We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

3

Design of Embedded Augmented Reality Systems

J. Toledo, J. J. Martínez, J. Garrigós, R. Toledo-Moreo and J. M. Ferrández
Universidad Politécnica de Cartagena
Spain

1. Introduction

The great majority of current Augmented Reality (AR) applications are built using general purpose processors as development platforms where the processing tasks are executed in software. However, software execution is not always the best solution for the high intensive requirements of the many processing tasks involved in AR, and it inevitably constrains frame rate and latency, which compromises real time operation, and magnifies size and power consumption, hindering mobility. These limitations make the spread of AR applications more difficult. This is particularly remarkable in the case of mobile real time applications.

To overcome the aforementioned constraints in the design of embedded AR systems, this chapter presents a hardware/software co-design strategy based on Field Programmable Gate Array (FPGA) devices and Electronic System-Level (ESL) description tools as an alternative to the traditional software-based approach. Modern FPGAs feature millions of gates of programmable logic, with dedicated hardware resources and with the widest range of connectivity solutions. FPGA internal structure makes itself perfectly suitable for exploiting parallelism at several levels. Moreover, because of its flexibility, it is possible to implement not only specific algorithms, but also AD/DA interfaces, controllers, and even several microprocessors, what makes it feasible to build more complex and powerful Systems on a Chip (SoC) with improved performance and reduced costs, size and power consumption. FPGA (re)programmability is also a key factor, which provides not just reduced time to market and design flexibility, but also in-the-field upgradability and intellectual property protection. Thanks to these characteristics, FPGAs are giving rise to a new paradigm in computation named Reconfigurable Computing. ESL, on the other hand, is an emerging electronic design methodology that focuses on building models of the entire system with a high-level language such as C, C++, or MATLAB, which are later used by improved electronic design tools to generate an automated and correct-by-construction implementation of the system. ESL codesign tools allow for developers with little or no prior hardware design skills to implement complex systems composed of mixed software and application-specific hardware modules.

The objective of this chapter is to provide a clear vision of the possibilities of FPGA devices and the new development methodologies for embedded AR systems. To do it so, the authors explain the FPGAs key features which make them suitable for the implementation of AR applications. The design flow and tools for hardware description and

Source: Augmented Reality, Book edited by: Soha Maad,
ISBN 978-953-7619-69-5, pp. 230, January 2010, INTECH, Croatia, downloaded from SCIYO.COM

hardware/software co-design from low to the highest level are described. A survey of the most noteworthy FPGA-based works in image processing, computer vision, computer graphics, multimedia, communications and wearable computing is presented. Finally, the chapter is completed with an example which illustrates the advantages of the FPGA-based approach as platform for developing AR applications: a portable real time system for helping visually impaired people. This system enhances the patient's knowledge of the environment with additional video information using a see-through head mounted display. The description of its main processing cores for video acquisition and processing, for hand recognition, for the user interface, etc. and the evaluation of their performances highlight the advantages of the FPGA-based design and reveal the key topics for the implementation of AR systems.

2. On the suitability of reconfigurable hardware for mobile AR applications

After a successful decade of exploration and consolidation of fields and applications, it is time for AR to break the border of the research domain and reach the common people domain. For it, the user needs to feel AR as a part of his own body, not as an external an uncomfortable artefact. Inevitably, this entails ubiquity and mobility. For such a qualitative jump, one of the major challenges that AR has to face is the hardware under applications and the development of new platforms for interaction (Veas & Kruijff, 2008). It is the key to find the optimum solution to the complicated trade-off between quality, speed, power and size. Most AR research published to date relies on the use of general purpose computing hardware to perform computations, render computer graphics and provide video overlay functionality. Systems that rely on general purpose computing hardware are larger in size and consume more power than those which have devices customised for specific tasks. When working in indoor environments, such issues are rarely considered since the systems are not normally required to be mobile, but the challenge of ubiquity and mobility raises a new scenario. In it, doubtlessly, computer processing hardware is one of the research topics where an extra development effort must be done in order to provide a more effective AR experience (Zhou et al, 2008).

Over the last years, several hardware platforms have been introduced to support mobile AR in two different directions: the head-mounted display direction using laptops in backpack systems, and the handheld direction using lightweight portable devices. Both directions share a design feature: the great majority of AR applications consist of software being executed on general purpose processors. However, not all applications can be solved through the use of the traditional software-based approach since it imposes limitations that force the designer to choose between robustness/compactness, power consumption and processing power, what, in last term, difficult the spread of AR.

In spite of their importance, only a minority of papers describing AR mobile systems report on features such as timing performance, frame rates, power consumption, size, weight, etc., which, in the last instance, determine the viability of the AR application. When these data are provided, they show important weak points, like in the case of the ArcheoGuide application described in (Dähne & Kariginannis, 2002), where the incapability of the hardware platform of the system of parallelizing processing tasks causes that authors rule out a gesture recognizer based on interaction due to its interference with the video tracking performance. This is also the case of the video see-through AR system on cellphone described in (Möhring et al, 2004), which hardly can manage four 160×120 12-bit colour frames per second.

Such limitations lead to approaches where the AR device tends to be 'dumb', acting simply as a viewport of the AR application with the largest part of the processing taking place in remote servers. This is the philosophy behind the AR-phone (Assad et al, 2003). In it, the system performance relies on the wireless networking. Without reporting frame rates, the authors admit the convenience of moving some parts of the processing into the user interface module with the aim of avoiding the overload in the communication that ruins the performance. The mobile phone-based AR application for assembly described in (Billinghurst et al, 2008) is also a client/server architecture where the complex processing is executed on a PC. It works with still images instead of video and the virtual model is quite simple, but the data transmission between the phone and the PC over the WLAN rises the time to send and receive an image up to 3.4 seconds. A similar client/server implementation is presented in (Passman & Woodward, 2003) for running AR on a PDA device. It uses a compression algorithm developed by the authors to relieve the communication of the rendering data overhead, but the best case refresh rate hardly reaches two frames per second. Another interesting approach can be found in (Wagner et al, 2005), where the authors present a system architecture for interactive, infrastructure-independent multi-user AR applications which runs on a PDA (personal digital assistance).

Due to these limitations, some applications resign to video processing on behalf of usability. AR on-demand (Riess & Stricker, 2006) presents a practicable solution on low-end handheld devices where images of the real scene are taken only when needed, superposing real-time virtual animations on that single still image. The information is not blended in the field of view of the user, but can be easily watched over the display of a PDA or a head worn display beside the eye. The goal of this approach is to develop a system which does not dominate the user, but offers support when required.

The use of image sensors is probably the most common way to capture the real environment and enhance it with virtual objects, i.e. to create augmented reality. For this reason, it can serve us well to understand the role of FPGAs and custom-made hardware platforms for AR applications. In the cases where designs must be prepared to meet frame rate, image resolution, algorithm performance, power consumption and size requirements, traditional platforms based on desktop computers and their corresponding slight variations based on laptops are unsuitable. Some examples of these works can be found in (Piekarski et al., 2001; Feiner et al, 1997; Hoellerer et al, 1999). Although these works may be interesting for validating their respective systems for their corresponding intended applications, the proposed devices appear to be complex, heavy, expensive and fragile. Developments made in standard microprocessors-based units are not efficient or inconvenient for the purpose of unconstrained mobility. The authors of (Piekarski et al., 2004) noticed this fact, and proposed modifications in backpack designs which aim to improve its size, weight and power consumption. The authors conclude that FPGAs and specialized video overlay units result beneficial to minimize power and to accelerate computation. In (Johnston et al., 2003), the authors point to FPGAs and developments in hardware for embedded systems as the most likely alternative platform for real-time performance of AR devices. Indeed, parallelism and concurrency can be exploited for image processing while keeping power consumption low, bringing a solution to the challenge of embedded image processing.

A good example of exploiting FPGAs with an image sensor can be found in (Matsushita et al, 2003), where a FGPA device controls a fast CMOS image sensor for ID recognition. In (Foxlin & Harrington, 2000), a self-reference head and hand tracker is presented and its applicability in a wearable computer is shown. Smith and colleagues in (Smith et al, 2005)

improved the work presented in (Piekarski et al, 2001) by migrating the proposed hand-tracking algorithm from the laptop to an FPGA. Indeed, this new development allows a further miniaturization of the system and minimizes power consumption. In (Luk et al, 1998; Luk et al., 1999), some good initial examples of the use of reconfigurable computing for AR are shown. In concrete, video mixing, image extraction and object tracking are run on an FPGA-based platform. In this line, our group presented in (Toledo et al, 2005) a fully FPGA based AR application for visual impaired individuals affected by tunnel vision. A Cellular Neural Network extracts the contour information and superimposes it on the patient's view. Finally, the authors of (Guimaraes et al, 2007) present an interesting platform which aims to help developers on the construction of embedded AR applications. The infrastructure is based on FPGA and enables the creation of hardware based AR systems.
This short review is enough to come to the conclusion that mobile AR applications are severely constrained by the up to date usual software-based approach and that many real-time applications require algorithmic speedup that only dedicated hardware can provide. Hardware implementation of algorithms and processing tasks can considerably improve the performance and by the hence the utility/viability of the AR system.

3. FPGA characteristics and architectures

The introduction of the Field-Programmable Gate Array (FPGA) devices about 25 years ago gave rise to the reconfigurable computing concept. Early FPGA generations were quite limited in their capacities. Nowadays, the most advanced manufacturing technologies are used to feature devices with millions of gates of programmable logic, with dedicated hardware resources, with the widest range of system connectivity solutions, enabling more complex and powerful systems on a single chip.
Reconfigurable hardware offers a trade-off between the very specialized solution of application specific integrated circuits (ASIC) and the inefficiency of general purpose processor (GPP), combining the desirable high performance of ASICs with the characteristics of system flexibility of GPPs. Like ASICs, it involves hardware implementation and consequently parallelism and high performance. Like GPPs, it provides reconfigurability, and hence, flexibility and rapid prototyping. The key of reconfigurable hardware lies on that the flexibility is provided by the hardware design rather than by software-programmable hardware. With clock frequencies an order of magnitude lower than that of typical microprocessors, FPGAs can provide greater performance when executing real-time video or image processing algorithms as they take advantage of their fine-grained parallelism.
There are several companies that produce diverse flavours of FPGAs: Xilinx, Altera, Atmel, Lattice Semiconductor, Actel, SiliconBlue Technologies, Achronix or QuickLogic are the main competitors. At an architectural level, however, all of them share some common characteristics, which define an FPGA as a regular structure of programmable modules, including three types of resources:

1. Logic Blocks (LB), used to implement small parts of the circuit logical functions.
2. Input/Output Blocks (IOBs) that connect internal resources with the device pins, and usually adapt the electrical characteristics of the signals to interface the output world.
3. Routing channels with programmable Interconnect Blocks (IBs), which allow the connection between LBs or between these and the IOBs.

In an FPGA, both the connection of the wire resources and the functionality of the logic blocks can be programmed by the user. Logic blocks and routing architectures differ from

vendor to vendor, but their regular distribution make all of them well suited for highly parallelizable algorithms, composed of bit-level operations that can be distributed in regular structures. Nevertheless, they are ill suited to high precision arithmetic operations, such as large operand multiplication or floating-point calculations. The presence of on-chip memory is also limited, thus many applications require the existence of an external RAM chip. The data transfer increases circuit delays, and increases power consumption and board area.

To overcome these difficulties, researchers and manufacturers proposed new architectures with specific purpose resources and advanced configuration characteristics. Up to Several hundred dedicated MACs (Multiply and Accumulate) modules are included in the larger devices for fast DSP calculations. To increase data storage capabilities, RAM blocks are distributed over the circuit, providing several Mbits of dedicated memory. Recently some devices come with dedicated interface controllers, such as Ethernet MAC, USB or PCI bridges. The list includes also AD/DA converters, PLLs for highly customizable clock generators, or hard-core microprocessors (with much better performance than their soft-core counterparts). As an example, Xilinx's latest Virtex 5 chips include one or two embedded PowerPC™ 405 processors. These 32-bit RISC processors from IBM Corporation are included as hard-cores, making them run at around 450MHz, without decreasing the number of user (LB) resources. Several real-time operating systems (RTOS), included different Linux porting are available for this processor, allowing the designer to centre his efforts in the application-specific parts of the project.

Devices can be classified with respect to their *granularity*, which is a measure of the number of resources that the logic block incorporates. *Fine-grain* devices consist of small cells containing for example a single NAND gate and a latch, or a multiplexer-based function generator and a flip-flop, like Actel IGLOO™ devices. In a medium-grain device, the typical LB consists of two or three function generators (called look-up tables, LUT), each one being able to implement any combinational function from 2 to 6 inputs, and (typically) two flip-flops. A LUT can also be configured as a small memory or as a shift register. The LUT's output can be connected to a flip-flop or routed directly to the cell's port. Finally, the logic block can include logic-specific resources, such as fast carry generators. Xilinx's Spartan and Virtex series are a typical example.

IOBs can be configured as input, output or bidirectional ports. They frequently include on-chip pull-up/down resistors and tri-state buffers. Connectivity is guarantied by supporting main standards, including 3GIO, Infiniband, Gigabit Ethernet, RapidIO, HyperTransport, PCI, Fibre Channel, Flexbus 4, between others. Some chips, like the Virtex 6 family, accommodate several multi-gigabit transceivers to perform serial I/O from 3.125, up to 11Gbps.

Most devices can be programmed by downloading a single-bit stream into the configuration memory. The device's configuration is typically memory-based, using any of the SRAM, EPROM, EEPROM or the most recent flash technologies. Other OTP (one-time programmable) devices use fuse or antifuse technologies (for a detailed description of the different technologies and chip characteristics, the interested reader is referred to companies' web pages).

SRAM devices are the dominant technology nowadays. However, SRAM cells are volatile, meaning that the stored information is lost when power is not applied. These devices require and external "boot", and are typically programmed from a host processor or a non-volatile memory after chip reset. Memory-based FPGAs have the advantage of been re- and in-system programmable. Devices can be soldered directly to the PCB, without using special

sockets. If the design changes, there is no need to remove the device from the board, but it can be simply re-programmed in-system, using typically a JTAG interface. On the other hand, the non-volatile EEPROM and flash devices are better protected against unauthorized use and reverse-engineering, because the programming information is not downloaded to the device at power up.

3.1 Dynamic reconfiguration

The re- and in-system programmability of the memory based FPGAs has opened a broad area of new application scenarios. The term (Re-) Configurable Computing refers to computers that can modify their hardware circuits during system execution. The key for this new computing paradigm has been the development of new FPGAs with extremely quickly configuration rates. While first devices required several seconds to get programmed, in newer FPGAs the configuration download can be done in about one millisecond, and devices with configuration times of about 100 μs are expected in the next years.

Dynamic reconfiguration can be used in a number of ways. The least demanding technique consists of switching between several different configurations that are prepared beforehand, what enables to perform more computational algorithms than those permitted by the physical hardware resources. This could be seen as the hardware equivalent of quitting one program and running another. When faster programming times are available, reconfiguration can be done in a kind of context swapping: the FPGA reconfigures itself time-sharing the execution of different tasks, making the illusion that it is performing all its functions at once (multi-tasking). An example of this techniques was used in (Villasenor et al., 1996) to build a single-chip video transmission system that reconfigures itself four times per video frame, requiring just a quarter of the hardware needed by an equivalent fixed ASIC.

The most challenging approach to dynamic reconfiguration, and surely the most powerful, involves chips that reconfigure themselves on the fly, as a function of requirements emerged during algorithm execution. In this computing system, if a functional unit is missed, it is recovered from the resources store and placed on the chip, in some cases replacing the space occupied by another not-in-use circuit. The problem here is double: loading the proper configuration *bitstream* in the device and finding an appropriate location for the unit, what gives an idea of the complexity of the task. To support this kind of applications, some commercial devices (VirtexII, Virtex4) are now offering dynamic partial-reconfiguration capabilities. These devices allow reprogramming only part of the configuration memory, in order to update just a selected area of the circuit.

Partial reconfiguration enables the remote upgrade of hardware across a network, by delivering new *bitstreams* and software drivers to the remote hardware. The benefits of this methodology include shortening time to market, because the hardware can be shipped sooner with a subset of the full functionality, and performance/corrections upgrading without the need for returns. This methodology has been proved in the Australian satellite FedSat launched on December 14, 2002, featuring a configurable computer (HPC-I) that enables the satellite to be rewired without having to be retrieved, thus drastically reducing cost and development time (Dawood et al., 2002).

With their re-configurability characteristics and the introduction of new special-purpose resources blocks, FPGAs offer a number of advantages over classical design methodologies based on general purpose processors or even the more specialized Digital Signal Processors (DSP), such as unmatched parallelism, versatility, and short time to market. Moreover, Re-

configurable Computing has been presented as a promising paradigm that will compete against the traditional von Newmann architecture and its parallel enhancements (Hartenstein, 2002), (Hartenstein, 2004). Augmented Reality applications and, particularly, embedded ones, can take huge benefits from these emerging so called config-ware technologies.

4. Tools and design flows for reconfigurable hardware

From a historical point of view, the first FPGA design tools where traditional schematic based editors coupled with a physical design software that performed the *place and route* of the architecture-specific components (or primitive cells) in which the circuit is decomposed for a particular FPGA device. Schematic-based tools had however several disadvantages. Firstly, the enormous number of sheets that a large design can consist of made it difficult to handle or update, with changes in one area propagating from sheet to sheet. Secondly, the design methodology proposed by the schematic tools consisted of *structural descriptions* of the circuit; that is, a system was described in base to the entities that composed it. However, in most cases, a system is more naturally described in base to its expected behaviour or *functionality*. The aforementioned limitations are the cause of the wide adoption of Hardware Description Languages (HDL), such as VHDL and Verilog, during the last decade.

4.1 Hardware description languages

Been textual, HDLs are more manageable than schematics. One the other hand, both of them support structural descriptions, however, only HDLs allow for higher-level "behavioural" descriptions. Finally, HDLs have specific constructions for modular design and component re-usability. This last characteristic has become crucial, as the size of the standard design has grown from ten or cents of *k*-gates to several million-gates.

Another important characteristic is that high-level HDL descriptions, contrary to schematic descriptions, are technology independent, what allows the designer to synthesize his project over a number of FPGA devices with minor changes at the description level. Different FPGA vendors and architectures can be benchmarked until an optimal implementation is met, with little impact on the design time.

Despite of the use of HDLs, the current design tools and methodologies have became inadequate to effectively manage the tens of millions gates that the silicon technology allows gather together in a single chip, furthermore when the pressure to reduce the design cycle increases continuously.

The tendency has been towards the use of pre-designed and pre-verified cores trying to bridge the gap between available gate-count and designer productivity. Instead of developing a system from scratch, designers are looking at effective methodologies for creating well-verified reusable modules that can be incorporated in a "mix & match" style to the application-specific blocks. These reusable hardware modules are called *cores* or Intellectual Property (IP). To face the design reuse challenge, IP cores must be designed not just application independent, but also technology independent and synthesis/simulator tool independent (Keating & Bricaud, 1999). Beyond typical basic blocks, modules are being designed specifically to be sold as independent articles. These include microprocessors (ARM, MIPS, PowerPC), communication interfaces (PCI, USB, Ethernet), DSP algorithms (FIR/IIR filters, FFT, wavelets), memory elements (SDRAM, FIFO), etc. One of the more

complete on-line IP repositories can be found in (Design & Reuse, 2009). OpenCores is another important repository, based the concept of freely usable open source hardware (OpenCores, 2009).

4.2 System level specification and codesign

The design process based on HDLs like VHDL, Verilog, etc., is not exempt of difficulties, as these traditional methodologies still require deep hardware skills from the designer. Moreover, the high integration levels of current chips have transformed the concept of System On a Chip (SoC) into reality, increasing design complexity up to an unprecedented level. A typical SoC consists of one or several microprocessors/microcontrollers, multiple SRAM/DRAM, CAM or flash memory blocks, PLL, ADC/DCA interfaces, function-specific cores, such as DSP or 2D/3D graphics, and interface cores such as PCI, USB and UART (Fig. 1).

The intensive use of predesigned IP cores can just mitigate the problem, but in a SoC project, designers can not describe the hardware-specific modules at the Register Transfer Level (RTL), as the HDL methodologies propose, and then wait for a hardware prototype before interacting with the software team to put the design together. New EDA (Electronic Design Automation) tools must incorporate system-level modelling capabilities, such that the whole system, software and hardware, can be verified against its specifications right from the beginning of the design process. This requires an integrated hardware-software co-simulation platform that permits to confer hardware engineers the same level of productivity of software engineers.

To meet the challenges posed by the SoC complexity, new languages, tools and methodologies with greater modelling capabilities are been developed. In the area of design languages, there has been a lot of discussion about the role and applicability area of the various existing and new languages. SystemC, SytemVerilog, Verilog 2005, Analogue and Mixed-Signal versions of Verilog and VHDL, or Vera are some of the new proposals.

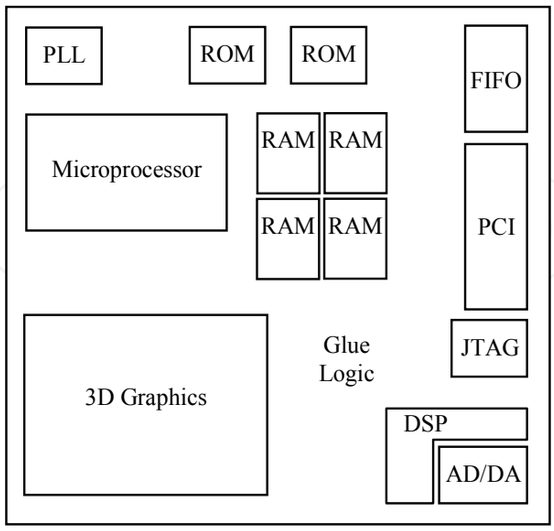


Fig. 1. Structure of core-based system on a chip

One of the most promising alternatives is been developed by the Open SystemC Initiative (OSCI), a collaborative effort launched in 1999 among a broad range of companies to

establish the new standard for system-level design. SystemC (OSCI, 2006) (Black & Donovan, 2004) is a modelling platform consisting of C++ class libraries and a simulation kernel for design at the system and register transfer levels. Been a C-based language, SystemC can bring the gap between software engineers used to work with C and C++, and hardware engineers, that use other languages such as Verilog or VHDL. Furthermore, a common specification language would favour the creation of design tools that allow designers to easily migrate functions from hardware into software and vice versa.
Following this approach, a new generation of tools for highly complex circuit design is been developed. This new methodology, known as ESL (Electronic System Level), aims to target the problem of hardware-software co-design from system-level untimed descriptions, using different flavours of High Level Languages (HLLs), such as C, C++ or Matlab. The main difference between tools is the projection methodology used to implement a given algorithm, that is, the approach used to partition and accelerate that algorithm; some tools used a fixed processor based architecture that can be expanded with custom application-specific coprocessors; other are best tailored to create just custom hardware IP modules that could be later integrated in larger systems; some provide a flexible processor architecture whose instruction set can be expanded with application specific instructions supported by custom ALUs/coprocessors; finally, some of them are intended to provide a complete SoC design environment, giving support for custom hardware modules design, standard microprocessors, application software and the necessary hw-to-hw and hw-to-sw interfaces. A detailed review of these tools is beyond the scope of this chapter, so we will just summarize some of them in no particular order, in Table 1. The interested reader can find an exhaustive taxonomy of the design methodologies and ESL design environments commercially or educationally available in (Densmore & Passerone, 2006).

Company	**Web page**	**Product**
Bluespec	www.bluespec.com	Bluespec Development Workstation
CriticalBlue	www.criticalblue.com	Cascade
Codetronix	www.codetronix.com	Mobius, XPSupdate
Impulse Accelerated Tech.	www.impulseaccelerated.com	CoDeveloper
Mitrionics	www.mitrionics.com	Mitrion Software Development Kit
Nallatech	www.nallatech.com	DIME-C
Poseidon Design Systems	www.poseidon-systems.com	Triton Builder
System Crafter	www.systemcrafter.com	SystemCrafter SC
ARC International	www.teja.com	ARChitect, others
Xilinx Inc.	www.xilinx.com	AccelDSP, System Generator
Mentor Graphics	www.mentor.com	Catapult C
Cadence Design System	www.cadence.com	C-to-Silicon Compiler

Table 1. Some companies providing ESL tools.

4.3 ImpulseC programming model

In this section, we analyze the main features and workflow of CoDeveloper™, an ESL tool from Impulse Accelerated Technologies, Inc. (Impulse, 2009) used for hardware-software co-

design, to evaluate its suitability for the non-hardware specialist scientist in general, as in the case of most AR researches. From our experience, we then provide some keys to get better results with this tool, which may be easily generalized for similar tools, with the aim of making the reconfigurable hardware approach for embedded AR solutions a bit closer for a broaden number of researchers.

ImpulseC compiler uses the communicating sequential process (CSP) model. An algorithm is described using ANSI C code and a library of specific functions. Communication between processes is performed mainly by data streams or shared memories. Some signals can be transferred also to other processes like flags, for non continuous communication. The API provided contains the necessary functions to express process parallelization and communication, as standard C language does not support concurrent programming.

Once the algorithm has been coded, it can be compiled using any standard C compiler. Each of the processes defined is translated to a software thread if the operating system supports them (other tools do not have this key characteristic, and can only compile to hardware).

The entire application can then be executed and tested for correctness. Debugging and profiling the algorithm is thus straightforward, using standard tools. Then, computing intensive processes can be selected for hardware synthesis, and the included compiler will generate the appropriate VHDL or Verilog code for them, but also for the communication channels and synchronization mechanisms. The code can be generic of optimized for a growing number of commercially available platforms. Several *pragmas* are also provided that can be introduced in the C code to configure the hardware generation, for example, to force loop unrolling, pipelining or primitive instantiation.

The versatility of their model allows for different uses of the tool. Let us consider a simple example, with 3 processes working in a dataflow scheme, as shown in Fig. 2. In this case, Producer and Consumer processes undertake just the tasks of extracting the data, send them to be processed, receive the results and store them. The computing intensive part resides in the central process, which applies a given image processing algorithm. A first use of the tool would consist in generating application specific hardware for the filtering process that would be used as a primitive of a larger hardware system. The Producer and Consumer would then be "disposable", and used just as a testbench to check, first, the correct behaviour of the filtering algorithm, and second, the filtering hardware once generated.

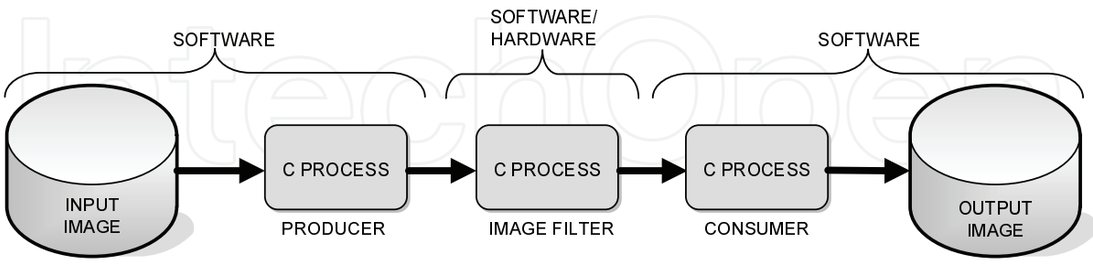


Fig. 2. Typical CoDeveloper model

A different way of using the tool could consist in generating an embedded CPU accelerated by specific hardware. In this case, Producer and Consumer would be used during the normal operation of the system, and reside in an embedded microprocessor. The filter would work as its coprocessor, accelerating the kernel of the algorithm. CoDeveloper generates the hardware, and resolves the software-to-software and hardware-to-hardware, communication mechanisms, but also the software-to-hardware and hardware-to-software

interfaces, for a number of platforms and standard buses. This is a great help for the designer that gets free of dealing with the time-consuming task of interface design and synchronization.
Finally, the objective can be accelerating an external CPU by means of a FPGA board. In this case, the software processes would reside on the host microprocessors, which would communicate to the application specific hardware on the board by means of a high performance bus (HyperTransport, PCI, Gigabit Ethernet, etc.). As in the previous case, software, hardware and proper interfaces between them (in the form of hardware synchronization modules and software control drivers) are automatically generated for several third party vendors.

4.4 Key rules for a successful system-level design flow

The results obtained in our experiments with different applications shown that AR-like algorithms can benefit from custom hardware coprocessors for accelerating execution, as well as for rapid prototyping from C-to-hardware compilers. However, to obtain any advantage, both, an algorithm profiling and a careful design are mandatory. These are the key aspects we have found to be useful:

- The algorithm should make an intensive use of data in different processing flows, to make up for the time spent in the transfer to/from the accelerator.
- The algorithm should make use of several data flows, taking advantage of the massive bandwidth provided by the several hundred o I/O bits that FPGA devices include.
- The working data set should be limited to 1-2MB, so that it may be stored in the internal FPGA memory, minimizing access to external memory.
- The algorithm should use integer or fixed point arithmetic when possible, minimizing the inference of floating point units that reduce the processing speed and devour FPGA resources.
- The algorithm must be profiled to identify and isolate the computational intensive processes. All parallelizing opportunities must be identified and explicitly marked for concurrent execution. Isolation of hardware processes means identifying the process boundaries that maximize concurrency and minimize data dependencies between processes, to optimize the use of onchip memory.
- Maximize the data-flow working mode. Insert FIFO buffers if necessary to adjust clock speeds and/or data widths. This makes automatic pipelining easier for the tools, resulting in dramatic performance improvement.
- Array partitioning and scalarizing. Array variables usually translate to typical sequential access memories in hardware, thus if the algorithm should use several data in parallel, they must be allocated in different C variables, to grant the concurrent availability of data in the same clock cycle.
- Avoiding excessive nested loops. This could difficult or avoid correct pipelining of the process. Instead, try partitioning the algorithm in a greater number of flattened processes.

5. FPGA applications

5.1 FPGA applications in image processing and computer vision

The nature of image processing demands the execution of intensive tasks that in many cases (as it is AR) must meet the requirement of high frame rate. This encourages the use of specific hardware in order to improve the performance of the intended applications. Indeed,

the correct choice of hardware can raise dramatically the system performance. Current systems offer different benefits and limitations depending on the type of processing performed and its implementation. In this sense, general-purpose CPUs are the best alternative for sequential tasks where it is necessary to perform complex analysis of images and to run highly branched algorithms. However, in applications of 3D image processing and fast rendering scenes, the Graphics Proccesor Units (GPU) are more suitable because they have specific processing architectures designed to perform vector operations on large amounts of data. FPGAs are especially suitable for performing pre-processing tasks like colour format conversion, image filtering, convolution, and more in general, any repetitive operation that does not require highly complex algorithms, even in those cases when these algorithms are parallelizable and can benefit from the use of unconventional specific architectures. The large number of memory blocks available on FPGAs provides parallel processing support and enables very fast access to data stored in these caches. Developers can leverage the high bandwidth I/O on these devices and thus increase the speed of the functions and data rate that traverse the FPGA on GPUs or CPUs. Thanks to the versatility to develop dedicated circuits and the high degree of parallelism, FPGAs can achieve performances similar to some other hardware alternatives that run at higher frequencies of operation. These reasons explain why the implementation of many algorithms is the focus of a wide number of works since the last decade, and why from early stages of the evolution of reconfigurable hardware, several FPGA-based custom computing machines have been designed to execute image processing or computer vision algorithms (Arnold et al., 1993; Drayer at al., 1995).

Of high interest for AR applications is the implementation of object tracking algorithms, where different approaches have been followed. For example, the authors in (Dellaert & Tarip, 2005) present an application where a multiple camera environment is used for real time tracking with the aim of assisting visually impaired persons by providing them an auditory interface to their environment through sonification. For this purpose an octagonal board can support up to 4 CMOS cameras, an Xscale processor and a FPGA which handles the feature detection in parallel for all cameras. Another FPGA-based application for counting people using a method to detect different size heads appears in (Vicente et al., 2009). More examples of FPGA-based approaches to object tracking can be found in the literature: for colour segmentation (Garcia et al., 1996; Johnston et al., 2005); for implementing an artificial neural network for specifically hand tracking (Krips et al., 2002; Krips et al., 2003); for recognizing hand gestures (In et al., 2008); and for increasing pixel rate to improve real-time objects tracking by means of a compression sensor together with an FPGA (Takayuki et al., 2002).

Similarly, the human exploration in virtual environments requires technology that can accurately measure the position and the orientation of one or several users as they move and interact in the environment. For this purpose a passive vision FPGA-based system has been proposed by Johnston et al. (Johnston et al., 2005). The aim of this system is to produce a generalised AR system in the sense that accurate estimation of a mobile user's position relative to a set of indoor targets is accomplished in real time. FPGA-based systems are also used to develop a system for tracking multiple users in controlled environments (Tanase et al., 2008).

Vision-based algorithms for motion estimation, optical flow, detection of features like lines or edges, etc. are also widely used in AR. Recently, several motion estimation alternatives have been proposed to be implemented on a FPGA platform. Some of them are compared in

(Olivares et al., 2006). Some other good examples interest of the recent literature on this issue can be found in (Yu et al., 2004), where mobile real-time video applications with good trade-off between the quality of motion estimation and the computational complexity are presented, and (Akin et al., 2009), that focuses on the reduction of the computational complexity of a full search algorithm from 8 bits pixel resolution to one. Optical flow can also be used to detect independent moving objects in the presence of camera motion. Although many flow-computation methods are complex and currently inapplicable in real-time, different reliable FPGA-based Real-time Optical-flow approaches have been proposed in the recent years (Martin et al., 2005; Diaz et al., 2006). Furthermore, other processing image techniques like super resolution, atmospheric compensation and compressive sampling may be useful in enhancing the images and to reconstruct important aspect of the scenes. These techniques are highly complex and the use of FPGAs is encouraged to achieve the necessary acceleration. These topics are covered in detail in several articles dedicated to reconfigurable computing (Bowen et al., 2008; Bodnar et al., 2009; Ortiz et al., 2007).

5.2 FPGA applications in computer graphics and multimedia

Computer graphics is another field that can benefit from the flexibility of software programmable devices. This explains the increasing attention paid to FGPAs in the last years for the purpose of graphics acceleration, traditionally assigned to GPUs or graphics cards.

The suitability of FPGAs for the implementation of graphic algorithms has been analysed since the mid 90s. Singh and Bellec (Singh & Bellec, 1994) introduce the notion of *virtual hardware*, a methodology to execute complex processes on limited physical resources, using the dynamic reconfigurability of FPGAs. More recently, Howes (Howes., 2006) compare the performance of different architectures based on FPGAs, GPUs, CPUs and Sony Playsation 2 vector units on different graphic algorithms using a unified description based on A Stream Compiler (ASC). This work shows how the FPGAs provide fast execution of the graphics algorithm with clocks at lower frequencies than its competitors. Nevertheless, performances are particularly dependent on the possibilities of optimization for each design.

Radiosity high computational cost has been improved using FPGA devices by Styles et al. (Styles & Luk., 2002). Ye and Lewis (Ye & Lewis, 1999) proposed a new architecture for a 3D computer graphic rendering system which synthesizes 3D procedural textures in an FPGA device, enhancing the visual realism of computer rendered images, while achieving high pixel rate and small hardware cost. In order to improve the efficiency of 3D geometric models represented by a triangle mesh, some mesh compression/decompression algorithms were developed. Mitra and Chiueh (Mitra & Chiueh, 2002) proposed the BFT algorithm and presented a novel FPGA-based mesh decompressor.

Styles and Luk. (Styles & Luk., 2000) analyzed the customization of architectures for graphics applications for both general and specific purposes, and prototyping them using FPGAs. Based on their results, the authors remark the suitability of FPGAs. In the same work an API that allows the execution of OpenGL graphics applications on their reconfigurable architecture is also presented.

5.3 FPGA applications in computer graphics and multimedia

Computer graphics is another field that can benefit from the flexibility of software programmable devices. This explains the increasing attention paid to FGPAs in the last

years for the purpose of graphics acceleration, traditionally assigned to GPUs or graphics cards.

The suitability of FPGAs for the implementation of graphic algorithms has been analysed since the mid 90s. Singh and Bellec (Singh & Bellec, 1994) introduce the notion of *virtual hardware*, a methodology to execute complex processes on limited physical resources, using the dynamic reconfigurability of FPGAs. More recently, the authors of (Howes at al., 2006) compare the performance of different architectures based on FPGAs, GPUs, CPUs and Sony Playsation 2 vector units on different graphic algorithms using a unified description based on A Stream Compiler (ASC). This work shows how the FPGAs provide fast execution of the graphics algorithm with clocks at lower frequencies than its competitors. Nevertheless, performances are particularly dependent on the possibilities of optimization for each design. Radiosity high computational cost has been improved using FPGA devices by Styles et al. (Styles at al., 2002). Ye and Lewis (Ye & Lewis, 1999) proposed a new architecture for a 3D computer graphic rendering system which synthesizes 3D procedural textures in an FPGA device, enhancing the visual realism of computer rendered images, while achieving high pixel rate and small hardware cost. In order to improve the efficiency of 3D geometric models represented by a triangle mesh, some mesh compression/decompression algorithms were developed. Mitra and Chiueh (Mitra & Chiueh, 2002) proposed an algorithm and presented a novel FPGA-based mesh decompressor. Styles et al. (Styles at al., 2000) analyzed the customization of architectures for graphics applications for both general and specific purposes, and prototyping them using FPGAs. Based on their results, the authors remark the suitability of FPGAs. In the same work an API that allows the execution of OpenGL graphics applications on their reconfigurable architecture is also presented.

5.4 FPGA applications in communications

FPGA technology has appeared to be also very useful for communication systems. Two important factors encourage its expansion in this field: the falling prices of the devices and the inclusion of DSP capabilities. Typical communication problems such as data formatting, serial to parallel conversion, timing and synchronization can be faced naturally in a FGPA device thanks to its specific features. Furthermore, FPGAs are convenient for the development of the necessary glue logic for the interconnection of processors, modems, receivers, etc. Several examples can be found in the literature of the field. In (Ligocki et al, 2004) the authors describe the prototype development of a flexible communication system based on a FPGA. The main focus of this work is on software concerns, considering that FPGA technologies are the core of the project. Other authors exploited the spatial/parallel computation style of FPGAs for wireless communications. Due to the computational complexity of WLAN (Wireless local area network), and taking into account the capabilities of modern microprocessors, an implementation based exclusively on microprocessors is not convenient, requiring a large number of components. Parallel computation allows improving the efficiency of the implementation of the discrete components, and makes it possible to accelerate some complex parts of WLANs (Masselos & Voros, 2007).

Of special interest results the benefits of FPGAs for embedded software radio devices. In (Hosking, 2008), it is shown how its inherent flexibility makes of FGPA devices an excellent choice for coping with the increasing diverse array of commercial, industrial, and military electronic systems. Additionally, the large number of available IP cores offer optimized algorithms, interfaces and protocols which can shorten significantly the time-to-market.

5.5 FPGA applications in wearable computing

It is a common understanding that the concept of wearable computing comes from the tools developed to intensify the experience of seeing, what led to augmented reality. However, it can be noticed that in the literature different authors understand the concept of wearable systems in very different manners. Some authors claim that a system is wearable as long as it can be transported by a human. According to this, some authors present wearable solutions based on small variations of desktop applications in a back-up with a laptop (Feiner et al, 1997; Hoellerer et al, 1999). However, let us focus in this chapter on only wearable devices which do not constraint the mobility of the user. For this purpose, the use of FPGAs results of the highest interest. In wearable systems, the problem of combining simultaneously high performance and low power consumption requirements in small dimensions can be overcome with FPGAs. Contrary to ASICs (application specific integrated circuit), reconfigurable logic offers more flexibility to adapt dynamically the processes and the possibility of integrating different processing units in only one device. This way, it is possible to reduce the number of chips in a system, which can be an important advantage.

An interesting study on the improvements regarding energy saving when implementing critical software processes on reconfigurable logic can be found in (Stitt et al, 2002). The LART board, presented in (Bakker et al, 2001) combines a low-power embedded StrongARM CPU with an FPGA device, which offers a better power/MIPS ratio, pursuing power consumption reduction. The FPGA is used for dedicated data processing and for interconnecting several LARTs working in parallel. In a step forward, the authors in (Enzler et al, 2001) analyze the applications of dynamically reconfigurable processors in handheld and wearable computing. They consider a novel benchmark which includes applications from multimedia, cryptography and communications. Based on that work, the authors of (Plessl et al, 2003) presented the concept of an autonomous wearable unit with reconfigurable modules (WURM), which constitutes the basic node of a body area computing system. The WURM hardware architecture includes reconfigurable hardware and a CPU. In the prototype, the implementation is done on only one FPGA, including the CPU as a soft core.

Finally, let us remark the importance of networking for wearable computing. Indeed, the constraints in power consumption, size and weight of wearable computers increase the need for network capabilities to communicate with external units. A study about the interest of FPGA approaches for network on chip implementations across various applications and network loads is presented in (Schelle & Grunwald, 2008). Recently, several authors have followed FPGA-based approaches in their solutions. In (Munteanu & Williamson, 2005), an FPGA is exploited to provide consistent throughput performance to carry out IP packet compression for a network processor. (Wee et al, 2005) presents a network architecture that processes in parallel cipher block chaining capable 3DES cores by using about the 10% of the resources of an FPGA Xilinx Virtex II 1000-4. Within the frame of the European Diadem Firewall Project, IBM suggests the use of standalone FPGA-based firewalls in order to achieve an accelerated network architecture (Thomas, 2006).

6. FPGA-based platform for the development of AR applications

We have proposed a platform for developing fully FPGA-based embedded systems aimed for image and video processing applications. It is a hardware/software system created for

speeding up and facilitating the development of embedded applications. As the survey of works in previous sections highlights, FPGA devices are very suitable for the implementation of the processing tasks involved in AR. The adoption of an FPGA-based approach allows executing different tasks and algorithms in parallel, which ensures the best performance and the optimum power consumption. The platform acquires video in standard analogue formats, digitizes, and stores it in external memory devices. In order to confer versatility to the embedded system, the platform includes as a key component an interface which allows for user interaction. This interface makes it possible to display text and, by means of hand pose recognition or voice recognition, to choose options and configure parameters. Thanks to it, the user can customize the functionality of the hardware at run-time.

6.1 Frame grabber

Video is a primary input to AR systems and can be typically used to develop video see-through systems, to execute vision-based tracking algorithms or as input to a user interface. In order to process video we have developed a frame grabber which accepts standard analogue video signal, converts it into digital and stores it in memory.

The frame grabber is based on the SAA7113 video input processor, from Philips Semiconductors, which is able to decode PAL, SECAM and NTSC from different sources (CVBS, S-video) into ITU-R BT 601. It is configured and controlled through I2C bus, so an I2C controller module must be included to properly manage the SAA7113. The SAA7113 presents video data at an 8 bit digital video port output, with 720 active pixels per line in YUV 4:2:2 format and a number of lines in a frame depending on the video standard (PAL, NTSC). The 4:2:2 output format implies that there is a luminance Y value for each pixel, but only a chrominance pair UV for two pixels. The YUV colorspace is used by the PAL and NTSC colour video standards. However, RGB is the most prevalent choice for computer graphics. Therefore, we have included a converter from YUV to RGB in the design. It just implements the corresponding linear equations, which can be found, e.g., in (Jack, 2005).

The image from the SAA7113 must be stored in a frame buffer. In our platform it is made of external asynchronous SRAM memory devices. A memory interface for generating the memory control signals and read /write operations was implemented on the FPGA.

Colour data stored in the memory is in YUV format since it optimizes the space and the access to memory. While in RGB format a pixel is defined with 24 bits, the YUV format from the video codec uses 32 bits to define two pixels, with exactly the same colour information in both RGB and YUV colorspaces. As physically each frame buffer consists of a 32 bit 256 KB SRAM, it is more efficient to store colour information in YUV colorspace, since it is possible to store two pixels in just one address, and so halving the number of access to the memory.

6.2 General purpose user interface

Unlike PC-based solutions, where visualization of text information is completely usual, this is not so natural in hardware-based solutions. Most of the present FPGA-based embedded systems do not offer an interface to the user. Sometimes they just consider a UART to connect with a computer and transfer some information. However, the use of a PC simply for running the software that manages the communications and the interface is a poor, very low efficient solution, even unfeasible in embedded systems. With the aim of overcoming this drawback of FPGA-based embedded systems, we have designed a hardware core which

facilitates the addition of a user interface to FPGA-based systems. The core is based on MicroBlaze, a standard 32 bit RISC Harvard-style soft processor developed for Xilinx FPGA devices. This gives flexibility and versatility, and ensures fast re-design of the hardware architecture for enhanced or new applications. The core is made up of hardware components and software functions in different levels. Thanks to the core it is possible to present text information in a VGA monitor to the user, who can navigate through menus, select options and configure parameters by means of a pointer device. So, it provides the flexibility of adapting the systems to the user requirements or preferences. Next, its basic modules are described.

Display of text and text fonts

In order to display text, all the bitmaps associated to the font characters were previously defined and stored in a ROM memory using FPGA internal logic resources. The ROM works as a MicroBlaze peripheral, using a dedicated Fast Simplex Link (FSL) channel.

To display text, we have considered a text screen, which manages the visualization of the strings. In the default mode, it is a 640×480 array, whose elements correspond to the pixels in the VGA output. A 64 colours palette has been considered, which implies 6 bits for each pixel. Due to its size, the array is stored in external SRAM.

The text screen is defined as a peripheral and connected to the MicroBlaze microprocessor by means of the On-chip Peripheral Bus (OPB). The hardware of this peripheral includes the SRAM memory interface to control write and read operations and the logic to interpret the data from MicroBlaze into address and data buses values. It carries out two different tasks:

- it receives data from MicroBlaze, and manages the write operations in the SRAM memory just when a modification in the text information displayed is done.
- it reads data from the memory to show the text screen in the VGA monitor. These data are sent to the VGA Generator module. This process is independent on MicroBlaze.

In order to create the interface presented to the user, the function mb_OutTextXY has been prototyped to be instantiated in the software application running in MicroBlaze. It is similar to the equivalent standard C function, and it allows to define a text string and to specify its colour and position in the screen. When the mb_OutTextXY function is executed, the writing instruction of the text screen peripheral is called to write in the SRAM the colour values of the pixels which correspond to each element of the string, according to its position and its colour.

Once the strings are stored in the text screen, the basic user application waits for an interrupt from pointer device. When it happens, an interrupt handler classifies the interrupt and reads the coordinates of the pointer position. Since the position of each text string is known, it is possible to determine in the software application which one has been selected by the user, and then to reply with the desired actions.

The pointing device

A pointing device is required to interact within the user interface. With this aim, we have proposed a hand-based interface designed for mobile applications. It detects the user hand with a pointing gesture in images from a camera, and it returns the position in the image where the tip of the index finger is pointing at. In an augmented reality application the camera will be placed on a head mounted display worn by the user. A similar system is proposed in (Piekarski et al, 2004), but our approach is based on skin colour, without the need of glove or coloured marks. Our hand-based interface is aimed for performing

pointing and command selection in the platform for developing FPGA-based embedded video processing systems herein described.

Vision-based algorithms have been used to build the hand and the pointing gesture recognizers. The image from the camera, once acquired, digitalized and stored by the previously described frame grabber, is segmented using an ad-hoc human skin colour classifier. Human skin colour has proven to be useful in applications related to face and hands detection and tracking. With this colour skin approach we try to generalize the use by eliminating external accessories, what reduces costs. The skin colour classifier is made of sub-classifiers, each one defined as a rule-based algorithm built from histograms in a colourspace. The rules in each colourspace define a closed region in the corresponding histogram: a pixel of an image is classified as skin if its colour components values satisfy the constraints established by the rules. Classifiers in the YIQ, YUV and YCbCr colorspaces have been considered, so three sub-classifiers have been implemented. To generate a unique output image, their outputs are merged using logical functions. The use of different colorspaces is aimed at achieving invariance to skin tones and lighting conditions. Further details can be found in (Toledo et al, 2006).

Once the image has been segmented the next processing task is to look for the pointing gesture. The solution adopted consists of convoluting the binary image from the skin classifier with three different templates: one representing the forefinger, other the thumb and the third the palm (Toledo et al, 2007). This modularity makes easier the addition of new functionality to the system through the recognition of more gestures. Due to the size of the hand and the templates, an optimized solution for the FPGA-based implementation of large convolution modules has been specifically developed. It can convolve binary images with a three-value template in one clock cycle independently of the template size. It is based on distributed arithmetic and has been designed using specific resources available in Xilinx FPGAs. The maximum size of the template depends on the FPGA device. In this application images are convoluted with 70×70 templates. Each convolution module sends to the MicroBlaze soft processor its maximum value and its coordinates on the image. A software algorithm running on MicroBlaze decides that a hand with the wanted gesture is present when the maximum of each convolution reaches a threshold and their relative positions satisfy some constraints derived from training data. Then, the algorithm returns the position of the forefinger. Otherwise, it reports that no pointing hand is detected.

The software application on MicroBlaze also includes an algorithm for dynamically adapting the skin classification and the parameters for hand recognition. Taking into account the number of pixels classified as skin in the image, the maximum value and the coordinates of each convolution and the detection or not of the pointing hand pose, it tunes each skin classifier and the merging of their binary output images in order to achieve the optimum classification rates, and it also tunes the values of the different constraints to their right values in order to find the desired hand posture. The FPGA implementation of these tasks allows taking advantage of parallelism in each processing stage at different levels. For example, the three classifiers for skin recognition are executed at the same time on an input pixel. Since the constraints of a classifier are all evaluated at the same time, the time required to classify a pixel is just the maximum delay associated to a constraint, three clock cycles in our case. Besides, the three convolutions that look for the hand position are performed in parallel, and the operations involved in each convolution are all executed at the same time in only one clock cycle. Meanwhile, the software application in MicroBlaze is using the

information extracted by the hardware modules as input parameters to the algorithm which estimates the presence and position of the hand. Thanks to exploiting the parallelism inherent to FPGA devices, the hand detection algorithm can process 640×480 pixel images at more than 190 frames per second with a latency of one frame. It also makes it feasible that new processing cores can be added to the system with small performance penalty.
In addition to the hand-based interface, a controller for a generic PS/2 mouse has been implemented and added to the MicroBlaze system as an OPB peripheral.

6.3 Generation of video signals

The platform can generate signals for displaying video on monitors and analog screens. The generation of the synchronization and RGB signals for a VGA monitor is carried out by the VGA generator module, which can be configured to generate different resolutions. The platform also includes the SAA7121, an integrated circuit from Philips which encodes digital video into composite and S-video signals. The video generator module also deals with the mixing of the video from the different sources included in the platform.

7. Portable real time system for helping visually impaired people

We have validated the usefulness of the described platform in an application for people affected by a visual disorder known as tunnel vision. It consists in the loss of the peripheral vision, while retaining clear and high resolution central vision. As shown in Fig. 3, it is like looking through a keyhole or a ring in the mid-periphery. Tunnel vision is associated to several eyes diseases, mainly glaucoma and retinitis pigmentosa, and reduces considerably the patient's ability to localize objects, which inevitably affects the patient's relationship with people and the environment.

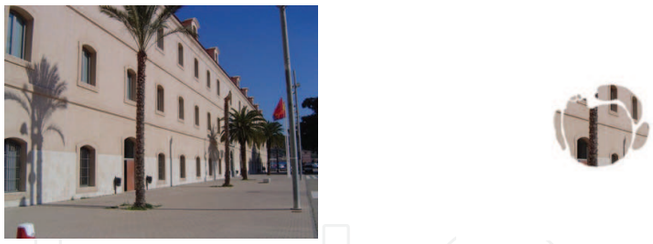


Fig. 3. Simulation of patient affected by tunnel vision view. A residual 10º field of view has been considered to simulate the tunnel vision effect. The severe reduction of the visual field (right) can be observed comparing with the normal vision (left).

To aid affected people, it is necessary to increase the patient's field of view without reducing the usefulness of the high resolution central vision. With this aim, (Vargas-Martín & Peli, 2001) proposed an augmented view system where the contour information obtained from the image of a camera is superimposed on the user's own view. In their work, contours are generated by an edge detection algorithm performed by a four-pixel neighborhood gradient filter and a threshold function, running on a laptop PC (Vargas-Martín & Peli, 2002). They draw the conclusion that, although patients consider the system useful for navigating and obstacle avoiding, a specifically designed system to perform image processing and increase frame rate is necessary. Obviously, an effective improvement of the user's environment perception requires real time processing. To achieve it, we have used our FPGA-based

hardware platform, which ensures the video frame rate and the low latency that the mobility of the application required.

Fig. 4. Simulation of patient's view through the HMD for outdoor and indoor environments. A residual 10º field of view has been considered to simulate the tunnel vision effect.

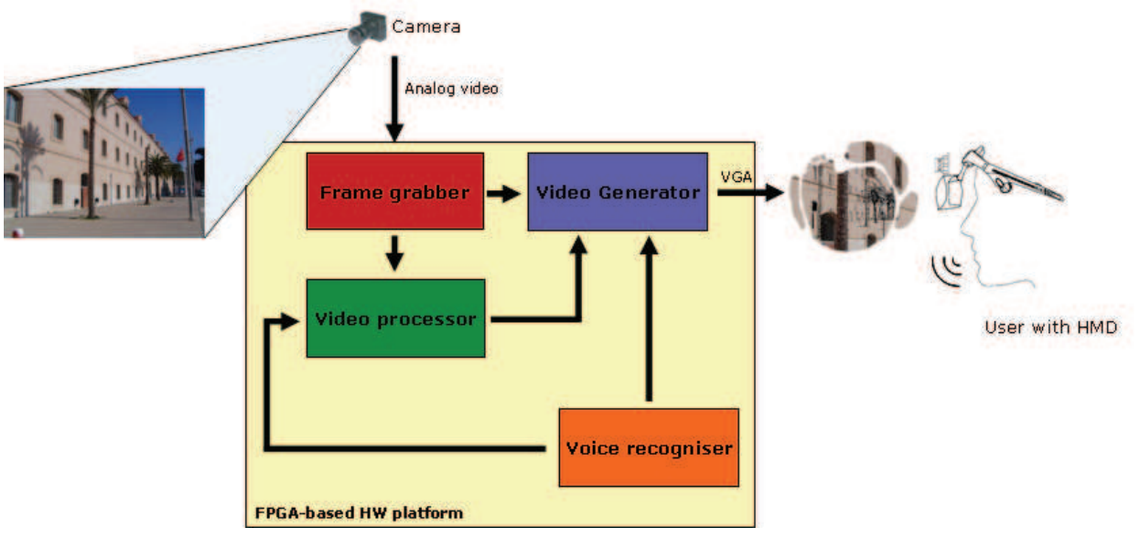


Fig. 5. Overall system schematic showing the main modules and the output view presented to the user.

In our system, the image acquired with the frame grabber is processed to extract contour information and it is used to enhance the user's perception of the environment by the superimposition on his own view of the entourage seen with a see-through head mounted display. To carry out the required processing we proposed the use of a Cellular Neural Network (CNN), which can be tuned to produce customized results and allows increasing the versatility of the system through the possibility of using different templates. The difficulties that rise when designing digital hardware implementation of CNNs are addressed in (Martínez et al, 2007; Martínez et al, 2008), where a novel approach is also proposed. It has been later optimized in (Martínez et al, 2009). After processed, the image from the camera must be properly zoomed out in order to be shown to the user in his residual central vision. A digital zooming algorithm has been included in the design with this purpose. It has been designed to minimize the number of access to the external memory where the original input data are stored.
The image resulting from the processing with the CNN, suitably minified, is sent to the VGA output available in the hardware platform. Fig. 4 shows some examples of the system output.

Due to the special characteristics of the target application, a vision-based interface such as the described in the previous section is not a suitable approach for interacting with this system. Instead, we have developed a voice recognition system which, as the whole of the herein described system, is implemented in FPGA. Since the aid device is conceived as a personal system, an easy and high-reliable speaker dependent recognition algorithm has been designed. In a simple and fast initial step, the user records a customized set of keywords. They are stored in external non-volatile flash memory, so the user only has to do it for the first time use. Later, in operating mode, the algorithm detects when the user says a word and convolves it with all the previously recorded ones. The latency response depends on the number of key words, but it is typically in the order of the milliseconds, fast enough to not appreciating any significant delay.

This user interface makes it feasible to adapt the functionality to the user preferences through, for example, basic modifications in the CNN processing, the minification factor or the colour and intensity of the contour information superimposed in his view.

The great amount of resources available in FPGA devices and their inherent parallelism make it possible to fulfill the requirements of the application without compromising the performance in speed, size and power consumption.

A simplified diagram of the whole system is shown in Fig. 5. The current prototype has been built with boards from Avnet, which populates Xilinx devices and the additional integrated circuits mentioned. The prototype also uses a Sony Glasstron PLMS700E as head mounted display to superimpose the video output on the user view. At the present moment, the system is under test and validation by visually impaired people, offering very successful initial results and improving the patients' ability to localize objects, orientate and navigate. Once passed the tests, a commercial device will be manufactured and packed into a small shoulder bag or belt bag.

8. References

Akin, A.; Dogan, Y. & Hamzaoglu, I. (2009). High performance hardware architectures for one bit transform based motion estimation, *IEEE Transactions on Consumer Electronics,* Vol. 55, No. 2, (May 2009), pp. (941-949), 0098-3063

Arnold, J.; Buell, D. & Hoang, D. (1993). The Splash 2 Processor and Applications, *Proc. of International conference on Computer Design*, pp. 482-485, 0-8186-4230-0, Los Alamitos, USA, 1993, IEEE

Assad, M.; Carmichael, D.; Cutting, D. & Hudson, A. (2003). AR phone: Accessible Augmented Reality in the Intelligent Environment, *Proc. of Australiasian Computer Human Interaction Conference*, pp. 232-235, Queensland, Australia, 2003

Bakker, J.; Langendoen, K. & Sips, H. (2001). LART: flexible, low-power building for wearable computers. *Proc. Int. Workshop on Smart Appliances and Wearable Computing*, pp. 255–259. Scottsdale, USA

Barfield, W. & Caudell, T. (2001). *Fundamentals of wearable computers and augmented reality,* Lawrence Erlbaum, 0805829016, USA

Black, D. & Donovan, J. (2004). *SystemC: From the Ground Up*. Kluwer Academic Publishers, ISBN: 978-0-387-29240-3

Billinghurst, M.; Hakkarainen, M. & Woodward, C. (2008). Augmented Assembly using a Mobile Phone, *Proc. of International Conference on Mobile and Ubiquitous Multimedia*, pp. 84-87, 978-1-60558-192-7, Umea, Sweden, december 2008, ACM

Bowen, O. & Bouganis, C. (2008). Real-Time Image Super Resolution Using an FPGA, *Proc. of International Conference on Programmable Logic and Applications*, pp. 89-94, 978-1-4244-1960-9, Heidelberg, Germany, September 2008, IEEE

Bodnar, M.; Curt, P.; Ortiz, F.; Carrano, C. & Kelmelis, E. (2009). An Embedded Processor for Real-Time Atmoshperic Compensation, *Proc. of SPIE conference on Visual Information Processing XVII*, pp. 1-8, Orlando USA, April 2009, SPIE

Dähne, P. & Kariginannis, J. (2002). Archeoguide: system architecture of a mobile outdoor AR system, *Proc. of International Symposium on Mixed and Augmented Reality*, pp. 263-264, 0-7695-1781-1, Darmstadt, Germany, September 2002, IEEE CS

Dellaert, F. & D. Tarip, S. (2005). A Multi-Camera Pose Tracker for Assisting the Visually Impaired, *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 31–39, 0-0-7695-2372-2, San Diego, USA, June 2002, IEEE

Dawood, A.; Visser, S. & Williams, J. (2002). Reconfigurable FPGAS for real time image processing in space. *Proc. IEEE Int. Conf. Digital Signal Processing*, pp. 845–848, ISBN: 0-7803-7503-3, Greece, July 2002

Densmore, D. & Passerone, R. (2006). A Platform-Based Taxonomy for ESL Design. *IEEEDesign & Test of Computers*, Vol. 23, No. 5, pp. 359–374, ISSN: 0740-7475

Design & Reuse (2009). *Design & Reuse*. Web page: www.design-reuse.com

Diaz, J.; Ros, E.; Pelayo, F.; Ortigosa, E. & Mota, S. (2006). FPGA-based real-time optical-flow system, *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 16, No. 2, (Feb 2009), pp. (274-279), 1051-8215

Drayer, T.; King, W.; Tront, J. & Conners, R . (1995). A Modular and Reprogrammable Real Time Processing Hardware, MORRPH, *Proc. of IEEE Symposium on FPGA's for Custom Computing Machines*, pp. 11–19, 0-8186-7086-X, Napa Valley, USA, 1995.

Enzler, R.; Platzner, M., Plessl, C., Thiele, L. & Tröster, G. (2001). Reconfigurable processors for handheld and wearables: application analysis. *Proc. SPIE Conf. on Reconfigurable Technology: FPGAs and Reconfigurable Processors for Computing and Communications,* vol. 4525.

Feiner, S.; MacIntyre, B., Höllerer, T. & Webster, A. (1997). A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. *Proc. of the First International Symposium on Wearable Computers (ISWC)*, pp. 74–81, Cambridge, Massachusetts, USA.

Foxlin, E; & Harrington, M. (2000) WearTrack: A Self-Referenced Head and Hand Tracker for Wearable Computers and Portable VR. *Proc. of the Fourth International Symposium on Wearable Computers (ISWC'00), pp.155.*

Garcia, R.; Batlle, J. & Bischoff, R. (1996). Architecture of an Object-Based Tracking System Using Colour Segmentation, *Proc. of International conference on Image and Signal Processing*, pp. 299-302, 0-4448-2587-8, Manchester, UK, November 1996, Elsevier Science

Guimarães, G.F.; Lima, J.P.S.M. & Teixeira, J.M.X.N., (2007). FPGA Infrastructure for the Development of Augmented Reality Applications *Proc. of the 20th annual conference on Integrated circuits and systems design*. pp. 336 – 341. Copacabana, Rio de Janeiro. ISBN:978-1-59593-816-9.

Hartenstein, R. (2002). Configware/Software Co-Design: Be Prepared for the Next Revolution, *Proc. of the 5th IEEE Workshop Design and Diagnostics of Electronic Circuits and Systems*, pp. 19–34, ISBN: 80-214-2094-4, Brno, Czech Republic, April, 2002.

Hartenstein, R. (2004). The digital divide of computing, *Proc. of the 1st conference on Computing frontiers*, pp. 357 – 362, ISBN:1-58113-741-9, Ichia, Italy, 2004, ACM, NY

Hoellerer, T.; Feiner, S., Terauchi, T., Rashid, G. & Hallaway, D. (1999) Exploring mars: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics 23* pp. 779–785.

Hosking, R.; (2008). Designing Software Radio Systems With FPGAs. www.pentek.com

Howes, L.; Price, P.; Mencer, O.; Beckmann, O. & Pell, O. (2006). Comparing FPGAs to Graphics Accelerators and the Playstation 2 Using a Unified Source Description, *Proc. of International Conference on Field Programmable Logic and Applications*, pp. 1-6, 1-4244-0312-X, Madrid, Spain, Aug 2006, IEEE

Hsiung, P.A.; Santambrogio, M.D. & Huang, C.H. (2009). *Reconfigurable System Design and Verification*. CRC Press, ISBN-13: 978-1420062663

In P.; Jung K.. & Kwang H. (2008). An Implementation of an FPGA-Based Embedded Gesture Recognizer Using a Data Glove, *Proc. of the 2nd International Conference on Ubiquitous Information Management and Communication*, pp. 496-500, 978-1-59593-993-7, Rennes, France, July 2008, ACM

ImpulseC (2009). *Impulse Accelerated Technologies*. Web page: www.impulseaccelerated.com

Jack, K. (2005). *Video Demystified,* LLH Technology, 2005, 1-878707-23-X, USA

Johnston C.; Gribbon, K. & Bailey, D. (2005). PGA Based Remote Object Tracking for Real-Time Control, *Proc. of International Conference on Sensing Technology*, pp. 66–72, Palmerston North, New Zealand, November 2005

Johnston, D.J.; Fleury, M., Downton, A.C. & Clark, A.F. (2005). Real-time positioning for augmented reality on a custom parallel machine. *Elsevier Image and Vision Computing,* No. 23 (2005) pp. 271-286, doi:10.1016/j.imavis.2003.08.002.

Keating, M. & Bricaud, P. (1999). *Reuse Methodology Manual*. Kluwer Academic Publishers, ISBN-13: 978-0792381754

Krips, M.; Lammert, T. & Kummert, A. (2002). FPGA Implementation of a Neural Network for a Real Time Hand Tracking System. *Proc. of IEEE International Workshop on Electronic Design, Test, Applications*. pp. 313–317, 0-7695-1453-7, Christchurch, New Zealand, 2002, IEEE CS

Krips, M.; Velten, J. & Kummert A. (2003). FPGA Based Real Time Hand Detection by Means of Colour Segmentation, *DOKLADY of Belarussian State University of Informatics and Radioelectronics*, (November 2003), pp. (156–162), 1729-7648

Ligocki, N.P.; Rettberg, A., Zanella, M., Hennig, A. & de Freitas, F.A.L., (2004). Towards a Modular Communication System for FPGAs, *Proc. of the Second IEEE International Workshop on Electronic Design, Test and Applications*, pp. 71.

Luk, W.; Lee, T.K., Rice, J.R., Shirazi, N. & Cheung P.Y.K. (1998). A Reconfigurable Engine for Real-Time Video Processing. *Proc. of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines* pp. 136. Springer-Verlag.

Luk, W.; Lee, T.K., Rice, J.R., Shirazi, N. & Cheung, P.Y.K. (1999). Reconfigurable Computing for Augmented Reality. FCCM archive. ISBN:0-7695-0375-6 IEEE Computer Society Washington, DC, USA.

Martín, J.; Zuloaga, A.; Cuadrado, C.; Lázaro, J. & Bidarte, U. (2005). Hardware Implementation of Optical Flow Constraint Equation Using FPGAs, *Computer Vision and Image Understanding,* Vol. 98, No. 3, (2005), pp. (462-490)

Martínez, J.J.; Toledo, F.J. & Ferrández, J.M. (2007). Discrete-Time Cellular Neural Networks in FPGA, *Proc. of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM07)* pp. 293-294, 0-7695-2940-2, Napa, CA, USA, april 2007, IEEE CS

Martínez, J.J.; Toledo, F.J.; Fernández, E. & Ferrández, J.M. (2008). A retinomorphic architecture based on discrete-time cellular neural networks using reconfigurable computing, *Neurocomputing,* vol. 71, 2008, pp. 766-775, 0925-2312.

Martínez, J.J.; Toledo, F.J.; Fernández, E. & Ferrández, J.M. (2009). Study of the contrast processing in the early visual system using a neuromorphic retinal architecture, *Neurocomputing,* vol. 72, 2009, pp. 928-935, 0925-2312.

Masselos, K.; & Voros N. S. (2007). Implementation of Wireless Communications Systems on FPGA-Based Platforms. Hindawi Publishing Corporation EURASIP Journal on Embedded Systems Volume 2007, Article ID 12192, 9 pages. doi:10.1155/2007/12192.

Matsushita, N.; Hihara, D., Ushiro, T., Yoshimura, S., Rekimoto, J. & Yamamoto, Y. (2003). ID CAM: A Smart Camera for Scene Capturing and ID Recognition. *Proc. of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)* pp. 227-236, Tokyo, Japan, October 7-10. IEEE CS, Los Amitos, CA.

Mitra, T. & Chiueh, T. (2002). An FPGA Implementation of Triangle Mesh Decompression, *Proc. of IEEE Symposium Symp. on Field–Programmable Custom Computing Machines*, pp. 22–31, 0-7695-1801-X, Napa, USA, April 2002, IEEE

Möhring, M.; Lessig, C. & Bimber, O. (2004). Video See-Through AR on Consumer Cell-Phones, *Proc. of IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 252-253, 0-7695-2191-6, Arlington, USA, november 2004, IEEE CS

Munteanu, D.; & Williamson, C. (2005). An FPGA-based Network Processor for IP Packet Compression. http://pages.cpsc.ucalgary.ca/~carey/papers/2005/FPGA.pdf

Olivares, J.; Benavides, I.; Hormigo, J.; Villalba, J. & Zapata, E. (2006). Fast Full-Search Block Matching Algorithm Motion Estimation Alternatives in FPGA, *Proc. of International Conference on Field Programmable Logic and Applications*, pp. 1-4, 1-4244-0312-X, Madrid, Spain, Aug 2006, IEEE

OpenCores (2009). Opencores. Web page: www.opencores.org

Ortiz, F.; Kelmelis, E. & Arce G. (2007). An Architecture for the Efficient Implementation of Compressive Sampling Reconstruction Algorithms in Reconfigurable Hardware, *Proc. of SPIE conference on Visual Information Processing XVI*, pp. 1-11, 0-8194-6697-2, Orlando USA, April 2007, SPIE

OSCI. (2009). *Open SystemC Initiative*. Web page: www.systemc.org

Passman, W. & Woodward, C. (2003). Implementation of an Augmented Reality System on a PDA, *Proc. of IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 276-277, 0-7695-2006-5, Tokyo, Japan, october 2003, IEEE CS

Piekarski, W.; & Thomas, B.H. (2001). Tinmith-evo5: A software architecture for supporting research into outdoor augmented reality environments, *Technical Report, Wearable Computer Laboratory*, University of South Australia.

Piekarski, W.; Smith, R.; Wigley, G.; Thomas, B. & Kearney D. (2004). Mobile hand tracking using FPGAs for low powered augmented reality, *Proc. of 8th IEEE Int. Symposium on Wearable Computers (ISWC04)* pp. 190-191, 0-7695-2186-X, Arlington, VA, USA, november 2004, USA

Plessl, C.; Enzler, R., Walder, H., Beutel, J., Platzner, M., Thiele, L. & Tröster, G. (2003). The case for reconfigurable hardware in wearable computing. Personal and Ubiquitous Computing, vol.7, no. 5, pp. 299–308. Springer-Verlag.

Riess, P. & Stricker, D. (2006). AR on-demand: a practicable solution for augmented reality on low-end handheld devices, *Proc. of AR/VR Workshop of the German Computer Science Society*, pp. 119-130, 3-8322-5474-9, Coblence, Germany, 2006

Schelle, G.; & Grunwald, D. (2008). Exploring FPGA network on chip implementations across various application and network loads. *Proc. of the Field Programmable Logic and Applications (FPL '*08) . pp. 41-46. Sept. 2008. ISBN: 978-1-4244-1960-9 DOI: 10.1109/FPL.2008.4629905.

Singh, S. & Bellec, P. (1994). Virtual Hardware for Graphics Applications Using FPGAs, *Proc. of IEEE Workshop FPGAs for Custom Computing Machines*, pp. 49–58, 0-8186-5490-2, Napa, USA, April 1994

Smith, R.; Piekarski, W. & Wigley, G. (2005). Hand Tracking for Low Powered Mobile AR User Interfaces, *Proc. of the Sixth Australasian User Interface Conference (AUIC '05)* pp. 7-16, Newcastle, Australia, January 31st – February 3rd, 2005, Australian CS, Sydney, NSW.

Stitt, G.; Grattan, B., Villarreal, J. & Vahid, F. (2002). Using on-chip configurable logic to reduce embedded system software energy. *Proc. IEEE Symp. on Field–Programmable Custom Computing Machines*, pp. 143–151. Napa, USA.

Styles, H. & Luk, W. (2000). Customising Graphics Applications: Techniques and Programming Interface, *Proc. of IEEE Symposium on Field–Programmable Custom Computing Machines,* pp. 77–90, 0-7695-0871-5, Napa, USA, 2000

Styles, H. & Luk, W. (2002). Accelerating radiosity calculations using reconfigurable platforms, *Proc. of IEEE Symposium on Field–Programmable Custom Computing Machines*, pp. 279–281, 0-7695-1801-X, Napa, USA, September 2002, IEEE

Takayuki, S.N. & Aizawa, K. (2002). Real time Objects Tracking by Using Smart Image Sensor and FPGA, *Proc. of International Conference on Image Processing*, pp. 441–444, 0-7803-7622-6, Rochester, New York, 2002, IEEE

Tanase, C.; Vatavu, R., Pentiuc, S. & Graur, A. (2008). Detecting and Tracking Multiple Users in the Proximity of Interactive Tabletops, *Advances in Electrical and Computer Engineering,* Vol. 8, No. 2, (2008), pp. (61-64)

Thomas D.; (2006). FPGA Network Firewalling. Report of the Diadem Firewall Project. *www.doc.ic.ac.uk/~dt10/public/diadem.ppt*.

Toledo, F.J.; Martínez, J.J. & Ferrández, J.M. (2007). Hand-based Interface for Augmented Reality, *Proc. of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM07)* pp. 291-292, 0-7695-2940-2, Napa, CA, USA, april 2007, IEEE CS

Toledo, F.J.; Martinez, J.J., Garrigos, J. & Ferrandez, J. (2005). FPGA Implementation of an Augmented Reality Application for Visually Impaired People. *Proc. of the Fifteenth International Conference on Field Programmable Logic and Applications (FPL '05)* . pp 723-724. Tampere, Finland, August 24-26, 2005. IEEE CS, Los Amitos, CA.

Toledo, F.J.; Martínez, J.J.; Garrigós, F.J., Ferrández, J.M. & Rodellar, V. (2006). Skin color detection for real time mobile applications, *Proc. of Int. Conference on Field Programmable Logic and Applications (FPL06)* pp. 271-274, 1-4244-0312-X, Madrid, Spain, august 2006

Vargas-Martín, F. & Peli, E. (2001). Augmented view for tunnel vision: device testing by patients in real environments, *Digest of Technical Papers, Society for Information Display International Symposium* pp. 602-605, San José, CA, USA, 2001

Vargas-Martín, F. & Peli, E. (2002). Augmented-view for restricted visual field: multiple device implementations, *Optometry and Vision Science,* vol. 79, no. 11, 2002, pp. 715-723, 1040-5488

Veas, R. & Kruijff, E. (2008). Vesp'R: design and evaluation of a handheld AR device, *Proc. of IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 43-52, Cambridge, UK, September 2008, IEEE

Vicente, A.; Munoz, I.B.; Molina, P.J. & Galilea, J.L.L. (2009). Embedded Vision Modules for Tracking and Counting People, *IEEE Transaction on Instrumentation and Measurement*, Vol. 58, No. 0, (September 2009), pp. (3004-3011), 0018-9456

Villasenor, J.; Shoner, B.; Chia, K.N.; Zapata, C.; Kim, H.J.; Jones, C.; Lansing, S. & Mangione-Smith, B. (1996). Configurable Computing Solutions for Automatic Target Recognition, *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, pp. 70–79, ISBN: 0-8186-7548-9, Napa, USA, 1996

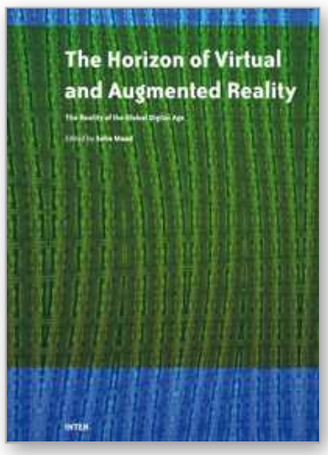
Wagner, D.; Pintaric, T., Ledermann, F. & Schmalstieg, D. (2005). Towards Massively Multi-user Augmented Reality on Handheld Devices. *Proc. of the Third International Conference on Pervasive Computing (PERVASIVE '05)* pp. 208-219, Munich, Germany, May 8-13, 2005, Springer Berlin/Heidelberg, New York, NY.

Wee, C.M.; Sutton, P.R. & Bergmann, N.W. (2005). An FPGA network architecture for accelerating 3DES - CBC. *Proc. of the IEEE 15th International Conference on Field Programmable Logic and Applications (FPL),* Tampere, Finland, pp. 654-657. August 2005. ISBN: 0-7803-9362-7

Ye, A. & Lewis, D. (1999). Procedural Texture Mapping on FPGAs, *Proc. of the 1999 ACM/SIGDA Seventh International Symposium on Field-Programmable Gate Arrays*, pp. 112 – 120, 1-58113-088-0, Monterey, USA, 1999, ACM

Yu, N.; Kim, K. & Salcic, Z. (2004). A new motion estimation algorithm for mobile real-time video and its FPGA implementation, *Proc. of IEEE Region 10 Conference: Analog and Digital Techniques in Electrical Engineering*, pp. 383-386, 0-7803-8560-8, Chiang Mai, Thailand, November 2004, IEEE

Zhou, F.; Been-lirn Duh, H. & Billinghurst, M. (2008). Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR, *Proc. of IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 193-202, Cambridge, UK, September 2008, IEEE



Augmented Reality
Edited by Soha Maad

ISBN 978-953-7619-69-5
Hard cover, 230 pages
Publisher InTech
Published online 01, January, 2010
Published in print edition January, 2010

Virtual Reality (VR) and Augmented Reality (AR) tools and techniques supply virtual environments that have key characteristics in common with our physical environment. Viewing and interacting with 3D objects is closer to reality than abstract mathematical and 2D approaches. Augmented Reality (AR) technology, a more expansive form of VR is emerging as a cutting-edge technology that integrates images of virtual objects into a real world. In that respect Virtual and Augmented reality can potentially serve two objectives: reflecting realism through a closer correspondence with real experience, and extending the power of computer-based technology to better reflect abstract experience. With the growing amount of digital data that can be stored and accessed there is a rising need to harness this data and transform it into an engine capable of developing our view and perception of the world and of boosting the economic activity across domain verticals. Graphs, pie charts and spreadsheet are not anymore the unique medium to convey the world. Advanced interactive patterns of visualization and representations are emerging as a viable alternative with the latest advances in emerging technologies such as AR and VR. And the potential and rewards are tremendous. This book discusses the opportunities and challenges facing the development of this technology.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

J. Toledo, J. J. Martínez, J. Garrigós, R. Toledo-Moreo and J. M. Ferrández (2010). Design of Embedded Augmented Reality Systems, Augmented Reality, Soha Maad (Ed.), ISBN: 978-953-7619-69-5, InTech, Available from: http://www.intechopen.com/books/augmented-reality/design-of-embedded-augmented-reality-systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen