

**N. PHAM NGOC<sup>1</sup>, G. LAFRUIT<sup>2</sup>, J-Y. MIGNOLET<sup>2</sup>, S. VERNALDE<sup>2</sup>, G. DECONINCK<sup>1\*</sup>, and R. LAUWEREINS<sup>2</sup>**

<sup>1</sup>Katholieke Universiteit Leuven-ESAT/ELECTA

Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

<sup>2</sup>IMEC-DESICS, Kapeldreef 75, B-3001 Leuven-Heverlee, Belgium

Nam.Phamngoc@esat.kuleuven.ac.be

## **REAL-TIME 3D APPLICATIONS ON MOBILE PLATFORMS WITH RUN-TIME RECONFIGURABLE HARDWARE ACCELERATOR**

### **Abstract.**

Due to the limited processing power of current mobile devices and the complexity of 3D content, interactive 3D applications on mobile devices should ideally be boosted by low-cost 3D graphics hardware acceleration. Different from the traditional hardware acceleration ASICs, we've developed a run-time reconfigurable mobile platform which consists of an instruction set processor and a run-time reconfigurable FPGA, supporting 3D real-time rendering. As a proof of concept, we present the implementation of a Quake-alike 3D game on our prototype platform. We show that with hardware implementation, the game can be played at a high, interactive frame rate.

**keywords:** *3D applications, real-time rendering, mobile devices, reconfigurable hardware*

## **1 INTRODUCTION**

Nowadays, with the introduction of the third generation (3G) cellular system using UMTS and with the advanced technologies like wireless LAN, EDGE and Bluetooth, many multimedia applications, including 3D, are emerging on mobile devices such as personal digital assistants (PDA) or mobile phones. Typical 3D applications are 3D mobile navigation [1] and 3D games. However, due to the limited processing power of current mobile devices and the complexity of 3D content, high quality interactive 3D applications should ideally be boosted by low-cost 3D graphics hardware acceleration. Much effort has been spent in

---

\* Geert Deconinck is a postdoctoral fellow of the Fund for Scientific Research-Flanders-Belgium

industry to develop special hardware accelerators for 3D applications on mobile devices [6]. Unfortunately, this approach does not provide flexibility, because new hardware has to be designed for each new emerging rendering technique. Moreover, if multiple, diverse applications have to be running on the same mobile device, adding hardware accelerator for each application incurs a high cost w.r.t hardware area and power consumption.

Based on our experience in reconfigurable systems [2] [3], we are convinced that instruction-set processors (ISP) combined with run-time reconfigurable hardware provide the best platform for future mobile devices. The introduction of the reconfigurable hardware represents an acceptable trade-off between low power consumption for high computation power on one hand, and flexibility by its reconfiguration aspects on the other hand. For example, if a user wants to play a Quake-alike 3D game from a service provider, the 3D polygonal rendering engine will be downloaded and implemented on the reconfigurable hardware. If however the user wants to play a 3D game using image based rendering techniques from another service provider, the image based rendering engine will be downloaded and implemented on the reconfigurable hardware. This example shows the flexibility of such a platform.

This paper presents a prototype platform for mobile devices, which enables real-time rendering of 3D applications. As a proof of concept, the implementation of a Quake-alike 3D game on the platform will be presented.

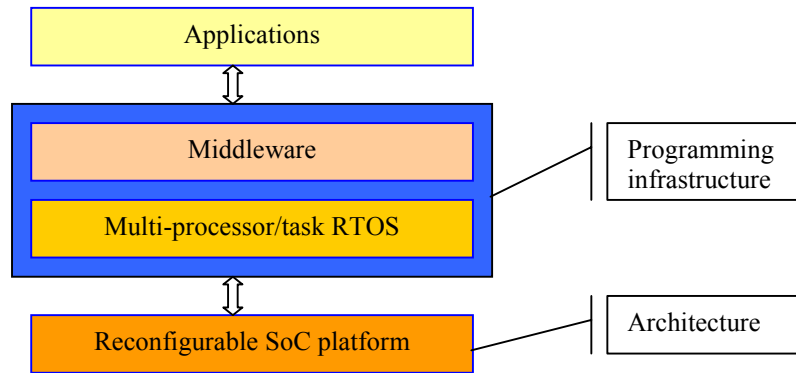
The rest of the paper is structured as follows. Section 2 presents the conceptual platform and a prototype platform. Section 3 describes the 3D game. The implementation of the 3D game and experimental results are detailed in section 4 and finally section 5 presents our conclusions.

## **2 RUN-TIME RECONFIGURABLE PLATFORM FOR MOBILE DEVICES**

We first propose the conceptual platform and its supported software environment for future mobile devices. We also present a prototype platform as an instance of the conceptual platform.

### **2.1 The conceptual platform**

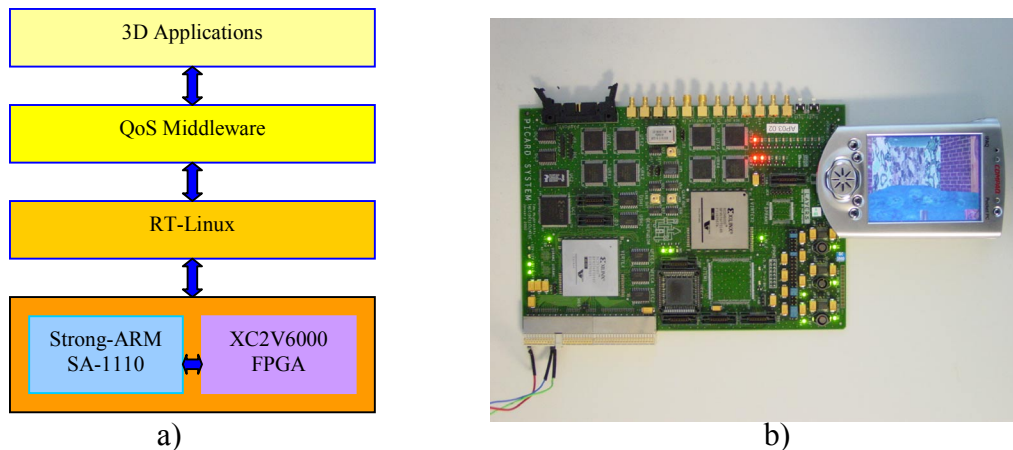
The conceptual platform consists of three layers as shown in Fig. 1. The lowest layer is the physical platform which consists of the reconfigurable hardware and the ISPs. The middle and upper layers compose the programming infrastructure that needs to be built on top of the physical platform to allow easy interaction with the applications. The middle layer is the operating system, which has to enable multitasking of both hardware tasks and software tasks, and has to provide some real-time services. The upper layer is the middleware that creates an abstraction layer for the programmers and provides Quality of Service (QoS) management services. The main function of the QoS management for 3D applications is to guarantee a high interactive framerate by trading off the picture quality or window size for framerate [8]. A detailed description of each layer can be found in [4].



**Fig. 1 Layered conceptual platform**

## 2.2 The prototype platform

We have developed a prototype platform which consists of a Compaq iPaq<sup>TM</sup> PDA, running RT-Linux on its Strong-Arm processor SA-1110 (206 MHz) and controlling a Xilinx Virtex<sup>TM</sup> XC2V6000 FPGA (Fig. 2). The FPGA is the reconfigurable hardware, which is divided into logical tiles of coarse granularity with fine-grain reconfiguration. For allowing run-time reconfiguration, these tiles are connected by a packet-switched interconnection network [5]. The iPaq and the FPGA board are connected together via the Expansion Bus of the iPaq. The communication between the processor and the network on the FPGA is performed by buffering messages in DPRAM. The glue logic on the FPGA board allows the processor to perform a partial reconfiguration of the FPGA. On top of the RT-Linux is the middleware layer, which performs HW/SW partitioning and QoS management.



**Fig. 2 The prototype platform a) system architecture and b) hardware set-up**

### **3 3D APPLICATION**

#### **3.1 Overview of the application**

In order to prove the soundness of the proposed platform, we've developed a Quake-alike 3D shooting game. Fig. 3 shows a snapshot of the game.



**Fig. 3 A snapshot of the 3D game**

The 3D scene in the game is composed of a number of walls, which are represented in the form of a BSP tree [7]. The aim of the game is to shoot the targets on the wall. The user can move left and right to go around the scene and can move the gun up and down to aim the targets. We also added water-ringing effects on the (wet) ground of the 3D scene.

The most important requirement of the 3D game is that it must have a high frame rate to allow the user to move smoothly in the 3D scene. Low frame rates will cause discontinuous and jerky motions and as a result, may make the user miss the targets he/she wants to shoot at.

#### **3.2 Software structure of the 3D game**

The software structure of the 3D game is shown in Fig. 4. The game controller is responsible for handling user interaction, scoring and calculating user viewpoints. The water processing block creates the water ringing effects based on the algorithm presented in [9], which basically performs repetitive 2D filtering operations. The rendering pipeline is similar to that in [7]. The DDA texture mapping algorithm of [7] is used with nearest neighbourhood sampling for texture mapping.

In this game, each wall is seen as one polygon and the texture mapping maps the source image onto this polygon after the polygon has been projected to the screen. Fig. 5 illustrates the texture mapping process in which the destination polygon is scanned vertically and for each pixel on the destination polygon, the coordinates of the corresponding pixel (texel) in the source image are calculated for proper mapping on the screen's destination location.

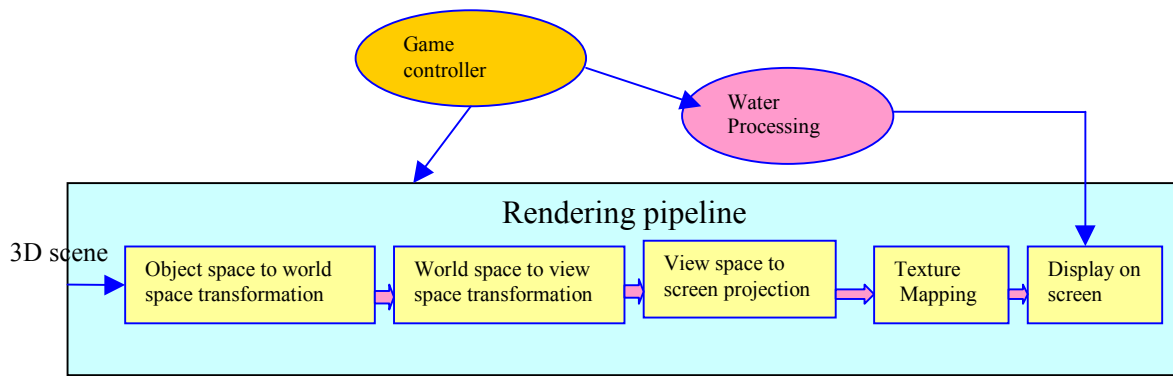


Fig. 4 Software structure of the 3D game

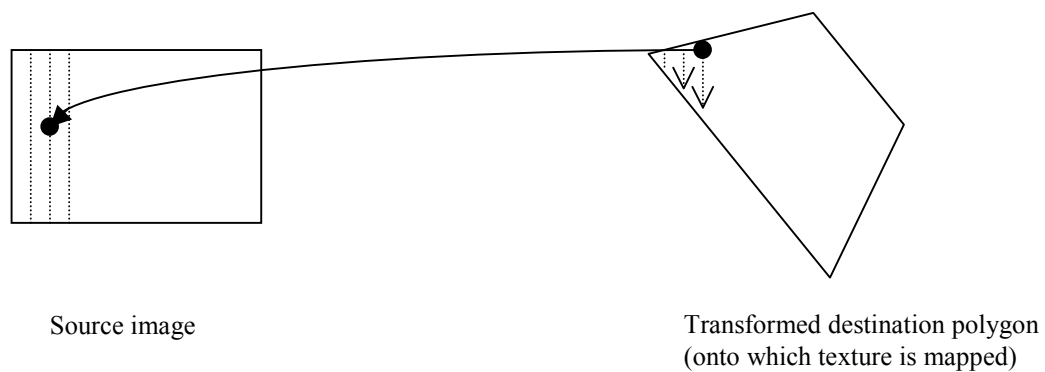


Fig. 5 Texture mapping

## 4 IMPLEMENTATION AND RESULTS

### 4.1 Software implementation

The whole game has been implemented in C and compiled for RT-Linux. The most important optimization step in the implementation is the use of fixed-point operations for replacing floating point emulation. The reason is that the Strong-ARM processor does not have a floating point unit and emulation is very slow (approximately 50 times slower).

Profiling on different software functions of the game pinpointed the bottleneck functions. The profiling results showed that the water processing and the texture mapping functions are the two bottlenecks and hence are good candidates for hardware implementation.

The water processing and the texture mapping functions have originally been implemented as two software threads (or tasks), which are called by a main thread via a common HW/SW API. This allows the operating system to swap between software (SW) and hardware (HW) tasks easily. In our system, depending on the QoS requirement of the user (e.g. the user may want to have a very high interactive frame rate), the QoS middleware may decide on which tasks to put in SW and which tasks to put in HW. Therefore, HW and SW tasks of the same function coexist, which is the case for the water processing and the texture mapping functions.

## 4.2 FPGA implementation

The water processing and the texture mapping functions have been implemented in OCAPI-xl [10][11][12], a C/C++ system-level hardware design environment. VHDL codes have been generated automatically by OCAPI-xl and the synthesis has been performed with Synplicity™ Synplify on the Virtex XC2V6000. The two functions have been put on two separate tiles on the FPGA, which can be seen as two HW tasks. The total hardware area for both functions is 1786 slices, which is about 5.2% of the whole FPGA. The HW tasks are clocked at 40 MHz. The speedup factor of the water processing function is 10. For the texture mapping function, only a speedup factor of 3 is obtained, because its computation workload is limited by the restrictive mode of to nearest neighbourhood sampling.

## 4.3 Results

The simulation results show that when both water processing and texture mapping functions are in software, a maximum of 4fps can be obtained, which is very slow and annoying for playing the game. When the water ringing effect is discarded, a frame rate of 10 fps can be obtained. At this frame rate, the game can already be played, but the motions are still jerky especially when the user moves rapidly in the 3D scene. When both functions are in hardware 22 fps can be reached. Finally, when the water ringing effect is discarded and texture mapping is performed by hardware, a frame rate of 30 fps can be reached. Tab. 1 summarizes the different implementation scenarios. If the user wants to have a very high interactive frame rate, he/she may send this request to the QoS middleware layer via a user interface. The system may eventually decide to discard the water ringing effects and perform texture mapping in HW.

**Tab. 1**

Implementation	Frame rate (fps)
Water ringing effects and texture mapping in SW	4
Without water ringing effects, texture mapping in SW	10
Water ringing effects and texture mapping in HW	22
Without water ringing effects, texture mapping in HW	30

## 5 CONCLUSIONS

As more applications, including 3D applications, are emerging for mobile devices, there is an obvious need for hardware acceleration. We have presented a cost-effective and flexible platform for mobile devices. The platform consists of an instruction set processor and a run-time reconfigurable hardware accelerator. HW tasks can be downloaded and reconfigured on the reconfigurable hardware accelerator at run-time. The implementation of a 3D game application on the prototype system has proven the soundness of the proposed approach. Experimental results show that real-time rendering of the 3D game has been enabled by the reconfigurable hardware accelerator.

## **Acknowledgements**

This work has partly been funded by the IST project OZONE. The work has been done at reconfigurable systems technology group of IMEC.

## **REFERENCES**

- [1] I. Rakkolainen, T. Vainio, “*A 3D City Info for Mobile Users*”, *Computer & Graphics, Special Issue on Multimedia Appliances*, Vol. 25, No. 4, Elsevier 2001, p. 619-625.
- [2] D. Desmet, P. Avasare, P. Coene, S. Decneut, F. Hendrickx, T. Marescaux, J.-Y. Mignolet, R. Pasko, P. Schaumont, D. Verkest, “*Design of Cam-E-leon: A Run-time Reconfigurable Web Camera*”, (accepted), *Lecture Notes in Computer Science*, Springer - Verlag.
- [3] D. Verkest, D. Desmet, P. Avasare, P. Coene, S. Decneut, F. Hendrickx, T. Marescaux, J.-Y. Mignolet, R. Pasko, P. Schaumont, “*Design of a Secure, Intelligent, and Reconfigurable Web Cam Using a C Based System Design Flow*”, *Proceedings of 35th Asilomar Conference on Signals Systems & Computers Pacific Grove*, pp. 463-467, CA, Nov. 2001.
- [4] J.-Y. Mignolet, S. Vernalde, D. Verkest, R. Lauwereins, “*Enabling hardware-software multitasking on a reconfigurable computing platform for networked portable multimedia appliances*”, (accepted ) *ERSA’2002*.
- [5] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde and R. Lauwereins, “*Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs*”, (accepted) *FPL’2002*.
- [6] <http://www.arm.com/armtech.nsf/html/ARM3DGraphics?OpenDocument&style=Graphics>
- [7] M. Abrash, “*Michael Abrash’s Graphics Programming Black Book*”, Coriolis group books, 1997.
- [8] N. PhamNgoc, W. van Raemdonck, G. Lafruit, G. Deconinck, and R. Lauwereins, “*A QoS Framework for Interactive 3D Applications*”, *Proceedings of The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision’2002*, pp. 317-325, 2002.
- [9] [http://freespace.virgin.net/hugo.elias/graphics/x\\_water.htm](http://freespace.virgin.net/hugo.elias/graphics/x_water.htm)
- [10] G. Vanmeerbeeck, P. Schaumont, S. Vernalde, M. Engels, I. Bolsens, “*Hardware/Software Partitioning of embedded system in OCAPi-xl*”, *CODES’01*, Copenhagen, Denmark, April 2001.
- [11] S. Vernalde, P. Schaumont, and I. Bolsens, “*An object oriented programming approach for hardware design*”, In *IEEE Conference on VLSI*, pages 68-73, 1999.
- [12] [www.imec.be/ocapi](http://www.imec.be/ocapi)