

# Towards Wearable Virtual Reality System Using Micro IMU Controller and FPGA Platform

Wen-Xian Wu, Yan Shao, and Quan-Biao Chen

*School of Software and Microelectronics at Wuxi, Peking University*

wxwu@pku.edu.cn

1101220661@pub.ss.pku.edu.cn

**Abstract**—Virtual Reality (VR) is a computer-simulated environment that can simulate physical presence in places in the real world or imaginary world. We use FPGA to implement this design to take advantage of its hardware/software co-design and its high computation speed. We use IMU as the main input. It captures and sends motion data to FPGA. FPGA plays an important role in this system. It drives hardware, processes data, and stores and manages pre-stored image or sound material. After processing, system outputs image and voice, and imports into media glass and speaker, which are used as output devices, respectively. The final result shows that our system designed with FPGA can run fast and smoothly, easy to maintain and update, and it shows great potential.

**Keywords**—FPGA; co-design; virtual reality; sensor; media glasses

## I. INTRODUCTION

Virtual Reality (VR) is a computer-simulated environment that can simulate physical presence in places in the real world or imaginary world, provides users with visual, auditory, tactile and other sensory simulation. Some advanced, haptic systems now include tactile information, generally known as force feedback, in medical and gaming applications.

Virtual reality is often used to describe a wide variety of applications commonly associated with immersive, highly visual, 3D environments. So it can be used in entertainment, training or other kinds of special applications.

There are two kinds of design trends of VR system. One is upsizing design. This kind of design can provides precise simulation of the real world to create a lifelike experience, for example, in simulations for pilot or combat training. But the equipments of this design are usually heavy and expensive. The other kind of design is miniaturization. It reduces the size of equipments at the expense of performance and response time.

In our design, we aim at building a small size wearable, quick responding VR system. The system uses inertial measurement unit (IMU) to acquire motion signal. And it is based on FPGA. It is a programmable system that combines the advantages of hardware and software to work together, achieves integration of upper task and underlying hardware, and reduces the difficulty of development. It uses hardware acceleration's core, which can independently work as a CPU, and that makes it possible to transmit and process large amount of data. In addition, FPGA has high computation

speed which can provide smooth user experience, low power consumption, and high level of integration. Therefore, we take FPGA as the core of our design.<sup>[1]</sup>

In this paper, we use a micro IMU as controller, and DE2-115 as our design platform. We use Bluetooth to communicate between nodes, and use SOPC design to build the main control unit.

## II. HARDWARE

To build a complete VR system, there are several components we need: sensor, controller, DE2-115, M92 media glass and speaker. Their relationships are shown in Figure 1.

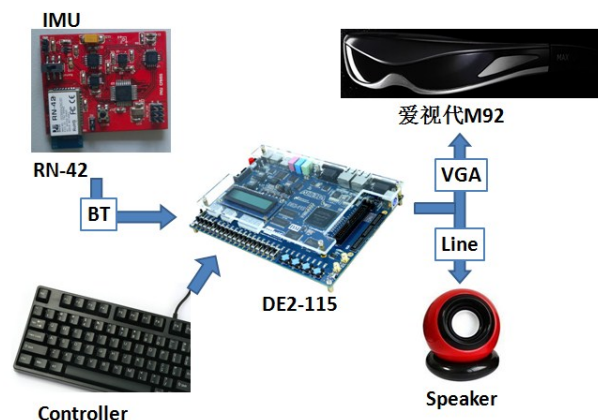


Figure 1 Relationships between components

1) *Sensor*. In our design, we use a homemade inertial measurement unit (IMU) as the main input. It is an electronic device that measures and reports on a human's velocity, orientation, and gravitational forces, using a combination of accelerometers (LPR550AL) and gyroscopes (ADX1335). Once the human's status changes, the IMU will detect this change and transmit corresponding signal to FPGA in the form of ADC signal.

2) *RN-42*. It is a Bluetooth module, which can provide wireless communication between sensor and FPGA, and send data into it.

3) *Controller*. A keyboard is used to work as auxiliary input. In this shooting game, we use sensor to control aim point, and use keyboard to control character's motion.

4) *DE2-115*. It is the core of our design. It provides all drivers that we need, like VGA, sound, SD card and FAT file system driver which designed for SD card read and write operation. Also it is main processor in the system which calculates raw signal it received from sensor, and send processed output data into display device.

5) *M92 media glass and speaker*. The glass and speaker are used to provide users immersion game experience.

In order to achieve the showing effect, we transplant a 3D vision software into the system. Since we have introduced the hardware, then the whole system's structure can be built. It is a three-layer structure, as shown in Figure 2.<sup>[2]</sup>

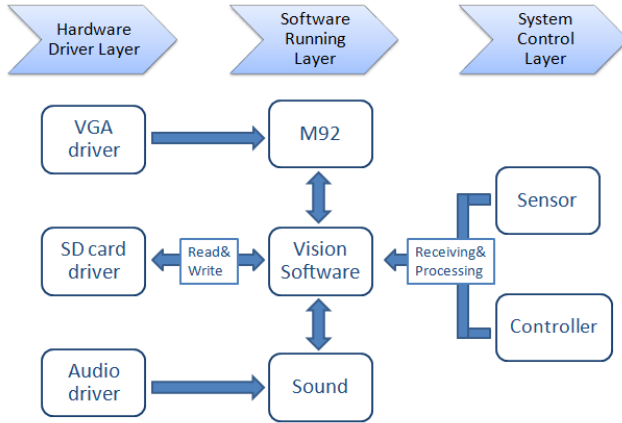


Figure 2 Whole system's structure.

### III. SYSTEM

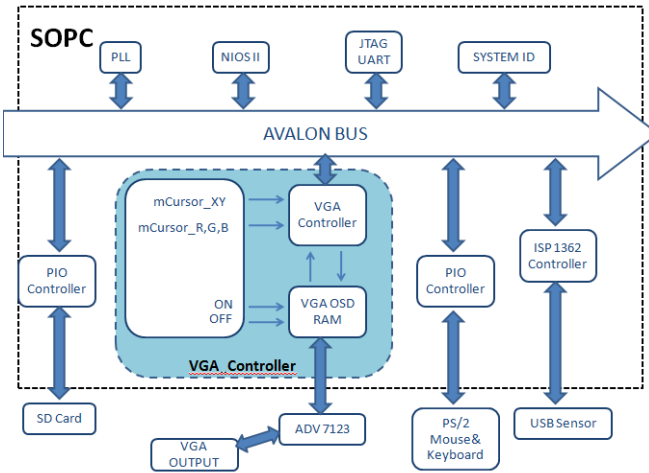


Figure 3 Vector in coordinate axis

In three-dimensional space, we can describe a force in the form of vector in coordinate axis, as is shown in figure 3. This R represents force vector, which is gravity or a combination of inertial forces, and can be detected by accelerometers (LPR550AL). Once a certain force acts on the sensor, the vector is certain, and the components of three axis are certain. The accelerometer detects these vector components, and output corresponding ADC signal, which we define as  $AdcR_x$ ,  $AdcR_y$ , and  $AdcR_z$ . Then we can use formula 1 to calculate  $R_x$ ,  $R_y$ ,  $R_z$ .

$$\begin{aligned} R_x &= (AdcR_x * V_{REF} / 1023 - V_{ZERO}G) / Sensitivity \\ R_y &= (AdcR_y * V_{REF} / 1023 - V_{ZERO}G) / Sensitivity \\ R_z &= (AdcR_z * V_{REF} / 1023 - V_{ZERO}G) / Sensitivity \end{aligned} \quad (1)$$

In this formula,  $V_{ref}$  is reference voltage. It is 3.3V in our design.  $V_{zero}G$  is zero voltage, which means the output voltage without force acts, and is detected beforehand. Sensitivity describes the change rate of voltage with the change of force. And it is 300mV/g.

However the result we get from accelerometer is not preciseness, because accelerometer is sensitive to vibration and mechanical noise. To solve this problem, we introduce a gyroscope. The gyroscope detects change rate of  $AXZ$ ,  $AYZ$ , and output corresponding ADC signal. We can calculate the rate by using formula 2.

$$\begin{aligned} RateA_{XZ} &= (AdcGyro_{XZ} * V_{REF} / 1023 - V_{ZERO}Rate) / Sensitivity \\ RateA_{YZ} &= (AdcGyro_{YZ} * V_{REF} / 1023 - V_{ZERO}Rate) / Sensitivity \end{aligned} \quad (2)$$

#### A. System Control

In our design, we try to build up a complete VR system, which means we should precisely acquire the motion signal. And the main part of this layer is sensor. Our sensor is a homemade inertial measurement unit, a combination of accelerometers (LPR550AL) and gyroscopes (ADXL335).

Since we get output, now we should combine the results. At first, we use variables  $R_{acc}$  to describe the results we acquire from accelerometer and  $R_{gyro}$  to describe the results we get from gyroscope. We detect every T seconds and store accelerometer's result as  $R_{acc}(0)$ ,  $R_{acc}(1)$ ,  $R_{acc}(2)$ ,  $R_{acc}(3)$ ,  $R_{acc}(4)$ ,  $R_{acc}(5)$ ,  $R_{acc}(6)$ ,  $R_{acc}(7)$ ,  $R_{acc}(8)$ ,  $R_{acc}(9)$ ,  $R_{acc}(10)$ ,  $R_{acc}(11)$ ,  $R_{acc}(12)$ ,  $R_{acc}(13)$ ,  $R_{acc}(14)$ ,  $R_{acc}(15)$ .

Racc(2), Rate(2) and so on. We introduce a new variable Rest to represent the calculated results. Assume accelerometer's result is right, and Rest(0)=Racc(0). Then we can acquire a series of Rest, like Rest(1), Rest(2), Rest(3). After n times of detect, we can get Racc(n), Rate(n) and Rest(n-1). To calculate the Rest(n), we should first calculate the Rgyro(n). We can use the following formulas to achieve that.

$$A_{xz}(n-1) = \arctan(\text{Rest}_x(n-1), \text{Rest}_z(n-1)) \quad (3)$$

$$A_{xz}(n) = A_{xz}(n-1) + \frac{\text{Rate}A_{xz}(n-1) + \text{Rate}A_{xz}(n) * T}{2}$$

Since we get the angles, we can then acquire the Rgyro's components.

$$R_{xgyro} = \frac{1}{\sqrt{1 + \cot(A_{xz}(n))^2 * \sec(A_{yz}(n))^2}} \quad (4)$$

$$R_{ygyro} = \frac{1}{\sqrt{1 + \cot(A_{yz}(n))^2 * \sec(A_{xz}(n))^2}}$$

$$R_{zgyro} = \text{Sign}(R_{zgyro}) * \frac{1}{\sqrt{1 - R_{xgyro}^2 - R_{ygyro}^2}}$$

Where Sign(RZgyro) is used to describe RZgyro's signal. When RZgyro ≥ 0, Sign(RZgyro)=1, and when RZgyro < 0, Sign(RZgyro)=-1.

The next step is calculating Rest(n), as is shown in formula 6.  $\partial$  is a variable that describe the degree we trust of accelerometer and gyroscope, and is ordinary set as 5 to 20.

$$\text{Rest}(n) = \frac{\text{Racc}(n) + \text{Rgyro}(n) * \partial}{1 + \partial} \quad (5)$$

And the last step is calculating angles.

$$A_{xr} = \arccos(\text{Rest}_x / \text{Rest}) \quad (6)$$

$$A_{yr} = \arccos(\text{Rest}_y / \text{Rest})$$

$$A_{zr} = \arccos(\text{Rest}_z / \text{Rest})$$

After that, we can start the next round of detect and calculation.

### B. Hardware Driver

The chief role of hardware driver layer is to enable our software system to invoke hardware components in FPGA, exchange data, and then send out data through different input or output ports. There are four main ports: SD card read & write port, VGA output port, sensor input port and controller input port.

1) *SD card read & write port.* Since the flash memory space of DE2-115 is only 8MB, which is too small to support a 3D software that has a large amount of pictures. SD card is then used to store and invoke programs we need. NIOS II software structure is based on a hardware abstraction layer (HAL), which provides NIOS II software developers, ports, application programmatic interface (API) and the standard C

library, as is shown in figure 4. The bottom of the SD card hardware connects with NIOS II system through Avalon bus.

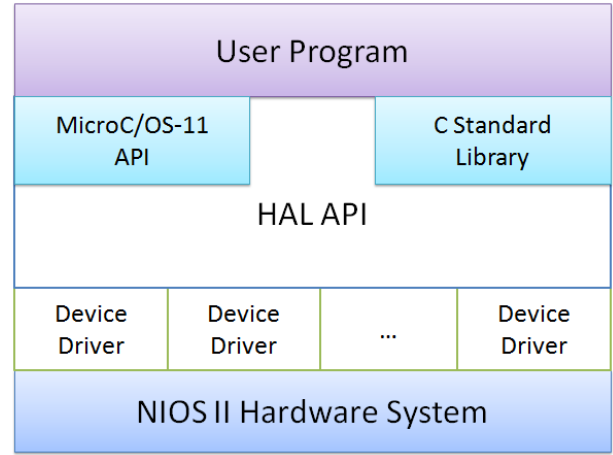


Figure 4 System structure [4]

To enable SD card read & write processes and manage data, we should effectively organize data in SD card, store and access in the form of files. Here we use the common used FAT file system. The application layer functions we use to invoke bottom functions are: FAT Init, FAT GetSize, FAT Open File, FAT Read File and so on. Then we can operate SD card by just invoking functions in top main.C file in NIOS II system.

2) *VGA Driver.* When the program runs, it outputs vision signal. The media glasses are used to show out this vision signal. The glasses support signal in the form of AV/CVBS/COMPONENT/VGA with resolution 640\*480. Considering about this resolution, we set these IP cores' parameters as figure 5.

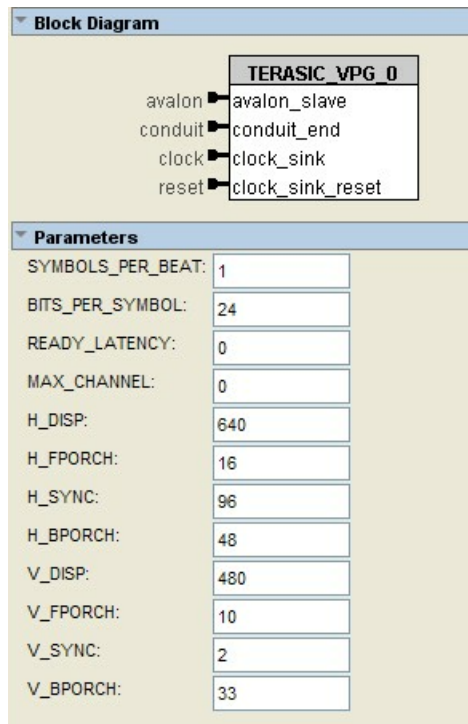


Figure 5 parameters of IP cores

Similar to the SD card driver, the hardware of VGA driver layer connects to Avalon bus by using Qsys. The Binary VGA Controller IP core is used to generate VGA control signal. VGA\_NIOS\_CTRL is the top module of the IP core; VGA\_Controller provides horizontal and vertical timing synchronizing signal to VGA interface; And has a Img RAM module which plays the role of Flame Buffer. VGA\_OSD\_RAM controls Flame Buffer to read or write. All these IP cores are generated by SOPC Builder, and can be invoked only after Qsys upgrading.

Also BSP in NIOS II can auto create HAL that corresponding to VGA driver, and can connect and be invoked like SD card part.

3) *USB Interface*. USB port is used in our design to receive data that detected by sensor. DE2-115 has integrated the Philips USB port chip, which provides USB host and USB device functions. So we just need to generate ISP1362 control signal, and set DE2-115 as hostport. And the control port is set as figure 6.

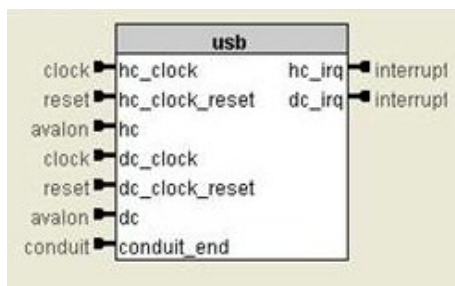


Figure 6 control port design.

### C. Vision Software Running

We have introduced system controlling and hardware driving layers. The software is based on these two layers. To achieve a good show effect, we transplant a general public license program named Stellarium into our system. This is a star field simulation rendered by openCV and coded by C++. As we try to complete this transplantation and generate NIOS II application, we found that it is very hard to direct run this software. So we implement the task by three steps.

1) *Transplanting Operation system*. The uClinux system is transplanted to provide necessary operation environment. Then we can set C/C++ Build and General environment, and add OpenCV interlinking library to make sure the software is running properly.

2) *Modifying OpenCV source code*. This task aims at modifying the software, to joint software and hardware. And it can also be decomposed into three parts:

- Reading & writing files and initialized setting, main of which is reading SD card files and write them into cache;
- Multi-input. The input devices are sensor, mouse and keyboard. We receive signals from these devices respectively, and convert them to signals that software can identify by modifying key values.
- Rewrite graphic output functions, main of which are VGA interlinks, to improve the quality of the graphics output.

3) *Setting and compiling software*. This task is mainly about generating statistic interlinking library of target platform, building project, and at last run/debug.

In this way, we successfully create the program and achieve the desired effect.

## IV. CONCLUSIONS

In this paper, we introduce our design ideas of the virtual reality system, implementation steps, and the transplantation steps. At last we achieve a complete VR vision system. Its acquisition accuracy is 90%. All these results are based on the extraordinary performance of FPGA. Its hardware and software co-design, integration layout, and fast computational speed have all contributed to the building of this VR system. This design takes full use advantages of FPGA, and show the great characteristics and potential. What FPGA can do is not only to support a simple vision software, but to play a big role in virtual reality entertainment and applications.

## REFERENCES

- [1] www.altera.com, Cyclone FPGA family, NIOS software CPU
- [2] Lin Xinming, Wang Chongsen, Wang Zhiheng. "Design and Development of 3DMotion Sensing Gaming Platform Using FPGA." The 1st Asia-Pacific Workshop on FPGA Applications, Xiamen, China, 2012.
- [3] Network. "Accelerometers and gyroscope." <http://www.docin.com/p-530688750.html>.
- [4] Chen Xia, Li Kaihang. "Design of SD card file system based on SoPC and NIOS II" Modern Electronics Technique, vol.35 No.16 2012

- [5] Wu Nian-xiang, Xie Fa-zhong. "The design of SOPC embedded system Based on FPGA". Science and technology of West China, vol.08, No.08, 2009.
- [6] Zhang zhen, Li lei. "Research on embedded system based on SOPC" Information Technology, No.12, 2007.
- [7] Yang Yang, Ye Peng, Li Li. "Design and implementation of UART based on FPGA" Electronic Measurement Technology, Vol.34,No.7 2011.
- [8] Du Peng. "Nios II software based SD memory card interface design" Computer Technology and Development <http://www.21ic.com/app/embed/201305/180817.htm>
- [9] Sun Hang. "The design of the serial communicational module on FPGA based on NIOS II" Microcomputer Information, Vol.24, No.10-2, 2010.
- [10] Jiang Shijie, Yu Hongying, Hong Yongxue, Lin Lirong. "FPGA implementation of VGA interface" Electronic Test, No.12, 2012.
- [11] Zhu Wenwei, Xu Zhongren, "Design and Implement ofVGA graphic controller based on FPGA" Journal of Guizhou University, Vol.26,No.2,2009.
- [12] Chen Yaojie, Lu Jianhua. "Research and Design of VGA interface based on FPGA" Journal of Transport Information and safety. No.2,2005