# Graphics on Key

Students : Eyal Sarfati and Eran Gilat
Supervisors: Dr. Shmuel Wimer, Amnon Stanislavsky and Mike Sumszyk
VLSI and HSDSL Laboratories, Dept. of Electrical Engineering.
Technion – Israel Institute of Technology

## Introduction

Computer graphics are commonly used in a very wide range of applications such as gaming, animated cinema movies, driving and flight simulations, medical imaging, etc. High quality real time computer graphics have become a reality mainly due to the development and implementation of custom hardware commonly known as a GPU (Graphics Processing Unit). Typically, GPU boards consist of numerous processors working in parallel in order to achieve the required performance. All the complex computations required by 3D animation are downloaded to the GPU by the main CPU thus enabling the system to perform animation in real time. In general, GPUs are available for desktop computers and are much less common in low power systems such as laptops and other mobile devices.

The goal of this project was to design and implement a 3D graphics processor that can be used as the heart of an external device integrated in a manner similar to a disk on key. Due to its similar physical appearance and mode of use to the disk on key, it was decided to name this device a “Graphics on Key (GoK)”. A standard USB connector will allow the GoK to seamlessly interface to any low-cost, low power device such as a laptop, cellular phone, PDA, etc. and will immediately provide significant enhancement of the graphics performance.

Fig 1 : Graphics on Key Concept

## Animation Algorithm

In the initial stage of the project, a suitable algorithm was selected for the graphics processing. An important feature of the algorithm was that it allowed pipelining of the computations and parallel processing of data. The algorithm includes triangulation of the 3D object, 3D transformation, computation of color and rasterization of each triangle, projection of 3D data onto a 2D plane and determination of which triangles are actually visible to the viewer. The algorithm was implemented in Matlab and was used to successfully perform a 3D animation of an object.

## Algorithm Improvements

The initial algorithm called of specific ordering to the triangulation data, but this would require additional memory accesses. The reorganization of this data reduced the memory accesses to a minimum and provided overall system speed up. In addition, the decision on pixel color was changed to a simpler model which reduced the total amount processing required without degrading the quality of pixel appearance.

## Graphics Processor Unit Architecture

The next stage of the project involved the design and implementation of an efficient architecture. The GoK architecture includes a triangle prefetch and visibility detection unit. In this stage, all triangles not visible to the viewer are discarded thus reducing the amount to triangles to be processed by 50%. The potentially visible triangles are then forwarded to a 3D transformation unit whose task is to compute their new orientation and location. In order to increase the speed of the rasterization stage, each triangle is then divided into two parts each of which can be processed separately and in parallel. This is performed in the triangle pre-proccesor unit which then submits a series of half triangles into a FIFO task queue where they wait till they are dispatched for rasterization.

The rasterization of the triangles is performed by 11 identical rasterization units all working in parallel, each one on a different (half) triangle. The task scheduler unit ensures that the tasks are dispatched efficiently to the 11 rasterization units so as to obtain maximum throughput.

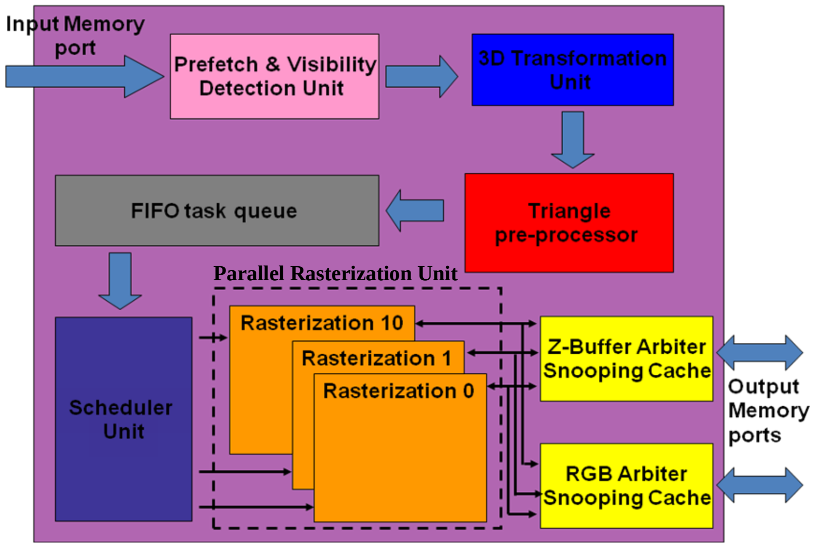

Fig 2 : GPU Architecture

## Prototype Platform

Once the core of the design was completed, the next task was to implement a working prototype on a FPGA. In the final GoK product the triangulation data will be downloaded though a USB2 interface to the graphics accelerators. Once the accelerator completes the computation it will return the pixel data to be displayed to the device (again through the same USB2) at a rate of 25 frames per second. USB2 is a well established interface, however, implementing it was beyond the scope of this project. Therefore, for the prototype built to prove the concept, it was decided to use the readily available USB1 interface to download the triangulation data to the GPU core and to use a VGA terminal to display the animation. The Altera DE3 and DE2 boards were found to be the most suitable platform for this task.

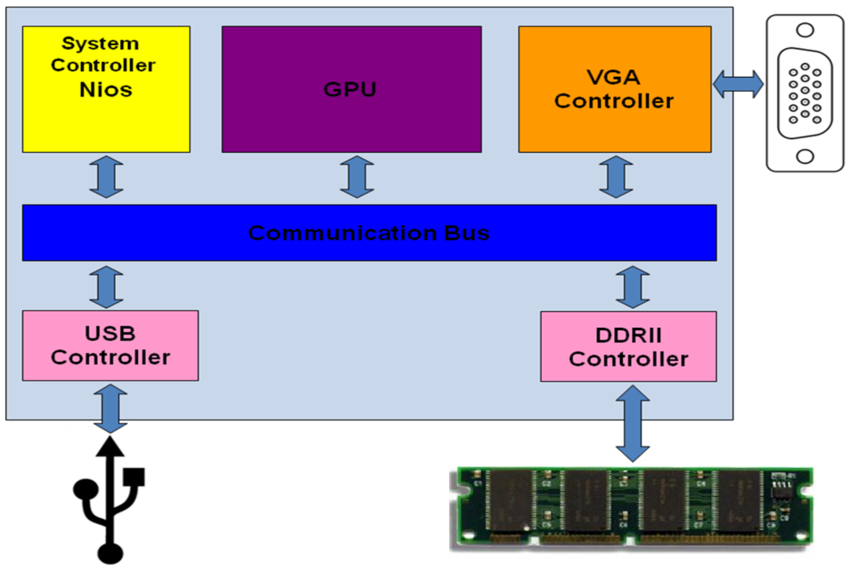

Fig 3 : Altera DE3 based GPU System

Implementing the design on the selected platform presented a number of challenging tasks. Firstly, the device driver available for the USB1 interface proved to be unreliable. An additional software layer was added to the driver to guarantee reliability of the data transfer. The memory interface proved to one of the most challenging issues. The RGB screen data and visibility data was stored in 2 different blocks of memory (implemented on the on board DDR2) shared by the 11 rasterization units. An arbiter had to be implemented to synchronize memory accesses between the 11 units and the two memory blocks. Another problem was that the mode of access to the RGB memory depended to the data read from the visibility memory. As 11 units were simultaneously competing for memory access it was almost certain that the data read from the visibility memory would be outdated and invalid by the time the RGB data could be written. To overcome this problem, a cache memory and cache controller with a snooping mechanism were implemented so as to guarantee that the most updated visibility data was readily available at the time the RGB data was ready for the memory update. The final task was to connect the RGB (pixel) memory to the VGA display. This was done with the aid of an IP block supplied by Altera. This block had to be modified to make it compatible to the DE3 board. In addition, to achieve smooth continuous motion on the display, a double buffer memory configuration was used.

## ASIC Implementation

As stated in the introduction, the design is the basis of a future GoK device. An ASIC version of the GPU was implemented to estimate area, timing and power consumption. The design was synthesized on a 65ns CMOS 8LM process. The post synthesis maximum clock frequency of the core was 300Mhz, the core area was 1.5x1.5 sq mm and the power consumption approx. 130mW.

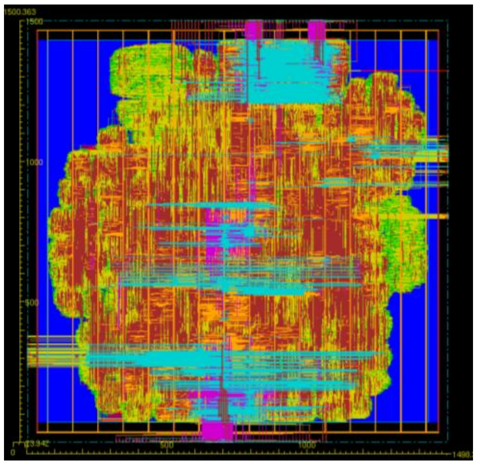

Fig 4 : GPU Layout

## Conclusions

This project successfully implements a prototype of a system that can be used as a basis for the design of a cheap Graphics on Key device that can be used to significantly enhance the graphic performance of low power, low cost gadgets. The implemented prototype receives object data and user animation commands through a user friendly GUI (that was implemented in additional software project), performs the required computations and displays the animation on a 640X480 monitor at 25f/s. The ASIC implementation results confirm that the GPU core is suitable for integration in a disk on key like device.