AUTOMATIC GENERATION OF FPGA HARDWARE ACCELERATORS FOR GRAPHICS APPLICATIONS

Alexandru Amaricai and Oana Boncalo
Department of Computer Engineering, University Politehnica of Timisoara, Timisoara, Romania

Keywords: FPGA Graphics, FPGA Acceleration, OpenGL.

Abstract: In recent years, FPGAs have been increasingly utilized in implementing graphic engines in application fields such as automotive, avionics or military. However, due to their high flexibility, designs for FPGA devices can target a very limited range of applications. In this paper, we discuss the opportunity of developing automatic generation tools for customized graphics hardware. These tools should generate an almost optimized RTL code only for the series of the required graphics operations of the specific application.

1 INTRODUCTION

Due to advancements in both display technology and computing capabilities, 2D and 3D graphics have been included in wide range of applications, such as mobile computing, automotive or avionics. Due to extensive use of graphic applications, extensions of standardized APIs for these types of applications have been developed in recent years. Such extensions include the OpenGL ES, the OpenGL API dedicated for mobile embedded systems, or OpenGL SC, the API targeting safety critical applications. Providing adequate hardware support is crucial in order to implement complex 2D/3D applications. While in mobile computing, dedicate graphic accelerator IPs are integrated in mobile processors, in automotive or avionics the preferred solution is represented by the FPGAs. The main advantages of utilizing these devices are:

1. The display technologies and their corresponding interfacing are changing frequently; therefore, new IPs have to be developed for these displays (Altera, 2009) (Xilinx, 2011)
2. The market in some applications is relatively limited; thus, providing dedicated ASIC based solution will result in high costs (Dutton, 2010)
3. FPGA can accommodate an entire system, including base processor core, graphic hardware acceleration and display interface (see Fig. 1) (Altera, 2009) (Xilinx, 2011).

Furthermore, as explained in (Majer, 2008), future graphic cards may contain reconfigurable fabrics, offering the prospects of mechanisms for run-time exchangeable hardware accelerators.

The main attribute of FPGA based solutions is their high flexibility. Therefore, solutions for such platforms can target a very narrow range of applications, which for ASICs does not represent a cost effective option. As presented in (Dinechin, 2011), dedicated hardware IPs represent the objectives of FPGA based design, and not more general solutions which are favoured for ASIC. In order to aid the IP developers, automatic generators of RTL code have been developed for specific applications. These include floating point arithmetic units (Dinechin, 2011), fast Fourier transforms (Nordin, 2005) or FIR filters (Daitx, 2008).

In this paper, we discuss the opportunity of providing automatic RTL code generators of graphic hardware accelerators for reconfigurable platforms. In the second section, we present the current solutions of 2D/3D graphic engines implemented in FPGAs. The third section represents an overview of the main developments regarding automatic hardware generators for FPGA.

2 FPGA IMPLEMENTATIONS OF GRAPHIC ENGINES

Several implementations for FPGA based acceleration of graphic engines have been developed

Amaricai A. and Boncalo O..
AUTOMATIC GENERATION OF FPGA HARDWARE ACCELERATORS FOR GRAPHICS APPLICATIONS.
DOI: 10.5220/0003906903830386
In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems* (PECCS-2012), pages 383-386
ISBN: 978-989-8565-00-6

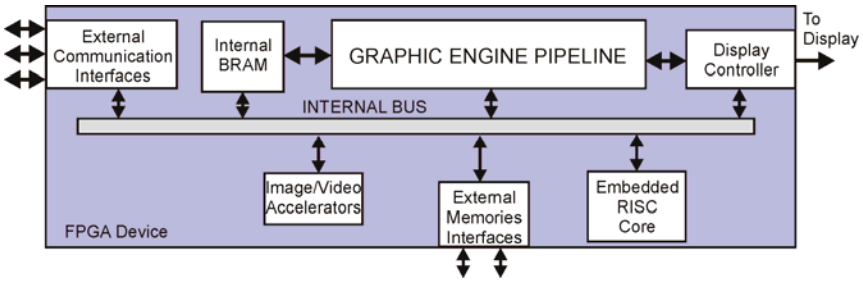

Figure 1: Graphic acceleration system on FPGA.

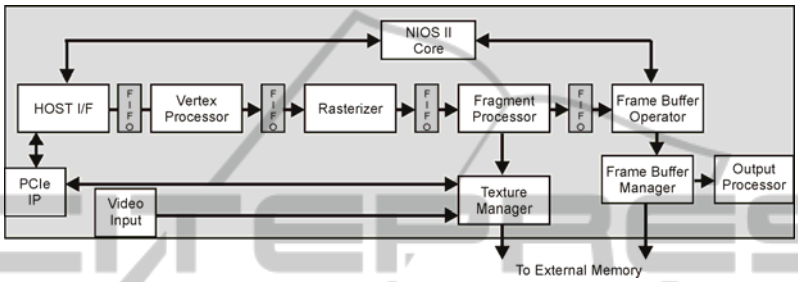

Figure 3: The (Dutton, 2010) graphic pipeline for avionics.

(Chen, 2011) (Mahdavikhah, 2010) (Majer, 2008). Regarding commercial IPs, these include the logicBricks 2D/3D graphics engines (logicBricks, 2011) and the D/AVE 3D graphic IP (TES, 2009). Regarding academic design, these include the rendering engine proposed within the Erlangen Slot Machine (Majer, 2008) or the OpenGL compliant graphic engine for avionics (Dutton, 2010).

The logicBricks 2D/3D (logicBricks, 2011) engine has been developed specifically for the automotive market. It targets Xilinx devices, and is included in the Xilinx Zynq-7000 System-on Chip. It incorporates a scalable 3D pipeline (Fig. 2), which include a shader, clipping and depth tests, a texture unit, a fog unit and an alpha blender. Furthermore, it has a separate post-filtering unit, targeting especially anti-aliasing. The proposed engine can connect via the AXI bus to memory, processor core or other IPs. The geometry operations are performed in software, usually by the ARM Cortex-A9 core with Neon coprocessor. D/AVE 3D graphics IP (TES, 2009) are targeting Altera devices.

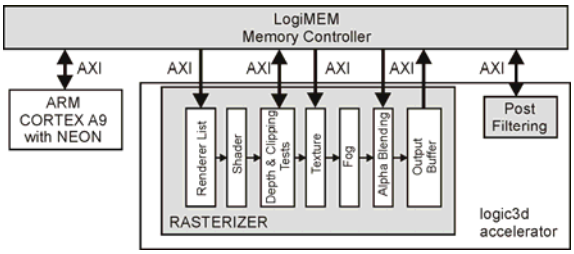

Figure 2: LogicBricks 3D graphic accelerator.

It includes a stream controller, a vertex processor, a geometry setup processor and a pixel pipeline. Cache strategies are used for textures (2 cache memories), frame buffer and ZSA. It can interface to the rest of the system either via an Altera Avalon bus, either via AMBA AHB or APB bus. It provides support for OpenGL ES 1.1, OpenGL VG 1.01 and EGL 1.3.

The rendering pipeline used in the Erlangen Slot Machine (ESM) is compatible to both OpenGL and Direct3D (Majer, 2008). It targets Xilinx FPGAs and its architecture and communication policy follows the structure of the ESM. The latter consists of a high performance FPGA on which the graphic pipeline is implemented and a lower performance crossbar FPGA which connects the first with the PowerPC processor and the rest of the system. As in the case of the logicBricks IP, a hardware/software partitioning is performed in order to cover the entire OpenGL or Direct3D operations.

The graphics engine presented in (Dutton, 2010) matches the full OpenGL pipeline. It includes a vertex processor, a rasterization pipeline, a fragment processor and a frame buffer operator. FIFO buffers are used between these stages. External memories are used for texture and frame buffer. It targets Altera devices and uses a Nios II core as host processor. It connects to other modules of the system via a PCIe bus. As this engine targets avionics applications, all the IPs used have been certified according to avionics standards.

The above engines make use of a relatively

"fixed" IP, similar to the one implemented in GPUs. In many cases, using a hardware/software partitioning, the presented graphic IPs can cover the entire OpenGL API which has been destined. However, FPGAs can be used for very specific applications, a very specialized hardware accelerator being suitable and desired (due to both cost and performance advantages) for reconfigurable devices.

3 AUTOMATIC GENERATORS FOR FPGA ACCELERATORS

A recent research direction that benefited from a lot of attention is that of automatic generation of soft cores for device specific applications. The wide application area and versatility of FPGA devices makes them attractive for a wide range of hardware accelerators (Daitx, 2008). These software tools aim at demolishing the myth of "one size fits all", and to offer the best trade-off within application requirements (e.g. performance, area, power). The generation process provides support for automatized design space exploration by tuning micro-operation concurrency degree (Milder, 2012) (Nordin, 2005), operator size (Dinechin, 2011)(Daitx, 2008)(Liang, 2003), fused paths (Dinechin, 2011), sub-block hardware reuse (Milder, 2012). These processes are mainly fully automated (Daitx, 2008) (Liang, 2003) (Milder, 2012), or as in the case of FloPoCo for some aspects of the design the user is guided throughout the process (i.e. pipeline construction).

FloPoCo (Dinechin, 2011) can be used to generate complex floating point (FP) datapaths. It provides a library of fixed and FP operators (e.g. sqrt, exponential, multiply, divide, add/substract).

Furthermore, it can be used to generate polynomial evaluators. Furthermore, it is intended as a back-end for high-level synthesis tools. The featured optimizations refer to redundant micro-operation reduction and pipelining (requires user intervention). A different FP generator with a slightly more reduced scope is (Liang, 2003) for operations such as multiply, add/subtract, and divide. A particular optimization for this approach is the selection of the FP algorithm in order to meet the desired design parameters tradeoffs.

Firgen (Ruckdeschel, 2005) and (Daitx, 2008) generate FIR hardware modules. The first uses compiler techniques such as: hierarchical partitioning, partial localization in order to generate the design in a systematic manner. The main ingredients are: automatic selection of pipeline stages, selection of memory elements, and concurrency through use of a processing elements array. (Daitx, 2008) targets tradeoffs between HW resources and performance for FIR. The required tuning is done by means of the filter parameters (coefficients, filter traps, type, input size). It starts with off-line coefficient computation, and takes advantage of the application particularities in order to reduce design exploration to a problem of multiple constant multiplications. Both approaches require some sort of offline processing. Firegen (Ruckdeschel, 2005) does offline characterization through a number of runs for design building-blocks, while (Daitx, 2008) performs offline coefficient computation.

ft_gen (Milder, 2012) focuses on DFT soft core generation. The work in (Nordin, 2005), also exploits concurrency through micro-operation (i.e. represented by a hardware algorithm basic block) parallelization. However, its backbone is the Pease d algorithm. Furthermore, the methodology may be used for other linear DSP transforms. The steps of the methodology may be used for other linear DSP transforms.

Table 1: An overview of hardware generators for FPGA.

	Tuning param	Optimizations
FloPoCo	- frequency - operand size	- micro-operation optimizations (redundant microoperations are removed – fuses datapaths of different FP ops)
(Daitx, 2008)	-filter parameters	-offline coefficient computation - sub-block reuse for multiple constant multiplication
Firgen	-operand size -latency &-throughput -filter taps	- compiler techniques (hierarchical partitioning, partial localization) -automatic selection of pipeline stages -selection between shift-regs and FIFOs
(Liang, 2003)	- operand size -throughput &-latency -rounding mode	- automatic FP algorithm selection -pipelining
(Milder, 2012)	- degree of concurrency - storage structures	-mico-operation (basic block) parallelization

4 CONCLUSIONS

FPGAs have become one of the most important platforms to implement dedicate 2D/3D graphic accelerators. Due to their flexibility and their good low-volume price, FPGA based solutions are preferred in applications fields where the display technology is changing frequently or the end market does not support high volume sales. A wide range of graphic IPs have been developed for reconfigurable devices, both industrial and academic. These approaches implement a more general rendering pipeline, on which graphic operations are "programmed". Many of the graphic IPs can be used for implementing the entire OpenGL set of operations.

Automatic generation tools for hardware IPs for FPGA have been developed for floating point units, discrete Fourier transform or FIR filters. The goal of these approaches is to optimize a specific set of operations which can be used for a very narrow range of applications. One project which exemplifies this type of approach is represented by the FloPoCo project, which aims at delivering hardware for any given set of arithmetic operations. The optimizations of such system result from the elimination of redundant micro-operations, such as normalizations or roundings.

This type of approach can also be used for dedicated graphic pipelines. The generated hardware will implement only the required sub-operations of the graphic pipeline, with a lower hardware cost and a possible performance improvement. This way, a restricted set of OpenGL functions, which are specific to a narrow range of applications, can be used on the generated graphic engine. The approach relies on the high flexibility offered by FPGAs, a different type of graphic application requiring a different accelerator IP.

ACKNOWLEDGEMENTS

This work was partially supported Romanian National Authority for Research CNCS – UEFISCDI project PN-II-RU-TE-2011-3-0186.

REFERENCES

Chen S. H., Lin. H. M., Wei H. W., Chen Y. H., Huang C. T. Chung Y.C., 2011 Hardware/software co-designed accelerator for vector graphics applications. In *Proceedings of 9th IEEE Symposium on Application Specific Processors*

Daitx, F. F.; osa, V. S.; Costa, E.; Flores, P.; Bampi, S., 2008. VHDL generation of optimized FIR filters, In *Proceeding of 2nd International Conference on Signals, Circuits and Systems*

Dinechin F., Pasca B. 2011, Designing Custom Arithmetic Data Paths with FloPoCo, In IEEE Design & Test

Dutton, M. , Keezer, D., 2010 The Challenges of Graphics Processing in the Avionics Industry. In *Proceedings of 29th Digital Avionics Systems Conference*

Liang J., Tessier R., Mencer O., 2003, Floating Point Unit Generation and Evaluation for FPGAs, In *Proceedings of 11th Annual IEEE Symposium on Field Custom Computing Machines*

Mahdavikhah B., Mafi R., Sirouspour S., Nicolici N., 2010, Haptic Rendering of Deformable Objects using a Multiple FPGA Parallel Computing Architecture, In *Proceedings of 18th ACM Symposium on Field Programmable Gate Arrays*

Majer M., Wildermann S., Angermeier J., Hanke S., Teich J. , 2008 Co-Design Architecture and Implementation for Point-Based Rendering on FPGAs. In *Proceedings 19th IEEE/IFIP Symposium on Rapid System Prototyping*

Milder P., Franchetti F., Hoe J., Püschel M. 2012, Computer Generation of Hardware for Linear Digital Signal Processing Transforms, In *ACM Trans. On Design Automation of Embedded Systems*

Nordin G., Milder P., Hoe J., Puschel M. 2005, Automatic Generation of Customized Discrete Fourier Transform IPs In *Proceedings of 2005 Design Automation Conference*

Ruckdeschel H., Dutta H., Hannig F., Teich J., 2005, Automatic FIR Filter Generation for FPGAs. In *Proceedings of 5th International Symposium of Embedded Computer Systems: Architectures, Modeling and Simulation*

Altera, 2009, *Using FPGAs to Render Graphics and Drive LCD Interfaces*, White Paper

logicBricks, 2011, logi3D Scalable 3D Graphic Accelerator, Data Sheet

TES Electronic Solutions, 2009, D/AVE 3D Graphic Accelerator, Data Sheet

Xilinx, 2011, *Addressing the Graphics Revolution for Automotive Instrumentation Design Using FPGAs*, White Paper