Kamil PANEK, Bartłomiej FLAK, Sebastian KORYCIAK, Kazimierz WIATR
AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY,
30 Mickiewicza Ave, Cracow, Poland

Basic 3D graphics processor implemented on small FPGA

Abstract

FPGAs have big computing possibilities and therefore are very popular as dedicated hardware accelerators. A few years ago, FPGAs were expensive and the cheapest ones had very limited capabilities, because of small amount of logic elements and slow internal clocks. Nowadays, cheap development boards are available at a price below 50€ with abilities to transmit even HDMI signals. This paper covers implementation of the soft processor with a 3D graphics coprocessor on the cheapest available FPGA board with HDMI connector, containing only 8k Logic Elements.

Keywords: FPGA, Verilog, 3D graphic, HDMI, Bresenham's line algorithm, perspective projection.

1. Introduction

The goal of this project was to create FPGA processing system containing microprocessor and hardware 3D graphic generator. Hardware-based peripheral gives the ability to fast process incoming data [1] and control HDMI output, while programmable microprocessor can be used to easily control graphics module and add algorithms in common programming languages, like C/C++.

Maximator board [2] was used as a base for the designed system. It contains Intel MAX10 FPGA with 8k Logic Elements (*LE*) and 378 kb of memory in M9K blocks. The most important is, that this board has a HDMI connector, being able to transfer 1024×576 video in 60 frames per second.

An important thing besides FPGA chip is that it can be featured with a programmable microprocessor unit, which will be important in process of controlling programmable logic part of this project. This feature opens the new possibility of creating more advanced embedded systems containing not only bare logic elements, but also complete processing systems with custom built IP cores connected to its peripherals.

Manufacturer of FPGA chip offers a free version of its FPGA design software, called Quartus, which was used to design hardware modules in Verilog HDL and microprocessor system.

For developing software running on a CPU core, Intel provides custom Integrated Development Environment (*IDE*) based on Eclipse distribution.

2. System pipeline

In the project, a hardware-based simple graphics pipeline is implemented (Figure 1). It consists of two stages: first, which gets a pair of 3D points from NiosII microprocessor and generates a pair of 2D points in perspective projection (*Perspective Generator*) and second, which draws a straight line between points using Bresenham's line drawing algorithm (*Line Generator*). Output pixels from Line Generator, which are desired points on LCD display are saved to Dual Port RAM created on M9K memory blocks. Independently, HDMI module reads the content of RAM using the second port and send signals to a display. First and last two modules are not part of the graphics pipeline, but are required in the whole system.

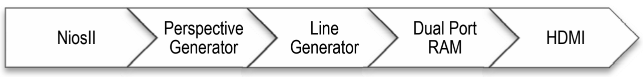


Fig. 1. System pipeline

2.1. NiosII

NiosII is a 32-bit RISC embedded processor designed by Altera. Representing a soft-core architecture that can be implemented in FPGAs programmable logic and memory blocks, it is a flexible solution in many different designs. In this project, NiosII was used as a graphics module controller which provides necessary data like wireframe points or requested view angles to the accelerator.

The first crucial thing when configuring custom processing system is the CPU core. In this case, NiosII/e (economy version) was used. This distribution consists of less than 600 logic elements. This is important when using development board with not so many *LE* available.

To make the CPU core able to store code and data, on-chip memory block was connected. At the first stage of development process, only 4 kB of chip memory was added. 6 kB from MAX10 memory space was configured as processor memory in the final development stage.

To make debugging and programming available, a JTAG core was included in the configuration. The binary application can be downloaded to chip with USB Blaster tool.

The main part of the created system is graphics accelerator module. It is connected to the processing system via Avalon bus, which provides easy access to custom peripherals.

On the other side, UART IP Core from Altera's University Program [3] enables connection with the external world. For purpose of this project, a special command line interface was implemented. This means that end-user can easily send wireframe points values into an internal buffer and then optionally choose the desired configuration for drawing.

Communication between core and modules is done with the use of Avalon Memory-Mapped interface - an address-based read/write interface typical for master-slave connections [4]. Data to transmit is being written into proper registers, which are accessed with special macro definitions. Some of the components have special software drivers generated during creation of board support package (*BSP*) files.

The complete system configuration is shown in diagram (Fig. 2).

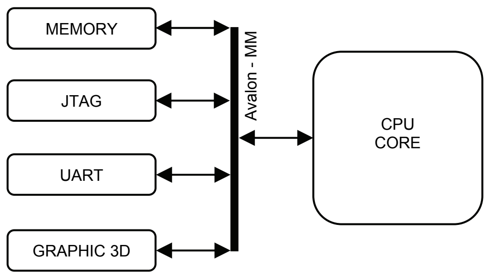


Fig. 2. NiosII processor with connected peripherals

Software for processing system has been written in C language. The program runs as a bare-metal application, however, there is a possibility to use a custom-built real-time operating system.

The main feature of the created application is a command line interface that allows end-user to proceed changes in display configuration of the graphics accelerator. Available commands are:
- *draw* - draws a figure on screen,
- *clear* - erase all objects,
- *turnr* - rotate object clockwise by ~2,8°,
- *turnl* - rotate object counter-clockwise by ~2,8°.

More options are willing to be implemented.

2.2. Communication interface

The communication interface for graphics controller is based on Intel's Avalon Memory-Mapped interface. It contains six 32-bits registers: five for writing data and one for reading. Description of used registers is shown in Table 1.

Tab. 1. Registers of Avalon MM

Offset	Type	Description
0	Write	Writing 1 performs the drawing operation
1	Write	Used to define first point coordinates in the format: 2 dummy bits, 10 bit signed X, 10 bit signed Y, 10 bit signed Z
2	Write	Used to define second point coordinates in the same format as above
3	Write	Used to set the angle of rotation in the Y axis, range from 0 to 127 (0 to 360°)
4	Write	Writing 1 clears the screen
0	Read	Used to inform NiosII processor, that desired operation is finished

Writing and reading from registers can be performed in C language by writing and reading from pointers at addresses made by adding peripheral base address and offset multiplied by 4 (or, using pointer arithmetic, by adding just offset). Because of code optimization, which sometimes causes undefined behaviour, two dedicated macros are used instead of direct pointers to memory, as shown in Listing 1.

```
IOWR_32DIRECT (base, offset * 4, value)
IORD_32DIRECT (base, offset * 4)
```

List. 1. Reading from and writing to registers

2.3. Dual Port RAM

Dual Port RAM is used as frame buffer of size 512×576 bits. Although output video format is 1024×576, there was not enough memory on chip to implement full frame buffer, because it requires about 590 kb of memory for binary images and over 14 Mb for 24-bit colour images.

To build memory, synchronous, dual-port M9K memory blocks were used. They are arranged to create the single memory with a data width of 512 pixels and 576 addresses. It allows HDMI module to read an entire line of the screen at once.

Thanks to dual read/write ports, data can be read from memory by HDMI stage, even if another stage is writing to memory at the moment. This simplifies design because HDMI module can be completely independent of other modules. The disadvantage of this approach is that vertical sync is not implemented, which may appear on screen as frame composed of two or more different frames. This effect is very common in video games, where Vertical Sync decreases frames per second to synchronize with display and is therefore rarely used.

After all, this effect is barely visible and in this simple implementation problem of vertical sync has been ignored.

2.4. Perspective Generator

To get the illusion of 3D object presence on 2D screen, perspective calculations must be done. In this project, the simple perspective projection is used. This requires assumption, that object is located on the Z axis. Then, all calculations can be made using two equations [5]:

$$x' = \frac{x*d}{z+d}, y' = \frac{y*d}{z+d}, \qquad (1)$$

where *x*, *y*, *z* are coordinates of the point in 3D space, *d* is the distance from the spectator, and *x'*, *y'* are point coordinates on the 2D screen. These calculations are made using Intel's *LPM_DIVIDE* megafunction, which allowed to save about 1,000 Logic Elements, compared to performing division in Verilog using slash mark. One calculation unit is initialized (Listing 2).

```
div d(input_z+256, input_xy*256, output_xy);
```

List. 2. Perspective calculations

However, the above calculations are only a part and last stage of Perspective Generator.

Before it, the positions of points and angle of rotation are received from the microcontroller using Avalon Memory-Mapped interface and points are rotated using predefined (precalculated using Python script) sin and cos values, saved in device registers.

2.5. Line Generator

Line Generator stage uses, as previously mentioned, Bresenham's line drawing algorithm [6]. This algorithm is used to draw pixels between two points, to get a close approximation to a straight line as in the example shown in Figure 4. Verilog implementation is based on David Rousset's C# implementation [7].

The algorithm is implemented as FSM, shown in Figure 3, and draws pixels one by one. When a pixel is generated to draw, corresponding *y* coordinate is set as a memory address and the memory is read. In the next step, bit 1 is shifted left by *x* positions, disjoined bitwise with previously read data and saved at the same address. Bitwise disjunction is needed to keep previously written pixels while adding a new one.

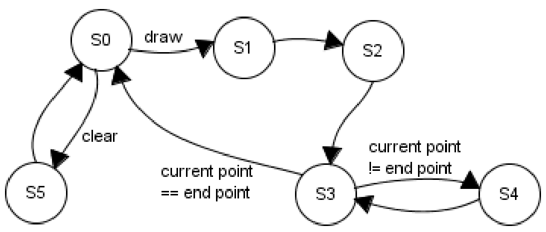


Fig. 3. Bresenham's algorithm FSM; S0 – idle; S1 – calculate the width, height, start point; S2 – calculate line direction; S3 – write current point to RAM; S4 – calculate the position of next point; S5 – clear screen

Because of the limited size of RAM, the displayed image has only 2 colours – black and white and there was no need for anti-aliasing generated lines. In case of project transfer to FPGA chip with more memory blocks, Breseham's algorithm can be replaced by Wu's algorithm, which is a line drawing algorithm with anti-aliasing.

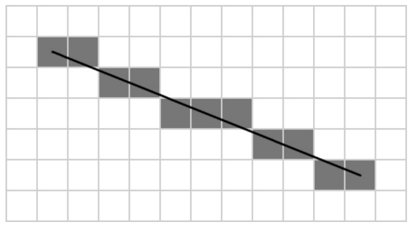


Fig. 4. Example of the line generated by Bresenham's algorithm

2.6. HDMI module

Maximator board has two video connectors – analog VGA connector and digital HDMI. Because VGA is becoming outdated and is less resistant to interferences, HDMI connector and interface has been used. Implementation is based on an example by Jean P. Nicolle [8]. It was adopted to Intel's FPGA and extended by the ability to read data from memory.

In a basic implementation, HDMI uses four differential pairs of signals. Three for colour channels (RGB) and one for the pixel clock. Although the colour depth is 8 bits per channel, HDMI uses 30 bits to send one pixel. The reason for this is using Transition Minimized Differential Signaling (*TMDS*), which is the technique of encoding allowing to achieve high data throughput and minimizing the negative impact of interferences between wires [9].

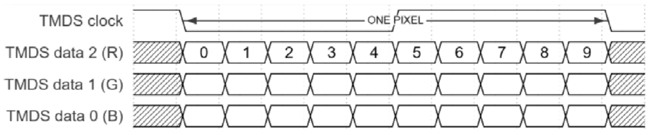


Fig. 5. HDMI basic signals

To get output resolution of 1024×576, HDMI module has to transmit video of size 1120×600. Additional area is used by a monitor for horizontal and vertical syncing. It can also be used for transmitting sound. TMDS clock has frequency of 40 MHz and data bits are clocked at 400 MHz, which is the limit of the used chip. Video has standard refresh rate of 60 Hz.

Because HDMI transmits images line-by-line, storing the entire lines of the image together in RAM simplifies the readout process. Figure 6. shows part of screen connected to the Maximator board, which display regular dodecahedron, from points generated in Blender software [10].

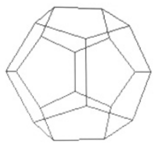


Fig. 6. Dodecahedron displayed on LCD screen (processed photography)

3. Conclusions

The created system has basic capabilities for generating 3D graphics with an easy-to-use interface. It contains NiosII soft-processor and hardware-based graphics accelerator written in Verilog HDL. Logic Elements utilization of all stages is about 4,000 (50%), but memory utilization reaches 100%, which is the bottleneck of the device.

The goal of the project has been reached, but the created system can be further developed. After moving to better FPGA chip, display area and colour depth can be enhanced. This will allow to display anti-aliased lines, drawn using Wu's algorithm. There is also the possibility to create a simple game on NiosII microprocessor.

This research is supported by statutory founds of AGH UST, Department of Computer Science, Electronics and Telecommunication. No. 11.11.230.017.

4. References

[1] Barszczowski M., Koryciak S., Dąbrowska-Boruch A., Wiatr K.: Communication with n-gram hashing hardware accelerator for the ARM using ACP. Measurement Automation Monitoring, vol. 60, no. 7, pp. 486-488, 2014.
[2] Intel, Avalon® Interface Specifications, MNL-AVABUSREF, 2017.05.08.
[3] Kamami, maXimator - Altera MAX10 FPGA development board, http://maximator-fpga.org/ (09.2017).
[4] Intel, RS232 UART for Altera DE-Series Boards, ftp://ftp.altera.com/up/pub/Intel_Material/16.1/University_Program_IP_Cores/Communication/RS232.pdf (09.2017).
[5] Łukawski G., Lasota M., Michno T.: Rzutowanie równoległe i perspektywiczne, http://achilles.tu.kielce.pl/Members/dkaczmarski/studia-dzienne/grafika_2d_lab7.pdf (09.2017).
[6] Bresenham's line algorithm, https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm (09.2017).
[7] David Rousset, Tutorial part 2: learning how to write a 3D soft engine from scratch in C#, TS or JS – drawing lines & triangles, https://www.davrous.com/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/ (09.2017).
[8] Fpga4fun.com, HDMI, http://www.fpga4fun.com/HDMI.html (09.2017).
[9] HDMI Founders, HDMI Specification 1.3a, https://www.microprocessor.org/HDMISpecification13a.pdf (09.2017).
[10] Blender.org, Open source 3D creation, https://www.blender.org/ (09.2017).

Received: 12.10.2017 *Paper reviewed* *Accepted: 01.12.2017*

Kamil PANEK, BSc

Kamil received his bachelor of engineering diploma from Mechatronics, then he decided to continue studies on Embedded Systems faculty at AGH University of Science and Technology, Cracow, Poland. Most interested in FPGA technology he is focusing on designing image co-processors for embedded chips. Currently he works as embedded software engineer.

e-mail: kamil.panek@wbudowane.pl

Bartlomiej FLAK, BSc

In 2017, he has received his BSc degree in Electronics Engineering from the AGH University of Science and Technology, then he continued studies on the same faculty for MSc degree. Concentrating mostly on embedded operating systems, he is working hard to be a Linux kernel source code contributor. Hardware designer and C language programmer.

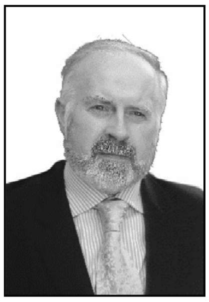
e-mail: flakbartlomiej@gmail.com

Sebastian KORYCIAK, MSc

Received his MSc degree in Electronics Engineering from the AGH University of Science and Technology, Cracow, Poland, in 2011. He is currently working toward the PhD degree. His research interests include SoC and FPGA design for digital image and video processing and coding, low power embedded design, pattern recognition. Currently, he is with Department of Electronics at AGH University of Science and Technology.

e-mail: koryciak@agh.edu.pl

Prof. Kazimierz WIATR, PhD



Received the MSc and PhD degrees in electrical engineering from the AGH University of Science and Technology, Cracov, Poland, in 1980 and 1987, respectively, and the DSc degree in electronics from the University of Technology of Łódź in 1999. Received the professor degree in 2002. His research interests include design and performance of dedicated hardware structures and reconfigurable processors employing FPGAs for acceleration computing. He currently is a director of Academic Computing Centre CYFORNET AGH.

e-mail: wiatr@agh.edu.pl