip-flop. As a distinctive feature not to be found in other ASICs, the eCell contains an independent inverter. The fixed interconnect is achieved by customizing the top metal layers and provides density next to ASICs. The use of LUTs offers the flexibility for incremental rectification and also the possibility of hidden programmable interconnect. 256 eCells form an eUnit. In an eUnit, there is only a clock domain and a unique scan-chain. The eUnit can be configured as a 256x16 bits RAM block. 16 eUnits form an eASICore.

A comparison made with respect to density, performance, power, cost of design and prototyping, turn around time and debugging indicates the intermediate position of eASIC between FPGAs and ASICs. This comparison shows clearly that the eASIC is filling the gap between FPGA and ASIC and appears as a viable alternative in the quickly changing world of digital design.

2.4 Wednesday

Prototyping Environment for Reconfigurable Processors
by Sergej Sawitzki (joint work with Steffen Köhler and Rainer Spallek)

This work introduces a prototyping environment for reconfigurable microprocessors. First, a short overview of known reconfigurable processors is given. A systematic approach (concerning both the hardware and software parts) to designing, testing and debugging a reconfigurable pocessor is discussed. The prototyping environment consisting of Altera's UP1 board with memory daughter card and dedicated software (Motif GUI and extendable communication protocol) is introduced. The CoMPARE processor is used as the first prototyping example to test the static and dynamic reconfiguration approaches. Prototyping results confirm the simulation data, attesting to performance gains of factors 2 to 28 for different applications achieved through reconfiguration. The 8-bit prototype achieves clock rates of about 10 MHz on an Altera FLEX10K20 FPGA and has a power consumption of about 2 W. Further work concentrates on the development of the retargetable assembler and compiler to automate the code generation and support variable instruction sets.

Evaluating DSP with Dynamic Reconfigurable Processing Units
by Steffen Köhler (joint work with Sergej Sawitzki and Rainer Spallek)

The processing of streaming data can be efficiently done by application specific digital signal processors. In several application domains (e.g. audio and video processing), the performance of these DSPs is not always acceptable, being only sufficient for a very limited scale of special algorithm implementation. Another fact is the problem of low average utilization possibilities of multiple DSP units in state-of-the-art VLIW or SIMD architectures, which causes low functional execution density on silicon. Dynamically reconfigurable DSP execution units provide a more flexible and efficient way for implementation of special purpose application system designs. In this presentation, we discuss the integration of dynamic reconfigurable structures and units in a standard SIMD/VLIW DSP environment. Design elements are considered as hardware and software components. We further introduce a retargetable compiler concept, which provides a flexible method for hypothetic DSP architecture component space exploration. To underline the advantage of reconfigurable DSP comonents in practice, we show some implementation results for a minimal reconfigurable processor. Finally, an outlook is given on the usability of new reconfigurable IC products for prototype implementation of our considered reconfigurable DSP systems.

The P2NC Project - Or How Much Parallelism is There?
by Yosi Ben-Asher

In this work, we consider the problem of speeding up the execution of sequential programs by means of parallelization. Good solutions to this problem are essential to the possibility of keeping producing computers that are faster than those available today. Modern processors rely on Instruction Level Parallelism (ILP) as a way to speedup the execution of sequential programs. However, experimental studies suggest that ILP might be limited.

The P2NC is a novel tool which compiles sequential programs into boolean circuits and then executes them in parallel using a probabilistic algorithm. It is based on the observation that parallelism is best exposed when programs are compiled into circuits. The goals of the P2NC project are:

- To produce upper bounds to the amount of parallelism that can be potentially extracted from actual programs using ILP techniques.
- Evaluate the potential to improve ILP involved with direct compilation of programs to circuits.
- The probabilistic algorithm used by P2NC can lead to a new type of CPU which is not based on the Von-Newman architecture. Moreover, fast evaluation times of this algorithm may suggest ways to attack the well known circuit evaluation problem, namely, constructing a fast circuit that can evaluate polynomial circuits in a logarithmic number of parallel steps.

An initial version of P2NC is currently being implemented and we already have some preliminary results. The large size of the resulting circuits, and the long evaluation times, requires that we use a powerful workstation for the experiments.

Reconfigurable Accelerators for Combinatorial Problems
by Marco Platzner

Many combinatorial problems show great amounts of data parallelism at the bit level, which makes them natural candidates for implementation in fine-grained FPGAs. However, combinatorial problems are often solved by control flow dominated search methods, such as backtracking or branch and bound. Recently, a number of architectures have been presented that implement these search algorithms in reconfigurable devices and exploit the massive data parallelism by compilation to instance-specific hardware.

In this talk, I present different architectures to solve the Boolean satisfiability problem. Simulation of these architectures shows that for examples from the DIMACS benchmark suite, high raw speed-ups over state-of-the-art software can be achieved. I present a design tool flow and prototype implementation of an instance-specific satisfiability solver and discuss experimental results. The overall speed-up of the instance-specific architecture was measured, taking the hardware compilation time into account. The results prove that many of the DIMACS examples can be accelerated with current FPGA technology.

Cellular Automata with Dynamically Reconfigurable Buses
by Thomas Worsch

We consider one-dimensional cellular automata which are extended by dynamically reconfigurable buses (RCA). This is a generalization of the bus automata model suggested by

Rothstein (1976). It is shown that up to a constant factor it does not matter whether bus segments are directed or not, as far as the time complexity is concerned. For both variants of the model it can be characterized in terms of the reversal complexity of one-tape one-head Turing machines.

The comparison of RCA with tree-shaped CA shows that the former are in the second machine class (according to van Emde Boas's taxonomy) and that they can be transformed in some simple normal form with an only polynomial overhead of time. In this normal form, adjacent cells are connected by only two bidirectional bus segments and during the computations only linear buses are configured on which in each step at most one cell is sending some information ('exclusive write').

2.5 Thursday

Why and How should we use FPGAs to run Mobile Code?
by Frederic Raimbault

Mobile code is a central concept of distributed systems. It has evolved from the client-server paradigm (e.g. SQL transactions) to the code on demand paradigm (e.g. Applets). The next paradigm becoming more and more used is the mobile agent paradigm, where the code is moving autonomously from host to host. This programming paradigm offers an interesting solution to the network overload: programs and computations are migrated to where the data lies instead of moving the data. Several applications in electronic services, information retrieval, telecommunications, groupware, etc. are already applying this programming model. To be effective, a homogeneous runtime has to be provided on each host receiving such mobile agents. The leading technology is Java; its Java Virtual Machine provides the needed platform independence. But it suffers from several drawbacks: software complexity, greed for computing resources, low speed execution, unsecured access. The worst one is its inability to evolve. The bytecode is fixed, bounded to the Java language without any specialization possibility. It is as rigid as a hardware Java processor.

We propose to develop a concept of Dynamically Reconfigurable Virtual Machine (DRVM) to address the adaptability problem and to implement it on FPGAs to improve speed execution. The target system is composed of a PC connected to a FPGA board. The PC is in charge of receiving the mobile code, (re)configuring the FPGA board and launching the execution. The FPGA is in charge of executing the mobile code and has access to a subset of host resources. The programming model is object oriented, classed based, with static and strong typing. The execution model is entirely distributed. It is built around object factories; each of those is responsible of creating, memorizing and applying operations of one class of the program. Primitives classes are implemented statically by predefined primitive factories that manage values. User classes are dynamically configured by code instructions into user object factories. The outcome of this execution model is that computation takes place where the object lies, so modularity, scalability and concurrency properties should be achieved. The key issue is the distributed control management overhead.

We are actually simulating the functionality of this model and testing distributed control management. The next phase will be to implement our execution model into FPGA, thanks to the Specialized Virtual Configurable Array concept of D Lavenier.

Specialized Virtual Configurable Arrays
by Dominique Lavenier

Today, the implementation of an application on a reconfigurable platform (accelerator) suffers from two major limitations. First, implementations are not portable: a design developed for one specific board cannot be directly re-mapped onto another board. Significant changes always occur because of the different structures (architectures) of the boards, the use of different technologies, the way the boards are interfaced, etc. Second, the implementation requires a long and a tedious conception time: traditional frameworks start from a VHDL description and loop on the compilation/debug/synthesis/place-and-route steps. These limitations are not compatible with the Reconfigurable Computing concept, which advocates the speed of hardware together with the flexibility of software. The solution we propose, to free us from these limitations, is based on the concept of virtual hardware for portability and specialization of the architecture for programming efficiency.

Software Portability on Reconfigurable Chips
by Bernard Pottier (joint work with Loic Lagadec)

In the context of the increase of hardware resources, the development time could be the major problem for the economic viability of integrated applications. Object Oriented methodologies and analysis allows reduction of complexity and cost due to reuse and modularity. We are investigating their applicability to system and hardware design and synthesis.

First, a smart sensor system has been programmed 100the compiler producing the code for the platform, which includes a CPU/FPGA and MAPP2200 (see http://www.ivp.se). The camera is downloaded and controlled by a remote station that exchanges flattened objects. Second, concerning the programmability of reconfigurable logic, it is proposed to model directly architectures in OO environments in order to obtain portable system control and basic software. A UML model or a small grammar could be enough to characterize the architectures and have generic basic functions working.

We are also developing a medium grain model for computations that will allow a system or a compiler to translate and allocate hardware resources. Translation of blocks to reconfigurable parts involves mapping to the cell size (SIS package), then place and route. Structural circuits are described by binding on variables and constructed by the drawing support. We provides a few statistics and views of synthesized circuits. In conclusion, we insist on the interest of the community investigating portability and the viability of a standard for reconfigurable architecture descriptions.

Lava
by Satnam Singh (joint work with Mary Sheeran)

Lava is an experimental HDL which has several properties useful for designing dynamically reconfigurable systems. Lava is embedded in the lazy functional programming language Haskell.

One useful property of Lava is the ability to effectively lay out regular networks using higher-order combinators. Usually for dynamic reconfiguration, one has to have a lot of control over placement and Lava provides this ability without resorting to clumsy calculations of cartesian coordinates. Lava also provides support for specifying a very specific kind of dynamic reconfiguration from a high level description. By using partial application we are able to specify dynamic circuit specialisation. This can be thought of as run-time constant propagation through hardware. A complex system which takes such high level specifications and automatically produces hardware and run-time software for effecting this partial evaluation has been built using an XC6200 based system. Experiments have been performed on multipliers which have been specialised at run-time to yield constant coefficient multipliers.

The FURI Runtime System
by Adam Donlin

In this talk, we introduce the concept and implementation of a runtime system for a self-modifying processor architecture: the Flexible Ultimate RISC (FURI). The FURI core is presented as an evolution of the simple, minimalist, transport-triggered style architecture of the Ultimate RISC. The FURI processor supports a minimal instruction set architecture, comprising a single word move instruction. Elements from the core of a standard RISC processor are migrated onto the system bus of the FURI core. Computations may be performed by orchestrating the flow of operands from memory, across the system bus, and through the memory mapped registers of the system bus elements.

In mapping the configuration RAM of the host FPGA into the memory map of the FURI core, we describe the creation of an autonomous, self-modifying processor. The FURI core exploits this self-modifying capability to dynamically alter the contents of the system bus and hence implement 'virtual circuitry'. Building on this, we present details of a flexible toolset for targeting and debugging code and circuitry for the FURI system. From here, we then describe the implementation and challenges of developing an operating system contained entirely within the FURI environment. Key features of this operating system, such as its support for flexible protocols between the FURI core and the external environment, are discussed.

Reconfigurable Functionality - The OS Perspective
by Michael Dales

In an attempt to bring FPL to the masses, there have been several attempts to introduce FPL into a desktop machine, ranging from simply an FPGA on a PCI card, to placing

FPL inside the heart of a microprocessor. However, most of this work concentrates on the issues at the bit level, spending little effort considering how this new resource will be manage by the software which runs upon it.

An Operating System (OS) provides an application with an abstraction of the hardware environment, but the hardware in turn needs to support this abstraction (e.g hardware support for virtual memory). In this talk, I present some of the issues I see, as an operating systems person, with the current attempts to integrate FPL into the desktop machine. This will include an outline of the OS's duties in managing a resource, then outlining some of the problems that will need low-level aid, thus requiring support from the integrated hardware. IT is likely that designers of integrated systems will need to provide this support if desktop FPL implementations are to become prevalent.

Operating System Support for Dynamically Reconfigurable Architectures (Run-time Partitioning for Just-in-time Compilation)
by Oliver Diessel (joint work with Grant Wigley and David Kearney)

The growing size of reconfigurable logic resources, the growing range and integration of applications, and the increasing orientation towards real-time tasks suggests considering the viability of multitasking reconfigurable systems. The question then arises: how to manage/support large reconfigurable logic resources in a multitasking environment. We believe the design of operating systems for multitasking reconfigurable systems will need to decide whether the reconfigurable logic resource should be space- or time-shared. In particular, should space-sharing be considered, it is necessary to decide whether to support fixed or variable partitioning of the reconfigurable logic resource. While fixed partitioning simplifies design and management, performance can suffer. On the other hand, variable partitioning may support optimal task performance at the cost of scheduling complexity. For example, to reduce the time tasks wait to enter the system, executing tasks might be moved about, or waiting tasks might be adapted to the amount of resource available at load time. However, this adaptation requires fast partitioning, placement and routing algorithms in order to complete the physical design of tasks at load time. Our research seeks to develop accurate and efficient indicators that can tell us whether to accept or reject real-time reconfigurable computing tasks into common architectural models. The long-term goal is to develop algorithms and techniques that allow us to complete the physical design of tasks without contributing significantly to overheads.

Task Rearrangement on Partially Reconfigurable FPGAs
by Martin Middendorf
(joint work with Hossam El Gindy, Hartmut Schmeck and Bernd Schmidt)

Partially reconfigurable FPGAs can be shared among multiple independent tasks. When partial reconfiguration is possible at run-time, the FPGA controller can decide on-line where to place new tasks on the FPGA. Since on-line allocation suffers from fragmentation, tasks can end up waiting despite there being sufficient, albeit non-contiguous resources available to service them. Rearranging a subset of the tasks executing on the FPGA often allows the next pending task to be processed sooner. In this paper, we study the problem of placing and rearranging tasks that are supplied by input streams which have constant data rates. When such tasks are rearranged, the arriving input data have to be buffered while the execution is suspended. We describe and evaluate a genetic algorithm for identifying and scheduling feasible rearrangements when moving tasks are reloaded from off-chip and buffer size is limited.

The Von Neumann Bottleneck and Other Myths
by Mark Shand

The goal of this talk was to stimulate discussion by arguing that many premises held true in the reconfigurable computing community regarding mainstream computer systems are in fact myths. We further argued that conventional systems have more in common with reconfigurable systems than is commonly granted, with the goal of this line of reasoning being to refine our ideas of what constitutes a reconfigurable computer.

2.6 Friday

Dynamically Reconfigurable Logic and System on Chip
by Patrick Lysaght

This goal of this presentation was to identify opportunities for new research into dynamically reconfigurable logic in the context of recent developments in deep sub-micron (DSM) and system on chip (SoC) technologies for mask programmed ASICs. Dynamically reconfigurable logic was introduced and an outline of the research into design methods and CAD tools for dynamically reconfigurable logic, at the University of Strathclyde, was presented. The Dynamic Circuit Switching (DCS) CAD framework for design capture, simulation, controller synthesis, technology mapping and verification was reviewed. The options for system on chip with FPGAs were considered and it was argued that dynamically reconfigurable logic remains relevant, even for the largest FPGAs.

The manner in which deep sub-micron (DSM) and system on chip (SoC) technologies are transforming ASIC design methodologies and CAD tools was described. The changes needed to address issues such as design productivity, timing and power closure, system verification, and manufacturing test were identified. It was argued that similar methodology changes are also needed for the design of dynamically reconfigurable logic. Specific instances of this trend include the need for closer coupling between logical and physical design phases, greater exploitation of logical structure and hierarchy, and the use of block-based design flows. It was concluded that DRL could benefit significantly from the catalytic influence of DSM and SoC.

Finally, four new opportunities for research into FPGAs and dynamically reconfigurable systems were identified. These included:

- New soft, parameterisable FPGA cores.

- Exploration of the concept of dynamically reconfigurable intellectual property cores.
- New FPGA architectures that incorporate communications networks for SoC.
- The use of reconfigurable FPGA cores in SoC ASICs as test access mechanisms (TAMs) during manufacturing test and application logic during normal system operation.

APPLES: An FPGA-based Application Specific Processor
by Damian Dalton

Parallel processing is only effective, as manifested in solving a problem more rapidly compared to a single sequential processor solution, if the parallel architecture and the parallel algorithm coincide with the intrinsic parallel nature of the problem. Many parallel processors have accelerated the solution time for numerous problems. However, the quest to reduce even further the computation time and find parallel solutions to those that have been elusive to such approaches remains.

Unfortunately, often parallel algorithms are extracted from sequential solutions without necessarily exploring options within the parallel domain. Frequently, the sequential solution is unsuitable for parallelisation and the target architecture inappropriate. New architectures and algorithms generated from adopting a new processing paradigm can present innovative and faster solutions. Logic Simulation is one such problem and the new design paradigm can be evaluated and implemented in Reconfigurable technology (Xilinx Virtex). In logic simulation, logic gates are characterised by their switching behaviour, which can be simple or quite complex in nature. The intrinsic problem involves mass computation of basic Boolean operations requiring fine grain processors. Typically, relatively large coarse grain processors are incorporated into the solution, reducing the amount of possible parallel processing and increasing overheads and the associated interprocessor communication. This severely impedes the speedup of these systems, limiting it to a value in many cases below a factor of ten.

APPLES (Associative Parallel Processor for Logic Event-driven Simulation) is a novel an alternative solution for Logic Simulation. Effectively, the architecture consists of a one-to-one correspondence between processors and logic gates by utilising an associative memory structure for holding signal values. Furthermore, gate processing is executed in memory in parallel. Speedup of the system is substantially higher than coarse grain processors, gates are evaluated within a few machine cycles and additional accelerated enhancements are possible.

This processor is only possible through the reconfigurability aspects of Field Programmable Gate Arrays (FPGAs) allowing prototypes to be implemented and explored. Even as a platform for final realisation, the FPGA-based parallel processor exceeds in speed the performance offered by large grain ASIC processors. This supports the view that FPGAs have the capability to be more than just the "Glue logic" of the digital environment, permitting viable and competitive application specific processors to be investigated and implemented.

Run-time Reconfiguration of AC Drive Controllers
by Vásárhelyi József

There are different approaches to defining reconfigurable systems, e.g. 'Reconfigurable Computing technology is the ability to modify a computer system's hardware architecture in real time'. Field Programmable Gate Arrays (FPGAs) are widely used in digital signal processing. In some applications, they perform better than DSPs. Also, there are known dedicated DSP processors for digital motor control. Comparing the number of applications known in the reconfigurable field, just a few of them are concentrated in the study of vector control for AC drives. Some successful implementations of vector control are referred to in the corresponding literature. A DSP implementation of speed-sensor-less induction motor drive using artificial intelligence is presented by Vas. Unfortunately, all these implementations, and especially their hardware structures, do not correspond to the reconfigurable system paradigm. The implementation of an efficient vector control algorithm for a single AC machine in a DSP processor is no longer a problem. Difficulties arise when one tries to extend this implementation to multiple-machine control or the need appears for implementation of a reconfigurable controller for the same machine. A solution for the multiple-controller implementation is presented in this paper, using Triscend's Configurable System on a Chip.

The necessity of reconfiguration is based upon the practical observation that the performance of different types of vector controlled drives are different, depending primarily on the range of speed. It is known that the rotor flux oriented vector control is more easily implemented and therefore more widely used. One drawback of this method is the low efficiency at low ranges of speed. For lower speed range, the stator flux oriented vector control is preferred. The control system presents modularity. This modularity allows exploitation of all the parallelism of the control algorithm. The method was introduced as Synchronous Vector Control. The introduction of the reconfigurable controller concept solves the problem. In fact, the same hardware support that implements one controller structure can be used not only to implement, but also to switch to another control scheme. In this way, the disadvantage of using adaptive control can be avoided.

Each control structure can be seen as a distinct state of a state machine. In fact, each state represents a different hardware configuration. Depending on the hardware support of the implementation, the reconfiguration can be done as partial reconfiguration or total reconfiguration. The controller states are quantified. Reconfiguration has to be done between two sampling events. Reconfiguration time has to be less than or equal to the sampling period of the controller. It may became critical and there is the need for reconfigurable structures with faster reconfiguration time.

Solution of Heiko's Problem or Cyclic Cutwidths of Meshes
by Ondrej Sýkora (joint work with Heiko Schröder and Imrich Vrťo)

The cutwidth problem is to find a linear layout of a network so that the maximal number of cuts (*cw*) of a line separating consecutive vertices is minimized. A related and more natural problem is the cyclic cutwidth (ccw), when a circular layout is considered. The main question is to compare both measures cw and ccw for specific networks, whether adding an edge to a path and forming a cycle reduces the cutwidth essentially. We prove exact values for the cyclic cutwidths of the 2-dimensional ordinary and cylindrical meshes $P_m \times P_n$ and $P_m \times C_n$, respectively.

Especially, if $m \geq n + 3$, then $ccw(P_m \times P_n) = cw(P_m \times P_n) = n + 1$ and if n is even then $ccw(P_n \times P_n) = n - 1$ and $cw(P_n \times P_n) = n + 1$ and if $m \geq 2, n \geq 3$, then $ccw(P_m \times C_n) = \min\{m + 1, n + 2\}$.

Motivation for the (cyclic) cutwidth problem comes from many areas of computer science: rearrangeability, parallel and distributed computing, all-optical networks, etc.