# Hardware Acceleration of Graphics and Imaging Algorithms Using FPGAs

Pavel Zemcik[1]
[1]*Brno University of Technology*

## Abstract

Computer graphics algorithms and algorithms used in image processing are generally computationally expensive. This fact is the reason why people struggle to accelerate such algorithms using any reasonable means. The traditional sources of speedup are faster processors, parallelism, or dedicated hardware. Development in digital circuit technology, especially rapid development of Field Programmable Gate Arrays (FPGA), offers alternative way to acceleration. Current FPGA chips are capable of running graphics algorithms at the speed comparable to dedicated graphics chips. At the same time they are configurable not only using schematics diagram but also through high-level programming languages, e.g. VHDL. The contribution addresses these issues, general development in the area, and shows examples of hardware platforms and algorithms that can be implemented on such platforms.

**Keywords:** computer graphics, image processing, FPGA, hardware acceleration

## 1 Introduction

Computer graphics and image processing are traditionally the areas with very high demands for computational power. Such demand is generally a result of a simple fact that the algorithms that are connected with images mostly work with large data sets and the requested processing time is in many cases quite short.

Applications areas that require short processing time for large image data sets are e.g. all virtual reality applications, computer games, biomedicine, scientific visualization, numerical simulation, visual quality inspection, and many others [1][2].

Historically, the computer graphics systems were at first built as pure hardware systems. As the computer technology was making progress, the lower end graphics systems were gradually changed so that the hardware was minimised while the tasks that the graphics system was performing were moved to the general-purpose processors. At the same time the higher end systems were built with single purpose “graphics accelerators”.

Today the graphics accelerators – single purpose engines capable of rendering triangles/polygons at very high speed – are present even in the lowest end desktop and notebook personal computers (PC) and they are starting to be present in pocket PCs and also in the portable devices, such as mobile telephones. The high performance systems that are being built today achieve the high performance using parallelism in addition to the speedup achieved by better architectures of the graphics subsystems and clock speedup.

While it is nice that the application developers can rely on the presence of accelerated graphics engines in the computers, it is quite unfortunate from the point of view of graphics and imaging algorithms research that the function of the graphics accelerators is usually quite strictly limited to rendering of planar triangles/polygons and limited choice of shading and texture algorithms and it is usually impossible to use them for implementation of any other algorithms. At the same time the real research of such high-performance graphics subsystems is being done by the manufacturers of the graphics subsystems and by the affiliated institutions, such as research laboratories and universities.

The above mentioned status of the computer graphics development is unpleasant specifically from the view point of the East European universities and also European universities and research institutions in general as the European region is not quite rich in companies designing or manufacturing graphics hardware but it traditionally does have strong research.

A reasonable way forward was offered by the recent development of Field Programmable Gate Arrays (FPGAs). Current technological progress [3][4][5] allows implementation of even very complex devices in the programmable logic devices and achieve good results even with architectures and algorithms that are not supported by the traditional computer graphics manufacturers.

In many cases it is reasonable to combine the FPGA design with a unit with a controller that is capable of executing non-critical but algorithmically complex tasks. Digital signal processors (DSPs) are suitable devices for this task [6][7].

## 2 Methods

The architectures of the hardware accelerated computer graphics applications can be seen from several points of view, e.g.:

- Algorithm implementation
- Data distribution
- Load distribution

Various experiments have been carried out and published by various authors with various focus on the above classification [8][9][10][11]. The load and data distribution issues are common for hardware acceleration units and general parallel processing while the algorithm implementation point of view is unique for the hardware acceleration units.

---

[1] zemcik@fit.vutbr.cz

The algorithms that cannot benefit from the “traditional” graphics acceleration engines are those that would benefit the most from other forms of hardware acceleration. The most frequently mentioned ones are:

- Volume rendering. The well known application for volume rendering is visualisation of medical data (CT/NMR) data or other 3D data obtained through 3D scanning. Currently, the medical data is visualised directly through ray casting of the volume (represented usually by raster or octree) or converted in surface representation (e.g. using marching cubes algorithm) and then displayed through the available graphics hardware. The direct method is slow in rendering, the conversion is slow itself – volume rendering generally remains difficult.
- Ray tracing. Ray tracing is usually not used in interactive applications but rather for pre-processing, video/film production, high-quality presentation rendering, etc. Regardless of the scene representation, ray tracing is quite demanding method and only few possibilities to accelerate it exist, the principal being subdivision of graphics scenes into subspaces and then tracing the subdivided scene. Ray tracing is very similar in implementation to particle tracing, or backward ray tracing – the methods that are even more computationally demanding but have better rendering features.

# 3 Technology

FPGAs have been designed as one of the types of the configurable logic devices suitable for the most complex designs. The structure of the FPGA consists of two maor parts. The Configurable logic blocks (CLBs) and the Programmable Switching Matrix (PSM). See Figures 3.1. and 3.2 for the schematics of the FPGA structure.

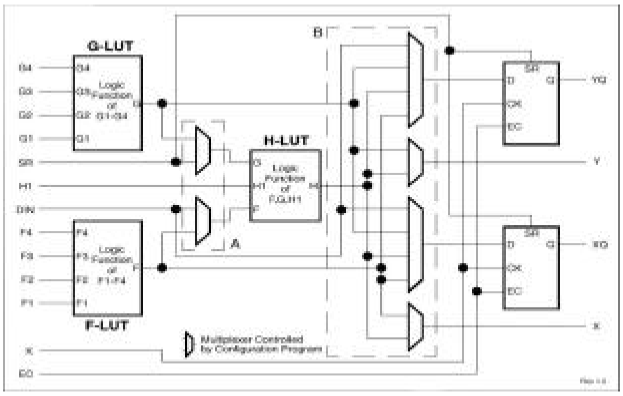

Figure 3.1: FPGA configurable logic blocks

Since FPGAs (Field-Programmable Gate Arrays) were invented they have created many new opportunities. FPGAs capable of being quickly configured at run time have significant potential for improved performance and resource usage for various applications comparing to applications specific chips and also comparing to general processors. In addition, FPGAs capable of partial reconfiguration allow for the reconfiguration of a portion of the FPGA while the remainder of the application is running. Partial reconfiguration is the ability of certain FPGAs to reconfigure only selected portions of their programmable hardware while other portions continue to operate undisturbed.

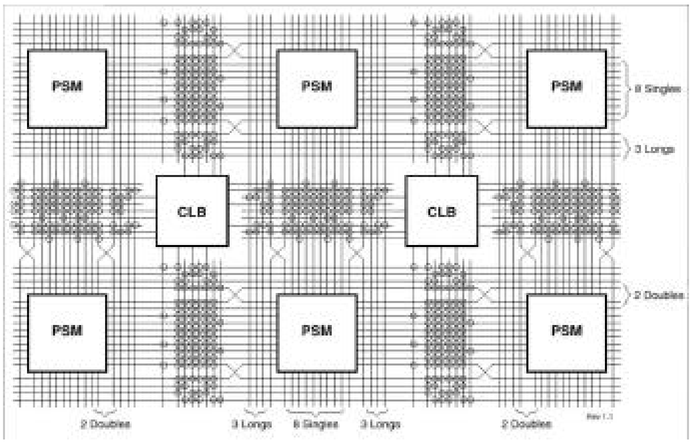

Figure 3.2: FPGA overall structure

The authors of this project consider Xilinx Virtex [4][5] the best currently available FPGAs on the market from the view point of suitability for computer graphics applications. Their architecture is similar to the Spartan [3] series (previously widely used series). With densities ranging from 40,000 up to 10 million system gates, the Virtex series solution delivers enhanced system memory and ultra-fast DSP through a flexible IP fabric. Additionally, significant new capabilities address system-level design issues including flexible system interfaces with complex system clock management.

Xilinx Virtex FPGAs fully addresses all aspects of system connectivity in high-performance designs. System connectivity consists of the physical interface and the protocols required to offer higher bandwidth. The FPGA technology provides the fastest and most flexible electrical interfaces available. Each user I/O pin is individually programmable for any of the 19 single-ended I/O standards or six differential I/O standards. This technology delivers 840 Mbps LVDS performance using dedicated Double Data Rate (DDR) registers.

The new series of Virtex devices contain up to 12 Digital Clock Managers (DCMs) Each DCM provides phase shifting and frequency synthesis capabilities, which are ideally suited for systems with multiple clock domains and critical timing requirements. The DCM delivers unsurpassed flexibility for managing both on-chip and off-chip clock synchronisation. Each Virtex device has 16 pre-engineered clock domains to support the multiple frequency and multiple phase requirements of complex system designs. Each built-in, low-skew clock network eliminates complex clock tree analysis and simplifies the system design process.

The high density on-chip memory in the Virtex solution increases overall system bandwidth by providing fast and resource efficient FIFO buffers, shift registers, and CAM. The distributed RAM, block RAM, and high-speed memory interfaces, provide a powerful memory-based data-path solutions for bandwidth intensive systems. The Virtex solution provides industry's highest memory to logic ratio with up to 3.5Mb of on-chip block RAM and delivers over 400Mbps DDR/QDR external memory interface performance.

The Virtex chips can deliver over 600 Billion MACs/s of performance. Up to 192 18 x 18 multipliers in a single device can be implemented. The multipliers can be fully combinatorial running between 140 and 250 MHz depending on bit width. Designers can use Virtex devices to implement computationally critical digital system elements such as sub-1-microsecond 1024 point FFTs, ultra-fast filters, etc. The Virtex series FPGAs are also accompanied by a suite of sophisticated design and simulation tools that support the industry's fastest runtimes and the most advanced design methodologies.

## 4 Experiments

At Faculty of Information Technology, Brno University of Technology, we are carrying out experiments with hardware accelerated graphics and image processing. Our experiments include simulation of image processing algorithms, simulations of graphics algorithms, and implementations of the accelerated algorithms. We have so far implemented two designs and one new design is under development. All of the designs are intended for real applications as well as research.

The general design goal of all of the systems was to develop modules that would be capable of high performance operation independently of the host PC or connected to a host PC. Important design goal was also to maximize the performance/price ratio rather than achieve the maximum possible performance at any price. Other goal was to make the modules scalable in that sense that it would be possible to interconnect them through high-capacity communication links. All of the designs do contain flash permanent memory and communication devices, that are not that important for the actual operation of the modules and therefore they are not indicated in the schematics, but they have a critical role in starting the systems.

In the first of our designs – Xilinx Spartan and Texas Instruments TMS320C32 (60MIPS floating point) DSP – we were testing the basic ideas of the FPGA and DSP co-operation. The FPGA in this design served as input/output module for the DSP and also as a processing unit for the bulk data. Due to limitations in computational speed and memory, the module was only used for image processing applications [13][14][15] and 2D and wireframe graphics. The design was finished in 1997 (see Figure 4.1).

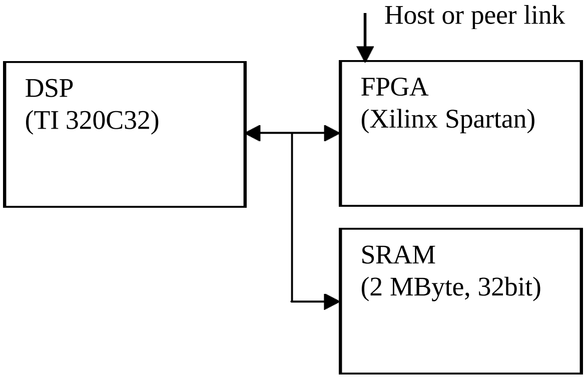

Figure 4.1: Design with C32 and Spartan FPGA

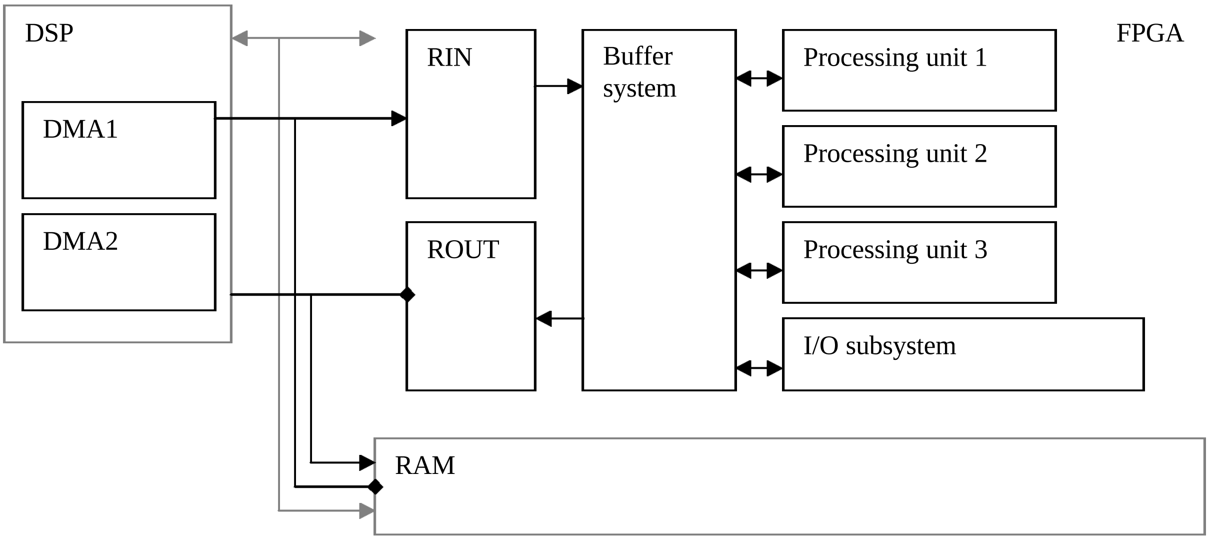

Figure 4.2: Raster processing using the DSP and FPGA

The design does have some limitations that were forced in the design because the design goal was to achieve the best possible performance/price ratio rather than the best possible performance. Specifically we decided (based on results of simulations of raster data processing) to leave the memory access purely on the DSP (its DMA controllers). This decision does not have adverse effect on performance but limits the class of algorithms that can be run efficiently to those that use the data in fixed order. The system is capable of running raster algorithms with significant speedup over the pure DSP systems. The typical way to use the system is shown in Figure 4.2.

The data is typically fetched from the memory (RAM) using the DMA1 controller built in the DSP chip and stored in the input register (RIN), then it is run through the buffer system and delivered in a processing unit. The results of the algorithms are transferred back to he output register (ROUT) and then transferred using the second controller (DMA2) back to the memory. When the algorithm is applied on all of the data, an interrupt is issued to the processor (DSP). The processor then handles further processing. The operation does not interfere with the processor function except they possibly compete for the memory bandwidth if the processor needs to access external memory (but for the optimal performance the system can be tuned so that only internal memory is used during the transfers.

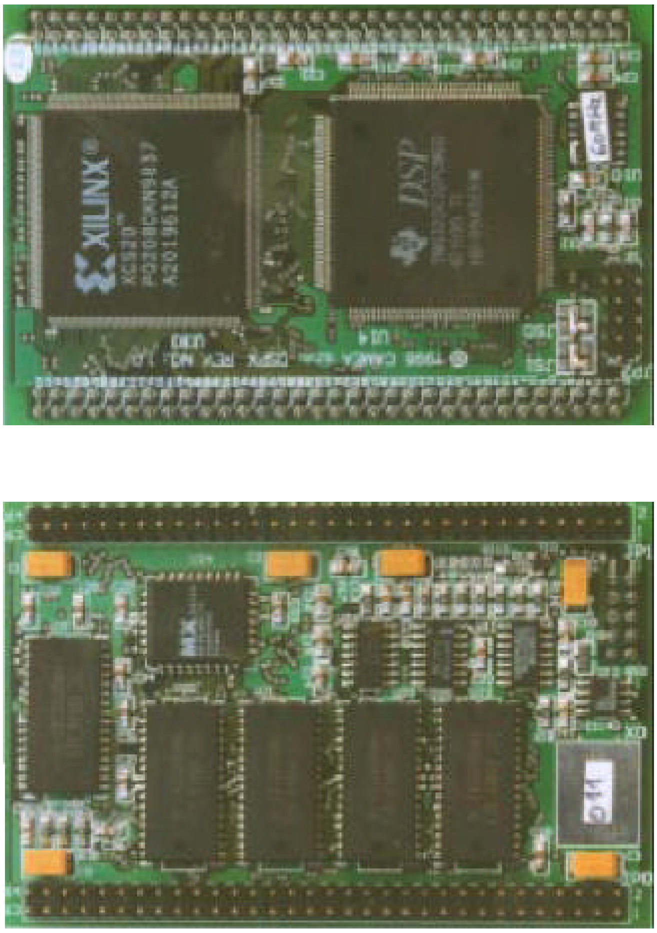

Figure 4.3: Photographs of the C32 module

The processing unit (or possibly set of units) in the FPGA implements the actual imaging algorithms, typically 3x3 convolution filter with fixed coefficients, morphological filters, single pixel functions, etc. The typical speedup of the system over the traditional implementation in the processor obviously depends on the algorithm implemented, but generally it is at least 20:1 for 3x3 window functions. The photograph of the system is shown in Figure 4.3.

An example of the processing speed of the system is the speed achieved for 3x3 convolution filtering – the system is capable of filtering around 5 milion pixels per second – the speed suitable for processing of the B/W video signal in real time.

The current generation of the designs uses the Xilinx Virtex FPGA and Texas Instruments TMS320C6711 (1000MIPS floating point) DSP. The design goal of the system was to develop a system capable of the high performance 3D graphics acceleration [16][17] and image processing [18][19][20].

The main advantages over the previous design is that the new design contains a DSP closer in performance with the PCs, significantly better FPGA, more memory, and that the FPGA can have a small local memory. The main effect of the new design on applications, however, is that the new design supports dynamic reconfiguration of the portions of the FPGA so that the FPGA can swap configurations in the real time (and thus maximize the exploitation of the FPGA circuits). The local memory connected to the FPGA allows for local storage of the intermediate results or constant tables, microcode, etc. The block diagram of the system is shown in Figure 4.4.

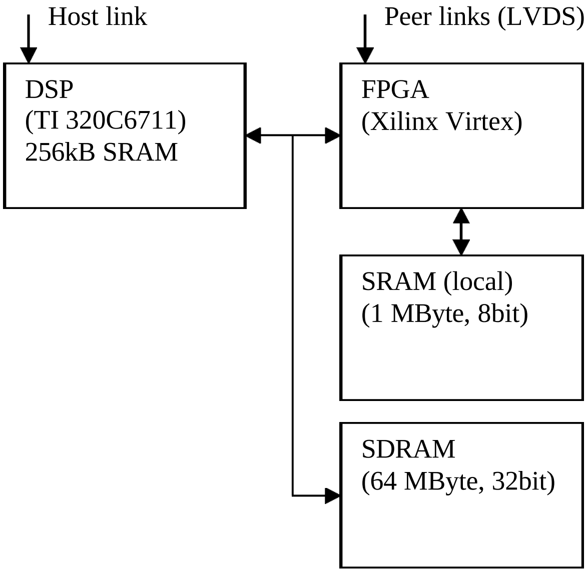

Figure 4.4: Design with C6711 and Virtex FPGA

The design with C6711 was finished in 2001 and currently we are developing the software support tools for the module. The software is based on the Texas Instruments DSP/BIOS multithreading core and channelbased communication subsystem for data exchange and synchronization between the threads and modules.

Connection of the module to the PC is planned through the PCI motherboard (currently being manufactured) that will also contain bulk memory shared by the Xilinx Virtex-E FPGA and the DSP module. The board provides a fast interface between the PC and one or more DSP modules connected to the board directly or through the LVDS links. The memory is intended primarily as the data buffer (see Figure 4.5).

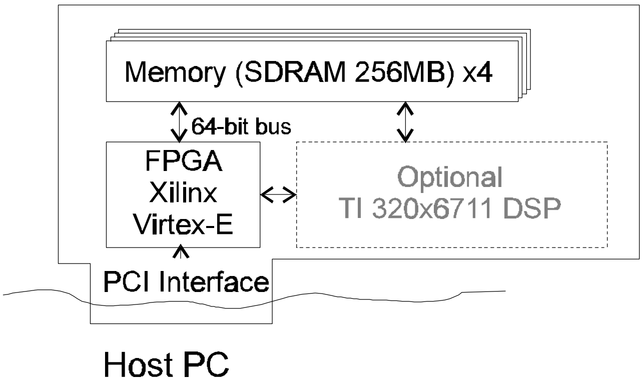

Figure 4.5: PCI motherboard design

The raster processing principle described earlier can also be applied on this design. The changes in the design, however, have significant impact on the performance and efficiency for several reasons:

- The DSP has much more internal memory so the DSP and DMA are less likely to compete for the memory.
- The processing units are reconfigurable, so the FPGA can be used more efficiently and more specialised functions can be afforded.
- The DSP has more DMA controllers, the memory bandwidth is faster, and supports multithreading so the operation of the FPGA processing units can be overlapped.

The system, however, can be used also in a more general way as the limitation of memory access done only through the DSP does not apply and the FPGA can fully address the memory. We have experimented with simulation/implementation of several algorithms. Generally the results were good and showed that the performance of the new module is better than the old one, as expected; however, some bottlenecks connected specifically with the memory bandwidth were found in algorithms that work with 3D volume data and lead in a new design that is being prepared.

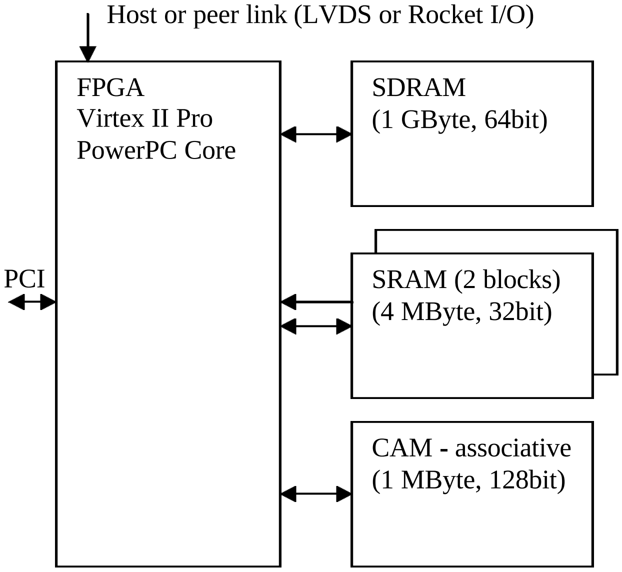

Figure 4.6: Future design - Virtex II Pro FPGA and CAM

The prepared new design that is now under development is specifically being prepared to overcome bottlenecks specifically in 3D volume data rendering.

Such bottlenecks were found in two cases:

- In volume rendering based on object order the bottleneck was found in image retrieval and storage – in such methods the temporary image frame must be retrieved and stored at least once per voxel slice and can be solved by adding two image buffers in the design.
- In ray tracing or ray casting the bottleneck was in searching for the rays whose processing was postponed for various reasons – the problem can be overcome by adding a small associative memory in the design.

An example of object order based volume rendering is shown in the figure 4.7. The volume dat is fetched from the SDRAM and put in the gradient/normal vector estimator along with the buffered content of the previous volume data slices (stored partially in the on-FPGA buffers and partially in the frame buffers). After the gradient is estimated, it is decided whether the part of the volume is visible or not. If it is visible, it must be shaded using the shading unit. In parallel, the previous temporary image frame is fetched from the frame buffer and merged with the results of the shading unit. After a slice of data is processed, the frame buffers are swapped. Finally, the results are passed to the host PC through the PCI interface.

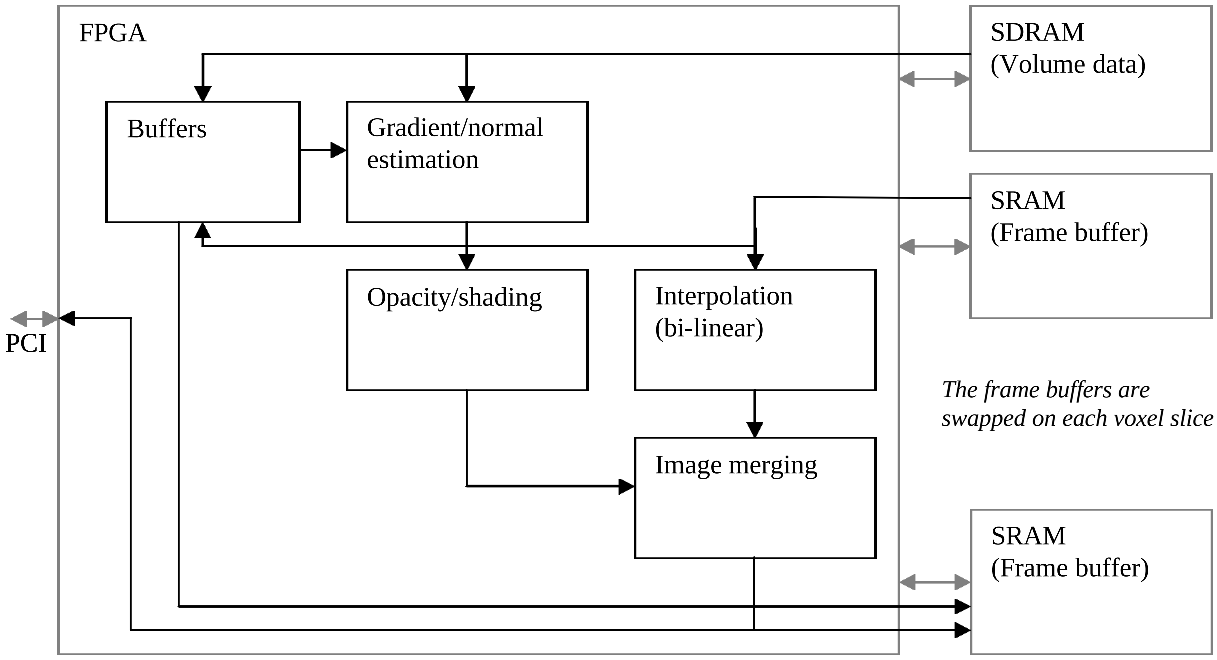
FPGA
Buffers
Gradient/normal estimation
Opacity/shading
Interpolation (bi-linear)
Image merging
PCI
SDRAM (Volume data)
SRAM (Frame buffer)
SRAM (Frame buffer)
The frame buffers are swapped on each voxel slice

Figure 4.7: Object order based volume rendering

The ray tracing/ray casting implementation in the new design is shown in the figure 4.8. The primary rays are generated in the ray generation engine (Implemented in the Power PC core) and processed by a set of 3D line generation engines combined with the gradient estimators. (So far, the design is only simulated, but it is estimated, that 16 such engines will fit in the FPGA used on the board). These engines fetch data from the voxel cache implemented on-chip in the FPGA. As the rays are shot in a very similar direction, a high probability exists that all the ray engines will need very similar data. When an object hit/suitable gradient is found, the data is passed to the pixel shader unit and then stored in the frame buffer. In some cases, the assumption that the volume memory content needed by the rays is similar fails. In such cases, the rays that violate the assumption are not processed but they are put in the ray cache for further processing, on the other hand, the ray cache is constantly searched for the rays that require the content of the memory that is currently present in the voxel cache and if a hit is found, the rays are fetched back in the engines.

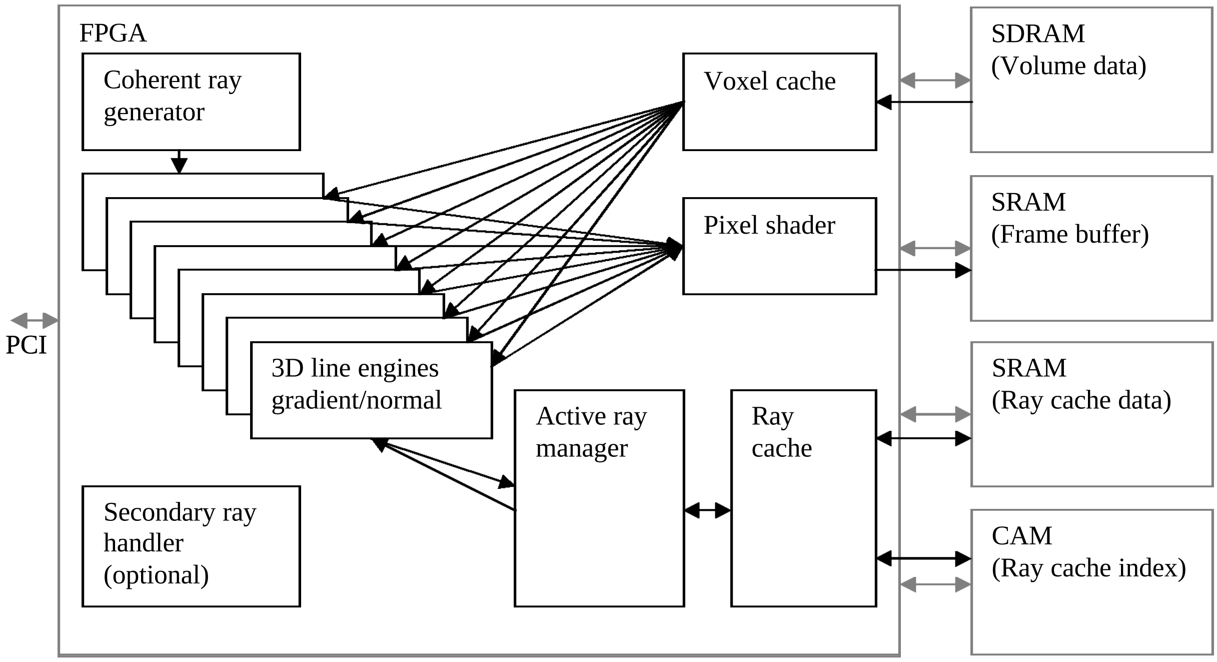

Figure 4.8: Ray casting/ray tracing of volume data

# 5 Conclusion

The hardware accelerated architectures for computer graphics and image processing are one of the promising approaches in computer graphics. The FPGA circuits offer a reasonable way to allow experiments to the people who are not connected to the chip manufacturers.

The experiments we have carried out show that even relatively simple designs can hep speed-up processing in computer graphics and image processing and that more complex designs can be used as co-processors for the PCs (or other graphic workstation architectures).

# Acknowledgements

I would like to thank for the support to my colleagues, especially Otto Fucik, who is responsible for the hardware design of the boards and to CAMEA (ltd.) who supplied the material for the experimental work.

I would also like to thank the organisers of SCCG 2002 for the opportunity to prepare this contribution.

The work has been supported by grant of the Grant Agency of the Czech Republic (GAÈR) GA102/02/0507 “Computer Graphics Algorithms with FPGA Support”.

# References

[1] Watt A.: 3D Computer Graphics, Addison-Wesley, Wokingham, UK, 1993

[2] B. Jahne, H. Haussecker, P. Geissler: *Handbook of Computer Vision and Applications*, Academic Press, San Diego, CA, USA, 1999

[3] *Spartan and Spartan-XL Families Field Programmable Gate Arrays*, Xilinx, DS060 (v1.6) USA, September 19, 2001, (available at http://www.xilinx.com)

[4] *Virtex™ 2.5 V Field Programmable Gate Arrays*, Xilinx, DS003-1 (v2.5), USA, April 2, 2001, (available at http://www.xilinx.com)

[5] *Virtex-II Pro™ Platform FPGAs: Introduction and Overview*, Xilinx, DS083-1 (v1.0), USA, January 31, 2002 (available at http://www.xilinx.com)

[6] *TMS320C32 Digital Signal Processor*, Texas Instruments, SPRS027C, USA, September 2001, (available at http://www.ti.com)

[7] *TMS3B0C6711, TMS320C6711B Floating point Digital Signal Processors*, Texas Instruments, SPRS088B, USA, September 2001, (available at http://www.ti.com)

[8] A, Kaufman, R. Bakalash: *Memory and Processing Architecture for 3D Voxel-based Imagery*, IEEE Computer Graphics and Applications, vol. 8, no. 6, p. 10-23, USA, Nevember 1988

[9] S. Juskiw, N. Durdle, V. Raso, D. Hill: *Interactive Rendering of Volumetric Data Sets*, Computer and Graphics, vol. 19, no. 5, p. 685-693, 1995

[10] I. Bittner, A. Kaufman: *A Ray-Slice-Sweep Volume rendering Engine*, In Proceedings of the Siggraph/Eurographics Workshop on Graphics Hardware, pages 121-138, USA, August 1997

[11] R. Harvey, H. Pfister, D. Silver, T. A. Cook: *Ray Casting Architectures for Volume Visualization*, IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 3, USA, July-September 1999

[12] F. Dachille, A. Kaufman: GI-Cube: *An Architecture for Volumetric Global Illumination and Rendering*, In Proceedings of the SIGGRAPH / Eurographics Workshop on Graphics Hardware. P. 119-128, August 2000

[13] P. Zemcik O. Fucik, J. Honec, P. Valenta: *Cost Efficient Image Digitising and Processing Using FPGAs*, In Proceedings Microcomputer'94, p. 225-229, Zakopane, Poland, 1994

[14] O. Fucik, P. Zemcik, R. M. Fors, D. Loker, R. Weissbach: *A Flexible DSP Hardware Platform for LVDT Signal Conditioning*, In Proceedings of Electromotion International Symposium, p. 8, Bologna, Italy, 2001

[15] P. Zemcik, O. Fucik, M. Richter, P. Valenta: *Imaging Algorithm Speedup Using Co-Design*, In Summaries Volume Process Control 01, p. 96-97, Strbske Pleso, Slovakia, 2001

[16] P. Zemcik: *An efficient algorithm for 3D line generation*, In Machine Graphics and Vision, vol. 2, no. 3, p. 231-235, Poland 1993

[17] P. Zemcik, A. G. Chalmers: *Optimised CSG Tree Evaluation for Space Subdivision*, Computer Graphics Forum, p. 139-146, Netherlands, 1995

[18] P. Zemcik, P. Valenta, O. Fucik, J. Honec. M. Richter: *Visual Analysis on the Production Line, In Proceedings of The 10th Scandinavian Conference on Image Analysis*, Lappeenranta, Finland, 1997

[19] A. Kaarna, P. Zemcik, H. Kalviainen, J. Parkkinen: *Compression of Multispectral Remote Sensing Images Using Clustering and Spectral Reduction*, IEEE Transactions on Geoscience and Remote Sensing, p. 1073-1082, Piscataway, NJ, USA, 2000

[20] A. Herout, P. Zemcik: *Raster Volume Data Graphics Library*, In Proceedings ASIS 2001, p. 15, Brno, MARQ, Czech Republic, 2001