ATLANTIS – A Hybrid FPGA/RISC Based Re-configurable System

O. Brosch, J. Hesser, C. Hinkelbein, K. Kornmesser, T. Kuberka, A. Kugel, R. Männer, H. Singpiel, B. Vettermann

Lehrstuhl für Informatik V, Universität Mannheim, D-68131 Mannheim, Germany
{brosch, hinkelbein, kornmesser, kuberka, kugel, maenner, singpiel}@ti.uni-mannheim.de, jhesser@rumms.uni-mannheim.de, b.vettermann@fh-mannheim.de

Abstract. ATLANTIS is the result of 8 years of experience with large stand-alone and smaller PCI based FPGA processors. Dedicated FPGA boards for computing and I/O plus a private backplane for a data rate of up to 1 GB/s support flexibility and scalability. FPGAs with more than 100k gates and 400 I/O pins per chip are used. CompactPCI provides the basic communication mechanism. Current real-time applications include pattern recognition tasks in high energy physics, 2D image processing, volume rendering, and n-body calculations in astronomy. First measurements and estimations show an acceleration up to a factor of 25 compared to a PC workstation, or commercial volume rendering hardware, respectively. Our CHDL, an object-oriented development environment is used for application programming.

1 Introduction

8 years of experience with FPGA based computing machines show that this new class of computers is an ideal concept for constructing special-purpose processors. As processing unit, I/O unit and bus system are implemented in separate modules, this kind of system provides scalability in computing power as well as I/O bandwidth.

Enable-1 [1] was the first FPGA processor developed at Mannheim University in 1994, tailored for a specific pattern recognition task. More general machines were introduced at about the same time, e.g. DecPeRLe-1 [2] or Splash-2 [3]. Enable-1 was followed by a general-purpose FPGA processor in 1996, the Enable++ [4] system. In addition to the large scale Enable++ system a small PCI based FPGA coprocessor – microEnable [5] – was developed in late 1997. It turned out that the simplicity together with the tight host-coupling of the smaller system was a significant improvement compared to Enable++.

The new FPGA processor ATLANTIS combines advantages of its predecessors Enable-1, Enable++, microEnable and others, and introduces several new features. The first is the ability to combine FPGA and RISC performance. A unique feature is the scalability and the fast data exchange between the different modules due to the CompactPCI and private bus backplane system. Another highlight is the configurable memory system which complements the flexibility of the FPGAs. We use CHDL, an unique object-oriented software tool-set that was at developed our institute, to create and simulate hybrid applications.

2 ATLANTIS System Architecture

A well-tried means to adjust a hybrid system to different applications is modularity. ATLANTIS implements modularity on different levels. First of all there are the main entities host CPU and FPGA processor which allow to partition an application into modules tailored for either target. Next the architecture of the FPGA processor uses one board-type (ACB) to implement mainly computing tasks and another board-type (AIB) to implement mainly I/O oriented tasks. A CompactPCI based backplane (AAB) as interconnect system provides scalability and supports an arbitrary mix of the two board-types, thus providing a high-speed interconnect. Finally modularity is used on the sub-board level by allowing different memory types or different I/O interfaces per board type.

Only FPGA devices with a high I/O pin-count and a complexity in the 100k gate-range are of interest for the ATLANTIS project. Two additional features are important either for our concept or for some applications: support for read-back/test and asynchronous dual ported memory (DP-RAM). In particular the partial reconfiguration is of great interest for co-processing applications involving hardware task switches. These features and a relatively low price guided the decision to use the Lucent ORCA 3T125 in the ATLANTIS system. The latest Xilinx family – the VIRTEX series – is also a good choice but was not available on the market at the time the ACB was designed. However, the AIB carries two VIRTEX XCV600 chips.

The ACB and the AIB both use a PLX9080 as PCI interface. This chip is compatible to the one used with the microEnable FPGA coprocessor. Furthermore the entire on-board support logic – like FPGA configuration and clock control – which is implemented in a large CPLD, is derived from microEnable. This high degree of compatibility ensures that virtually all basic software (WinNT driver, test tools, etc.) are immediately available for ATLANTIS.

Clock generation and distribution is an important issue for large FPGA processors. The basic approach in Atlantis is to provide a central clock from the AAB. Additionally the I/O ports of all FPGAs on either ACB and AIB have their individual clock sources. Finally each ACB and AIB provides a local clock which can be used if the main AAB clock is not available or if the application requires an additional clock. All clocks are programmable in the range of a few MHz up to at least 80 MHz. Programming is done under software control from the CPU module.

2.1 ATLANTIS Computing Board (ACB)

The core of the main processing unit of the ATLANTIS system consists of a 2*2 FPGA matrix. Assuming an average gate count of approximately 186k per chip for the ORCA 3T125 sums up to 744k FPGA gates. Each FPGA has 4 different ports:

- 2 ports @ 72 lines to a neighboring FPGA each in vertical and horizontal direction
- 1 logical I/O port @ 72 lines and

- 1 memory interconnect port @ 206 lines.

Theses 4 ports use a total amount of 422 I/O signals per FPGA. The 72 lines of FPGA interconnect provide for high bandwidth as well as multi-channel communication between chips. The memory interconnect port is built from 2 high-density 124 pin mezzanine connectors per FPGA. Depending on the application, memory modules with different architectures can be used to optimize system performance. E.g. the HEP TRT trigger (see below) will employ memory modules organized as a single bank of 512k * 176 bit of synchronous SRAM per module, leading to a total of 44 MB per ACB. The 3D-rendering algorithm will use a single module of triple width with 512 MB of SDRAM organized in 8 simultaneously accessible banks. A more generalized module – also used for 2D image processing – will take 9 MB of synchronous SRAM organized in 2 banks of 512k * 72 bits.

The I/O port serves different tasks on the 4 FPGAs, depending on the physical connection of the respective chip:

- One FPGA is connected to the PLX9080 PCI interface chip thus providing the host-I/O functionality.
- Two FPGAs are connected to the private backplane bus.
- One FPGA is attached to two parallel LVDS connectors for external I/O.

The connectors can be used to attach I/O modules, e.g. S-Link[1], to set up a down-scaled or test system without the need to add AAB and AIB modules. The 2 back-plane ports support high-speed I/O of 1 GB/s @ 66 MHz, 2*64 bits. The host-interface via PCI is compatible to the one used with microEnable, allowing 125 MB/s max. data rate.

2.2 ATLANTIS I/O Board (AIB)

The task of the ATLANTIS I/O units is to connect the ATLANTIS system to its real-world environments via the private backplane bus. To provide a maximum flexibility in connecting to external data sources or destinations a modular design of the I/O boards was selected. Depending on the standard CompactPCI card size every AIB is able to carry up to four mezzanine I/O daughter-boards.

Two Xilinx VIRTEX XCV600 FPGAs control the four I/O ports. Interfacing to the AAB and to the local PCI bridge is done in the same fashion as on the ACB. The default capacity of any of the four channels is 32 + 4 data bits @ 66MHz (or 264 MB/s ignoring the 4 extra bits). Thus the four I/O channels provide the same band-width as the 2 backplane ports: 1GB/s. To provide a sustained and high I/O bandwidth even at small block sizes buffering of data can be done in two stages (numbers per I/O channel):

- A 32k * 36 FIFO-style buffer connected directly to the I/O port, implemented with dual-ported memory.
- A 1M * 36 general purpose buffer implemented with synchronous SRAM.

The fact that both FPGAs are connected to the PLX local bus provides a communica-tion means in case channel synchronization, loop-back or the like is needed.

[1] S-Link is a FIFO-like CERN internal standard for point-to-point links.

2.3 ATLANTIS Active Backplane (AAB)

ACBs and AIBs share the same I/O-circuit with 160 signal lines. Connections between boards are done using the private bus system of the AAB. The default configuration of the I/O lines will be 4 channels of 32bit plus control, however any granularity from 16 channels of a single byte to 2 channels of 64 bit might be useful.

Different backplanes can be used in order to scale the ATLANTIS system to the respective application. A simple pipelined, passive, i.e. not configurable, backplane is currently used for system and performance tests.

The total bandwidth is 1 GB/s per slot. For example configuring the backplane for two independent pairs of ACBs and AIBs, an integrated bandwidth of 2 GB/s will result for a single ATLANTIS system. Like all other boards, the backplane is controlled by the host CPU via the PCI bus.

2.4 Host CPU

The host computer to be used with ATLANTIS is an industrial version of a standard x86 PC – a CompactPCI computer – that plugs into one of the AAB slots. This industrial computer is equipped with a mobile Intel Pentium-200 MMX or Celeron-450 processor and thus 100% compatible to a standard PC desktop workstation. All standard operating systems can be used, in particular Windows NT and Linux, without the need to adapt drivers or I/O handlers, etc. The compatibility at the device driver level of ATLANTIS with the small scale FPGA processor microEnable allows a quick start using the tools already available.

The CPU module allows to have the complete FPGA development tool-set be run on the target system, as well as the application itself. The ACB and AIB boards act as coprocessors, accelerating time and resource consuming parts of an application, and providing high I/O bandwidth. Moreover, the CPU is needed for control, when task switching and re-configuration of FPGAs is desired. Additionally, high precision floating point operations that are too much resource consuming on FPGAs, may be carried out in the CPU.

2.5 CHDL Development Environment

CHDL (C++ based Hardware Description Language) was designed to support simulation of FPGA coprocessors. The use of commercial VHDL products to simulate FPGA coprocessors shows some insufficiencies:

1. A test bench must be implemented for emulating the FPGA environment using VHDL while the application operating the FPGA is mostly written in C/C++.
2. The test bench has to emulate the behavior of the microprocessor system exactly, including bus system and DMA controllers at the level of bus signals.
3. Implementing the test bench is redundant work because the application already contains the whole algorithm needed for simulation.

CHDL provides a hardware description based on C++ classes for entering structural designs and state machine definitions. A CHDL design description is a traditional C++ program linked to a class library. This enables the developer to implement complex high level software which generates the structural CHDL design automatically.

The developer uses the original application to simulate the designs. No traditional hardware oriented test benches are needed. One single language, C++, is sufficient to manage the whole development process. In both the application and the hardware description the features of this powerful programming language can be used.

More details can be found in [6].

3 Applications

FPGA processors have shown to provide superior performance in a broad range of fields, like encryption, DNA sequencing, image processing, rapid prototyping etc. Very good surveys can be found in [3] and [7]. We are in particular interested in hybrid CPU/FPGA systems for:

- acceleration of computing intensive pattern recognition tasks in High Energy Physics (HEP) and Heavy Ion Physics,
- subsystems for high-speed and high-frequency I/O in HEP,
- 2-dimensional industrial image processing,
- 3-dimensional medical image visualization and
- acceleration of multi-particle interaction (e.g. N-Body [8], SPH) calculations in astronomy.

3.1 High Energy Physics

In the field of HEP many FPGA algorithms have been implemented at our institute during the past 5 years. Results show speedup rates in the range from 10 to 1,000[2] compared to workstation implementations [9]. The most recent HEP pattern matching algorithm tries to find straight or curved tracks in a 2-dimensional input image delivered by a transition radiation tracking detector (TRT) with a repetition rate of up to 100 kHz. The size of the detector image is 80,000 pixels. The number of patterns varies from 240 to more than 2,400 depending on the operating frequency. The working principle of the algorithm is as follows:

- Predefined patterns are stored in a large look-up table (LUT) with every data bit representing one pattern.
- Each pixel in the input image contributes to a number of patterns, defined by the content of the LUT.
- For every pattern a counter increments if its corresponding data bit is set. The total of all counter values builds the track histogram.
- A track is considered valid if its value is above a predefined threshold.

A description of the algorithm and its implementation can be found in [10].

In particular this algorithm is ideally suited for an FPGA implementation because it can be extremely parallelized. Adjustable memory boards allow RAM access with a width of e.g. 4*176 bits. Therefore, 706 straws can be processed simultaneously on a single ACB board equipped with 4 memory modules, thus providing an enormous speed-up compared to other systems, e.g. a state-of-the-art PC.

[2] Measured on Enable-1 with parallel histogramming only, no I/O was needed.

3.2 Image processing

Almost all image processing applications involve tasks where image elements (pixels or voxels) have to be processed with local filters. Among others, hardware implementation of algorithmically optimized real-time volume rendering is a current project at our institute in this area.

The following rendering - or ray processing - pipeline is assumed:

- Starting from each pixel of the resulting image rays are cast into the virtual scene.
- At equally distant positions on the rays sample points are generated by tri-linear interpolation of the neighboring voxel values.
- Sample points are classified with opacity or reflectivity according to gray values and gradient magnitude.
- Finally, the absorption for each voxel is determined. The reflected fraction of the light intensity reaching the sample point is calculated and added to the contributions of all other sample points on that ray.

The new architecture uses algorithmic optimizations: regions with no contribution are skipped, and processing is aborted as soon as the remaining intensity drops under an adjustable threshold. To overcome the resulting data and branch hazards in the rendering pipeline multi-threading is introduced. Each ray is considered as a single thread, and after each sample point the context is switched to the next ray. Our implementation has the same speed-up like software implementations of this algorithm, compared to volume rendering without algorithmic optimizations. However, compared to conventional architectures the number of pipeline stalls is reduced from more than 90% to less than 10% of rendering time.

Details of the algorithm and its FPGA implementations can be found in [11].

3.3 Astronomy

Using FPGAs to accelerate complex computations using floating-point algorithms has not been considered a promising enterprise in the past few years. The reason is that general floating-point [12] as well as particular N-Body [13] implementations have shown only poor performance[3] on FPGAs.

Usually N-Body calculations need a computing performance in at least Tera-FLOP range and are accelerated with the help of ASIC based coprocessors [14]. Nonetheless we have recently investigated the performance of a certain sub-task of the N-Body algorithm on the Enable++ system [15]. The results indicate that FPGAs can indeed provide a significant performance increase even in this area.

3.4 Measured and Estimated Performance

HEP. Besides principle parameters like system frequency the DMA performance plays a dominant role for the execution time of the TRT algorithm. Therefore DMA Read/Write access was the main focus of the measurements. Following are some

[3] In 1995 approx. 10 MFLOP per Xilinx chip were reported for 18 bit precision, and 40 MFLOP with 32 bit precision on an 8 chip Altera board.

results showing the data throughput over CPCI for various applications, measured with ATLANTIS, microEnable driver, design speed 40 MHz.

Table 1. ATLANTIS DMA performance

Block size (kByte)	1	4	32	256
DMA Read perf. (MB/s)	8.8	24.6	75.3	97.7
DMA Write perf. (MB/s)	7.4	21.6	54.3	65.3

The effect these results suggest for the performance of a distributed system largely depend on the respective application. For the TRT algorithm, the time needed for I/O is indeed the bottle-neck, in case the ATLANTIS sub-systems are employed as co-processors and thus receive their data from the host CPU.

Measurements of histogramming performance were done using a single-memory ACB (176 bit RAM access) [16]. The execution time on the test system (algorithm plus I/O), 19.2 ms compared to 35 ms using a C++ implementation on a Pentium-II/300 standard PC, extrapolates to 2.7 ms using 2 ACB with 4 memory modules each (1408 bit RAM access). This corresponds to a speed-up by a factor of 13.5.

Volume Rendering. The hardware speed is limited by several factors. One is the memory bandwidth. Assuming 100 MHz devices, simulations have shown that 4 Hz frame rates for 1024^3 data sets can be achieved for typical data with hard surfaces and otherwise empty space in between [17]. With our FPGA solution we will achieve a clock rate of >25 MHz that reduces the frame rate accordingly.

For detailed simulation we used a CT data set with 256*256*128 voxels. This data set is viewed from three different viewing directions and three different levels of opacity for soft tissue is applied.

On average one achieves efficiencies of between 90% and 97%. The number of sample points varies between 10-15% of all voxels if the data set consists mainly of empty space and opaque objects and 25-40% for semi transparent opacity levels.

The above results correspond to rendering rates from 20 Hz on semi-transparent data sets to 138 Hz for opaque objects and parallel projection. The results are achieved from images of size 256*128. Perspective views reduce the rendering speed by a factor of about 2.

Comparing these results with the performance of the only commercially available volume rendering hardware, VolumePro [18], simulations suggest a speed-up by a factor of 10 to 25 when using 1024^3 data sets.

4 Summary and Outlook

ATLANTIS is a CompactPCI based computing machine that combines the advantages of FPGA and RISC architectures. Its unique features are scalability, flexibility with respect to memory, configurable high-speed I/O, and it comes with a powerful object-oriented development environment, CHDL.

ATLANTIS has proven its supreme power regarding bandwidth and speed in applications we have investigated so far. An ACB is available since 09/1999 and is currently tested with different memory modules and a simple backplane, with different applications. A second ACB and an AIB will be completed shortly. Though the full system is not available by now (01/2000) it is planned to have an implementation of a HEP trigger application run in a real experiment (FOPI at GSI, Darmstadt, Germany) within this year. Other implementations concern future experiments, or have prototype character.

References

[1] Klefenz F., Zoz R., Noffz K.-H., Männer R., “The ENABLE Machine - A Systolic Second Level Trigger Processor for Track Finding”, Proc. Comp. in High Energy Physics, Annecy, France; CERN Rep. 92-07 (1992) 799-802

[2] DECPeRLe-1, an FPGA processor containing 16 Xilinx XC3090 FPGAs, http://pam.devinci.fr/hardware.html#DECPeRLe-1

[3] D. Buell, J. Arnold, W. Kleinfelder, “Splash-2 – FPGAs in a Custom Computing Machine“, CS Press, Los Alamitos, CA, 1996

[4] H. Hoegl et al., “Enable++: A Second Generation FPGA Processor”, Proc. IEEE Symposium on FPGAs for Custom Computing Machines, pp. 45-53, 1995

[5] microEnable, a PCI based FPGA co-processor by Silicon Software GmbH, http://www.silicon-software.com/

[6] K. Kornmesser et al, “Simulating FPGA-Coprocessors Using the FPGA Development System CHDL”, Proc. PACT Workshop on Reconf. Comp., Paris (1998) pp. 78-82

[7] J. Vuillemin et al., “Programmable Active Memories: Reconfigurable Systems Come of Age”, Proc. of the 1996 IEEE Trans. On VLSI Systems

[8] R. Spurzem, S.J. Aarseth, “Direct Collisional Simulation of 10,000 Particles Past Core Collapse", Monthly Notices Royal Astron. Soc., Vol. 282, 1996, p. 19

[9] V. Dörsing et al., “Demonstrator Results Architecture – A”, ATL-DAQ-98-084, CERN, 26 Mar 1998

[10] A. Kugel et al., “50kHz Pattern Recognition on the Large FPGA Processor Enable++”, Proc. IEEE Symp. on FPGAs for Custom Computing Machines, CS Press, Los Alamitos, CA, 1998, pp. 1262-3

[11] J. Hesser, B. Vettermann, “Solving the Hazard Problem for Algorithmically Optimized Real-Time Volume Rendering”, Int. Workshop on Vol. Graph. 1999, Swansea, UK

[12] W. Ligon et al, “A Re-evaluation of the Practicality of Floating-Point Operations on FPGAs”, Proc. IEEE Symp. on FPGAs for Custom Computing Machines, 1998

[13] H.-R. Kim et al, “Hardware Acceleration of N-Body Simulations for Galactic Dynamics”, SPIE Conf. on FPGAs for Fast Board Develop. and Reconf. Comp. 1995, pp. 115-126

[14] J. Makino et al, “GRAPE-4: A Massively Parallel Special-Purpose Computer for Collisional N-Body Simulations”, Astrophysical Journal, Vol. 480, 1997, p. 432

[15] T. Kuberka, Diploma Thesis, Universität Mannheim, Germany, 1999

[16] C. Hinkelbein et al, “LVL2 Full TRT Scan FEX Algorithm for B-Physics Performed on the FPGA Processor ATLANTIS”, to be publ. as ATL-DAQ-Note, CERN

[17] B. Vettermann et al, “Implementation of Algorithmically Optimized Volume Rendering on FPGA Hardware”, IEEE Visualization '99, San Francisco, CA (1999)

[18] VolumePro, a PCI based volume rendering coprocessor by Mitsubishi Electronics America, Inc. RTVIZ, http://www.rtviz.com/