Jürgen Becker, Reiner W. Hartenstein, et al.: A Reconfigurable Parallel Architecture to Accelerate Scientific Computation; Proc. of Int. Conf. on High Performance Computing, New Delhi, India, December, 1995

A Reconfigurable Parallel Architecture to Accelerate Scientific Computation

Jürgen Becker, Reiner W. Hartenstein, Rainer Kress, Helmut Reinig

Department of Computer Science
University of Kaiserslautern
D-67663 Kaiserslautern, Germany
abakus@informatik.uni-kl.de

Abstract

This paper introduces a new reconfigurable accelerator for high performance computing at a moderate price. By using a new machine paradigm, this accelerator is especially suited to scientific computations, where the hardware structure can be configured to match the structure of the algorithm. The MoM-3 efficiently uses reconfigurable data-paths to provide instruction-level parallelism, and multiple address generators to have the complete memory bandwidth free for data transfers (instead of fetching address computing instructions). Speed-up factors in the range from 7 to 2000 have been obtained for different kinds of algorithms, compared to state-of-the-art workstations. This allows to achieve supercomputer performance from a low-cost workstation, by adding the MoM-3 as an accelerator.

1 Introduction

The Map-oriented Machine 3 (MoM-3) is an architecture based on the Xputer machine paradigm [4]. Its motivation is to provide a hardware architecture, which matches the regularities found in many performance critical algorithms. Quite often, a block of always the same operations is applied to a large amount of data, showing generic address sequences in the data accesses. Such algorithms can be addressed efficiently with the following architecture.

Instead of a hardwired ALU with a fixed instruction set, an Xputer has a reconfigurable ALU (rALU) based on field-programmable devices (figure 1). All data manipulations, which are performed in the loop bodies of an algorithm, are combined to a set of compound operators. Each compound operator matches a single loop body and takes several data words as input to produce a number of resulting data words. The compound operators are configured into the field-programmable devices. After configuration, an Xputer's "instruction set" consists only of the compound operators as they are required by the algorithm actually running on the Xputer.

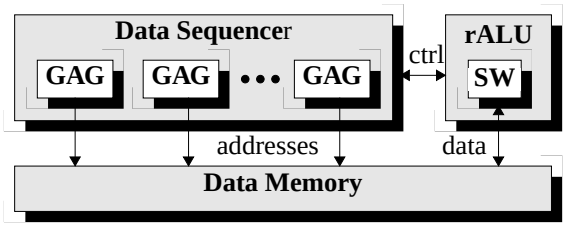

Figure 1. Block Diagram of an Xputer

Since many scientific algorithms compute array indices in several nested loops, the sequence of data addresses in a program trace shows a regular pattern. This leads to the idea to have complex address generators compute such address sequences from a small parameter set, which describes the address pattern. The address generators (GAG) automatically activate the appropriate compound operator, each time a new set of input data is fetched from memory and the previous results have been written back. This so-called data sequencing mechanism directly matches the loop structure of the algorithm, where the index computations serve as a means to provide the right data to ever the same operations.

The following section introduces the hardware structure of the MoM-3, including the address generators and the rALU devices. Afterwards, the features of the C compiler for the MoM-3 are outlined. The fourth section explains the way algorithms are executed on the MoM-3 by means of an example. The final sections provide a performance comparison and conclude the paper.

2 The MoM-3 Architecture

MoM is an acronym for Map oriented Machine, because the data memory is organized in two dimensions like a map. The MoM-3 efficiently supports a reconfigurable ALU (rALU), because the data sequencing mecha-

nism shows only a loose coupling to the ALU structure. All input and output data to the compound operators in the rALU is stored in so-called Scan Windows (see figure 1). A Scan Window (SW) is a programming model of a sliding window, which moves across data memory under control of a data address generator. All data in the Scan Windows can be accessed in parallel by the rALU operator. The rALU operators are activated every time a new set of input data is available in the Scan Window. This data sequencing mechanism is deterministic. So-called Generic Address Generators (GAGs) compute a generic sequence of data addresses from a set of algorithm-dependent parameters. The MoM-3 Data Sequencer contains several Generic Address Generators running in parallel, to be able to efficiently cope with multiple data sources and destinations for one set of compound operators.

In the MoM-3, the Data Sequencer is distributed across several computational modules (C–Modules), as can be seen in figure 2. The MoM-3 includes up to seven C–Modules. Each C–Module consists of a Generic Address Generator (GAG), an rALU subnet and at least two megabytes of local memory. All C–Modules can operate in parallel when each Generic Address Generator accesses data in its local memory only. The global MoMbus is used by the MoM-3 controller (M3C), to reconfigure the address generators and the rALU whenever necessary. Via the MoMbus-VMEbus interface, the host CPU has access to all memory areas of the MoM-3 and vice versa. The host CPU is responsible for all disk I/O, user interaction and memory allocation, so that the MoM-3 completely uses the functionality of the host's operating system.

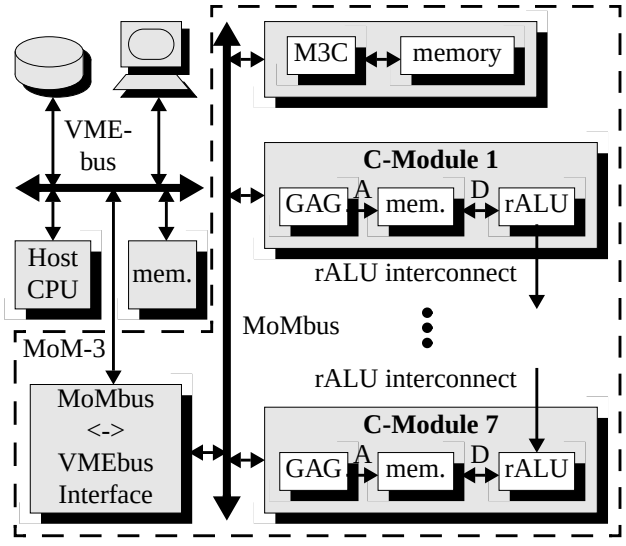

Figure 2. The MoM-3 Hardware

2.1 The Data Sequencer

The MoM-3 Data Sequencer consists of up to seven Generic Address Generators (GAGs) running in parallel and the MoM-3 controller. Each GAG controls one Scan Window. It operates in a two-stage pipeline. The first stage HPG (Handle Position Generator) computes two-dimensional handle positions for the Scan Windows. The sequence of handle positions (scan pattern) describes how the corresponding Scan Window is moved across the data memory (figure 3). Scan patterns may be nested and parallelized [4].

The second pipeline stage MAG (Memory Address Generator) computes an arbitrary sequence of offsets to the handle positions, to obtain the effective memory addresses for the data transfers.

The number of memory cycles required is substantially reduced by:

- avoiding addressing overhead by GAG hardware use
- problem-specific reduction of memory access (compiler generated access mode flags)
- support of interleaved memory possible (compiler generated storage scheme)

The hardware controlled generation of long address sequences allows to access data at the maximum speed of the bus protocol and the memory devices. With the VME-bus-like MoMbus this can be done in 120 ns, using 70 ns memory devices and a conventional, non-interleaved memory organization.

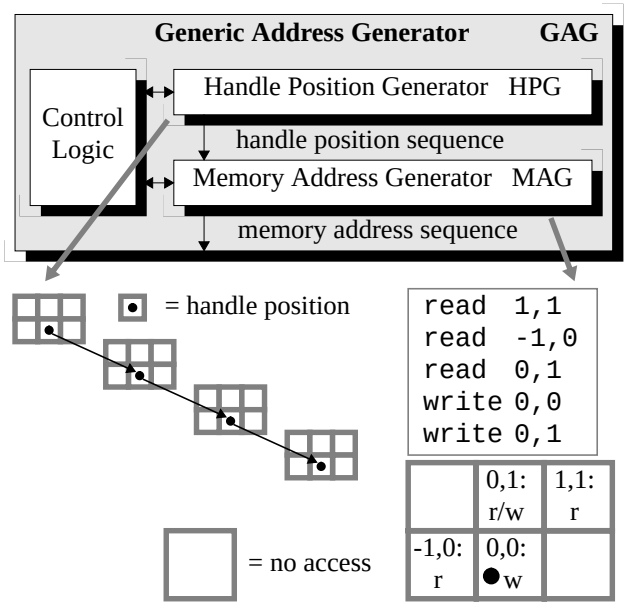

Figure 3. Generic Address Generator

2.2 The Reconfigurable ALU

SRAM-based field-programmable devices available commercially today are not well suited to implement arithmetic operations on wordlengths of 32 bits or more. This is because each bit of the datapath is configured separately, wasting a lot of silicon area for configuration memory. Furthermore, the available routing resources are far to restricted to have multiple 32-bit buses, as they are required for high-speed datapaths. Therefore, we developed our own architecture, the so-called rDPA (reconfigurable datapath architecture). It provides a small array of so-called Datapath Units (DPU), where each can be configured to implement any operator from C programming language, as well as some others, which are stored in an extensible library. The wordlength of a single DPU is 32 bits. The configuration is stored in RAM cells, which are directly addressable. This allows to perform fast partial reconfigurations.

The rALU subnet of a single C-Module is shown in figure 4. It contains an rDPA array made of eight rDPA chips, arranged in a two by four array. Each rDPA chip contains an array of four by four Datapath Units (DPU).

A Datapath Unit consists of a conventional ALU (add, sub, shift, logical operations), a microprogram memory and a sequencer for horizontal microcode. The microprogrammed control allows to implement complex operators like multiplication or random number generation with little hardware resources. Each Datapath Unit has two input registers, one at the north and one at the west, and two output registers, one at the south and one at the east. The registers in the Datapath Units in fact are a distributed implementation of the Scan Window of the Xputer paradigm.

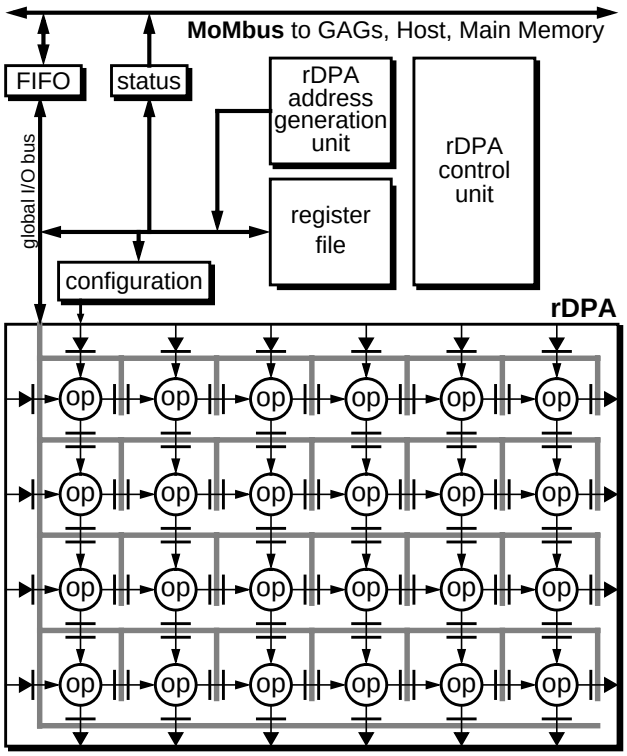

Figure 4. One subnet of the reconfigurable data-driven ALU

The Datapath Units can read 32-bit intermediate results from their neighbours to the west and to the north and pass 32-bit results to their neighbours in south and/or east direction. A global I/O bus allows input and output data to be written directly to a Datapath Unit without passing them from neighbour to neighbour. The Datapath Units operate data-driven. The operator is applied each time new input data is available either from the bus or from a neighbouring Datapath Unit. This decentralized control simplifies pipelining of operations, when each operation takes a different time to execute.

The array of Datapath Units extends across rDPA chip boundaries in a way that is completely transparent to the compiler software. To overcome pinout restrictions, the neighbour to neighbour connections are reduced to two bits wide serial links with appropriate converters at the chip boundaries.

The main features providing high performance and area efficiency by the rDPA are:

- pipelined complex expression execution
- pipelining beyond loop boundaries
- inter-expression parallelism
- rDPA I/O data stream optimization
- pipelining beyond rDPA chip boundaries reduce operator‘s time of waiting for external operands

In addition to the rDPA array, an rALU controller circuit interfaces to the MoMbus. It provides a register file as a kind of cache for frequently used data, and controls the data transfers on the global I/O bus. This is done by the rDPA address generation unit, which gives direct access to all input and output registers of all DPUs.

3 The Software Development Environment

Software for the MoM-3 can be written in a subset of C language, which excludes only pointers, operating system calls and recursive functions. Constructs requiring a dynamic memory management are to be run on the host computer. The C compiler automatically transforms such a sequential algorithm notation to make use of the fine grain parallelism of the Xputer paradigm. The results are

a Data Sequencer parameter file, rALU descriptions in ALE-X language (arithmetic and logic expressions for Xputers), and a data map description for an efficient distribution of data elements in memory. All these text files are processed by additional tools, which produce the MoM-3 binaries. This allows to program the MoM-3 at low level as well, by editing such text files appropriately. A more detailed description of the MoM–3 programming environment can be found in [9].

4 Example: Three-Dimensional Ising Model Simulations

The Ising model is used for the analysis of phase transitions. The Ising model can be used to explain, how short-range interactions between components of a larger structure (e.g. molecules in a crystal) give rise to a long-range, correlative behaviour, and to predict in some sense the potential for a phase transition.

An Ising spin system consists of a one-, two-, or three-dimensional lattice (figure 5). Each lattice point is associated with a so-called spin variable (or spin) S_i, which can take the values +1 (up) or –1 (down).The state of an Ising spin system consisting of N spins at a given moment is determined by the spin configuration {s}, i.e. the values of all N spins at the moment, $\{s\} = (S_0, S_1, \ldots, S_{N-1})$. Spins only interact with each other, if they are nearest neighbours in the lattice, connected by a line segment as shown in figure 5. A general introduction to the Ising model can be found in [2].

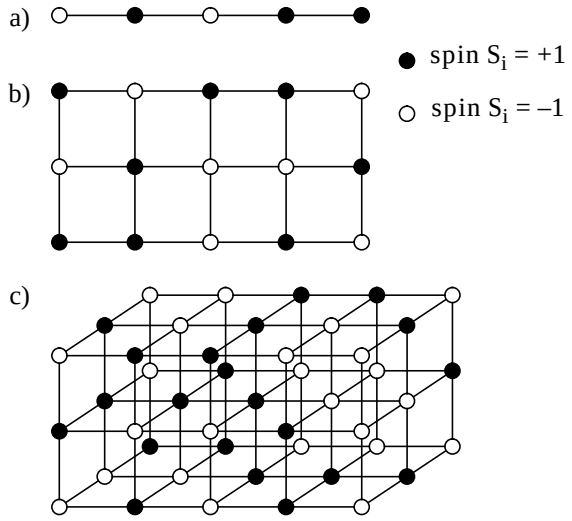

Figure 5. Ising model lattices: a) 1-D; b) 2-D; c) 3-D

Since analytic solutions cannot be computed for three-dimensional Ising spin systems, Monte Carlo simulations are performed, e.g. using the so-called heatbath algorithm [1]. Each spin S_i is set to the value +1 (up) with a probability $P(S_i=1)$ depending on the configuration of the interacting spins. For a three-dimensional lattice, six nearest neighbours have to be taken into account. Since the probabilities are time-independent, they can be computed beforehand and stored in a look-up table. The six interacting spins result in a look-up table of 64 entries for $P(S_i=1)$. The six values of the interacting spins S_k are concatenated to a six bit address to read the appropriate probability that S_i will become +1 from the look-up table. The probabilistic nature of a Monte Carlo simulation is introduced by comparing the probability $P(S_i=1)$ from the look-up table with a random number N, and setting S_i to +1 if and only if N $P(S_i=1)$.

One application of this procedure is called a spin update. About 17 x 10^6 sweeps through the complete lattice are required to complete a Monte Carlo simulation for one temperature value. A single sweep through the lattice requires a spin update at every lattice position. Near the critical temperature, where phase transitions are expected, several simulations at different nearby temperature values have to be run to obtain meaningful results.

4.1 MoM-3 Implementation

The simulation of up to three-dimensional Ising systems with nearest neighbour interactions can be handled efficiently on the MoM-3, independent from the size of the lattice. The spin configuration of the lattice is stored in the data memory. Since typical Ising simulations require multiple runs at different temperatures near the critical temperature, multiple C–Modules can be used most efficiently to perform separate Ising simulations s in parallel.

The value of a single spin variable S_i can be coded in a single bit, where a set bit represents a +1 (up) and a cleared bit represents a -1 (down) spin. To reduce memory and I/O requirements, 32 spin variables are packed into a single 32-bit memory word and transferred in a single memory access. The rALU manipulations on such a 32-bit block of spin variables are done in a pipeline following the heatbath algorithm described above. A whole compound operator for a single spin update fits into a bounding box of two by eight DPUs (figure 6). Therefore eight spin updates can be computed in parallel with the rDPA array of a single C-Module. The “up…” DPUs are used to unpack the 32-bit memory words to a sequence of

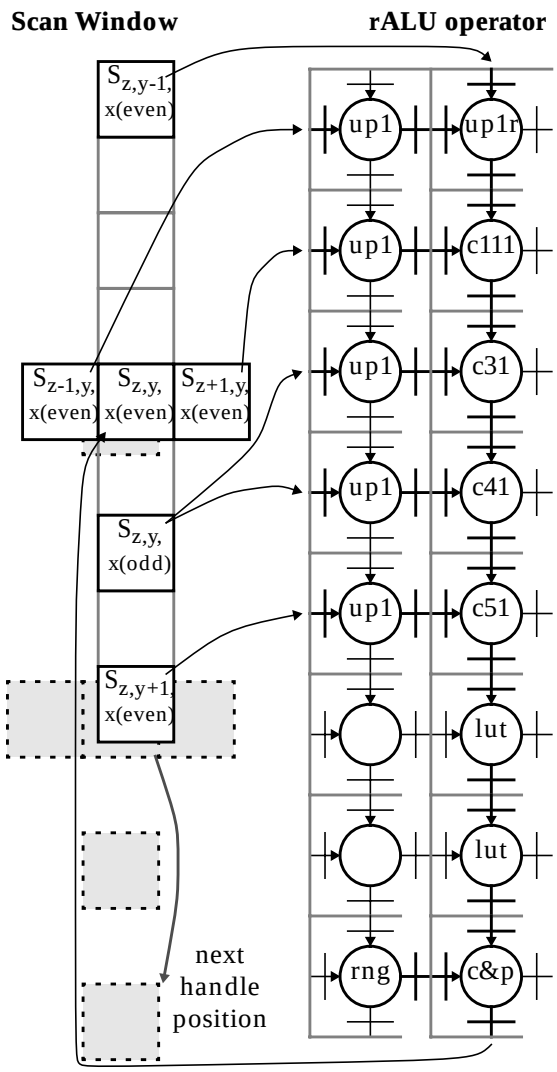

Figure 6. rALU operator for a single spin update

32 spin variables, which are fed into the pipeline. The "c…" DPUs are used to combine spin variables in the appropriate order to obtain the 6-bit addresses for the look-up table. Two "lut" DPUs contain two copies of the complete look-up table of $P(S_i=1)$ values. They are used in an alternating sequence, because it has turned out that their operation is time-critical. A 64-bit linear feedback shift register produces random numbers of sufficient period length in the "rng" DPU. Finally a "c&p" DPU compares the random number against the $P(S_i=1)$ value and packs 32 consecutive resulting spin variables into a 32-bit memory word.

Since 32 spin variables fit into a single memory word, four memory words make a complete line of 128 spin variables. A complete line of spins is updated in a pipelined fashion. Therefore, the next handle position computed by the Generic Address Generators moves four memory words ahead, as shown in figure 6. The data accesses to fill the pipeline are handled by the Memory Address Generators. Only a snapshot of these accesses is shown in figure 6.

5 Performance Comparison

With a 33-MHz prototype of the MoM-3, using serial links of two data bits at 33 MHz, the following performance can be achieved. Eight spin updates can be performed in parallel every 600 ns, resulting in 13.3 million spin updates per second with a single C-Module. Computing seven simulations in parallel, using seven C-Modules, this extends to 93.3 million spin updates per second.

A MoM-3 prototype based on state-of-the-art workstation technology could easily run at 50 MHz and utilize larger chip packages, allowing four data bits per serial link and a serial link speed of 66 MHz. Now the serial links are no longer the bottleneck, but the speed of the computations inside the DPUs. Such a "new technology" MoM-3 would produce eight spin updates every 390 ns with a single C–Module, corresponding to 20.5 million spin updates per second. Again seven C-Modules would allow the simultaneous simulation, resulting in 144 million spin updates per second.

In literature, several performance figures of implementations of the Ising model on different machine architectures have been reported. All of these are based on the same heatbath algorithm, explained in section 4. A special-purpose processor has been built by Pearson, Richardson, and Toussaint [7] from off-the-shelf components. It is optimized to simulate 64^3 Ising spin systems. Reddaway, Scott, and Smith [8] compare the performance of a 64 x 64 DAP to a CYBER 205 on Ising model simulations. Depending on the size of the lattice, the DAP produces far different results. A 128x128x144 lattice produces the best results, which degrades a bit for a 128^3 lattice. A 64^3 lattice does not map efficiently into the DAP storage, resulting in a drastic drop of performance to only 42.6 million spin updates per second. Monaghan, O'Brien, and Noakes [6] have built a low-cost reconfigurable processor specialized on Ising model simulations. It is based on a few Xilinx FPGA devices, hosted in an IBM-compatible PC.

We have implemented the same heatbath algorithm as on the MoM-3 on a modern Sparcstation 10/51 running at 50 MHz, with 1 MByte SuperCache and 192 MByte main memory. For a fair comparison, we have implemented different versions on the Sparcstation. The first uses the

same packed memory storage as in the MoM, resulting in many bit manipulations to extract the spin variables. The second stores each spin variable in a single byte, resulting in more, but simpler accesses to the spin variables. It turns out, that the latter is better suited to the Sparcstation, producing higher performance. The number of Ising spin updates in millions per second for all machines are shown in table 1.

machine	10^6 spin updates / second
Pearson et al., special purpose processor	25
64 x 64 DAP, depending on lattice size	42.6 … 218
CYBER 205	22
Monaghan, FPGA-based processor	4
Sparc 10/51, 50 MHz	1.74
MoM-3, 33 MHz, 2 bit serial links	93.3
MoM-3, 50 MHz, 4 bit serial links	144

Table 1. Ising system simulations: performance comparison

Despite of the moderate technology of the MoM-3 (only standard cell implementation, microprogrammed sequential operation in DPUs), the MoM-3 architecture improves performance by a factor of 83, compared to a state-of-the-art workstation. The MoM-3 is not restricted to Ising spin systems, of course. For example, two-dimensional FIR filters [5] and a JPEG compression algorithm [3] have been implemented, resulting in speed-up factors ranging from 7 to 82 compared to a Sparcstation.

6 Conclusions

The MoM-3, a configurable accelerator has been presented, which provides acceleration factors in the range of 7 to 83 compared to a state-of-the-art workstation. The custom designed rDPA circuit provides a coarse grain field-programmable hardware, especially suited for 32-bit datapaths and pipelined arithmetic. The address generation under hardware control leaves all memory accesses free for data transfers, providing a better usage of the available memory bandwidth. The loose coupling of the data sequencing paradigm allows to fully exploit the benefits of reconfigurable computing devices.

The MoM-3 prototype is currently being built as a co-processor to a VMEbus-based ELTEC workstation. All integrated circuits are based on 1.0 m CMOS standard cells, a technology made available by the EUROCHIP project of the CEC. The MoM-3 performance figures were obtained from simulations of the completed circuit designs, which were integrated into a simulation environment, describing the whole MoM-3 at functional level. The C compiler and its development environment are implemented in C programming language, running under SunOS on Sparcstations.

References

[1] K. Binder (ed.), *Applications of the Monte Carlo Methods in Statistical Physics*, Springer-Verlag, Berlin, 1984

[2] Barry, A. Cipra, An Introduction to the Ising Model, *American Mathematical Monthly*, Vol. 94, pp. 937 - 959, 1987

[3] R. W. Hartenstein, J. Becker, R. Kress, H. Reinig, K. Schmidt, A Reconfigurable Machine for Applications in Image and Video Compression, *European Symposium on Advanced Services and Networks / Conference on Compression Techniques and Standards for Image and Video Communications*, Amsterdam, March 1995

[4] R. W. Hartenstein, A. G. Hirschbiel, K. Schmidt, M. Weber, A Novel Paradigm of Parallel Computation and its Use to Implement Simple High-Performance Hardware, *Future Generation Systems 7* (1991/92), Elsevier Science Publishers, North-Holland, pp. 181-198, 1992

[5] R.W. Hartenstein, R. Kress, H. Reinig, A Reconfigurable Accelerator for 32-Bit Arithmetic, *2nd Workshop on Reconfigurable Architectures at IPPS'95*, publicly available via ftp: pub/reconfig/RAW95@sage.newcastle.edu.au, 1995

[6] Sean Monaghan, Tom O'Brien, Peter Noakes, Use of FPGAs in Computational Physics; in Will Moore, Wayne Luk (eds.): *FPGAs, Oxford 1991 Int'l Workshop on Field Programmable Logic and Applications*, Abingdon EE&CS Books, pp. 361 - 372, 1991

[7] Robert B. Pearson, John L. Richardson, Doug Toussaint, A Fast Processor for Monte-Carlo Simulation, *Journal of Computational Physics*, Vol. 51, Academic Press, pp. 241 - 249, 1983

[8] S.F. Reddaway, D.M. Scott, K.A. Smith, A Very High Speed Monte Carlo Simulation on DAP, *Computer Physics Communications*, Vol. 37, North-Holland, pp. 351 - 356, 1985

[9] K. Schmidt, *A Restructuring Compiler for Xputers*, Ph. D. Thesis, University of Kaiserslautern, 1994