Third Prize

Omnidirectional Mobile Home Care Robot

Institution: **Department of Electrical Engineering, National Chung-Hsing University**

Participants: **Hsu-Chih Huang, Chia-Ming Chen, and Tung-Sheng Wang**

Instructor: **Professer Ching-Chih Tsai**

Design Introduction

In modern manufacturing, robot arms are used in automated production, replacing humans to avoid risk, increase efficiency, and improve accuracy. Fixed robot arm technology is very well-developed worldwide. In our project, we designed a smart robot to expand the applications for which robot arms are traditionally used. The robot arm is installed on an omnidirectional mobile vehicle; a network camera is installed on the front of the arm.

Our project, the omnidirectional mobile home-care robot, provides home care for the disabled and improves their quality of life. Because of physical disabilities, a disabled person cannot move and reach for a desired item as a non-disabled person would. The smart robot can perform these tasks with the help of its network camera, the platform, and the robot arm. With our system, the user grabs the article's image in the camera and the system directs the platform and arm to pick up the object. The robot helps the disabled person pick up objects and perform household tasks.

Figure 1 shows the omnidirectional mobile home care robot.

Figure 1. Omnidirectional Mobile Home Care Robot

The robot arm and the omnidirectional mobile vehicle system uses two Altera® Nios® II test boards. The Stratix® and Stratix II devices control the system, motor position, and speed. A PC performs image processing. We used a Nios II embedded processor to implement the system design. The Nios II processor has a consistent instruction set and programmable module, and can cooperate with assorted peripheral devices to transfer information, enabling joint use of the client's instruction and hardware acceleration unit. The Nios II processor provided excellent performance.

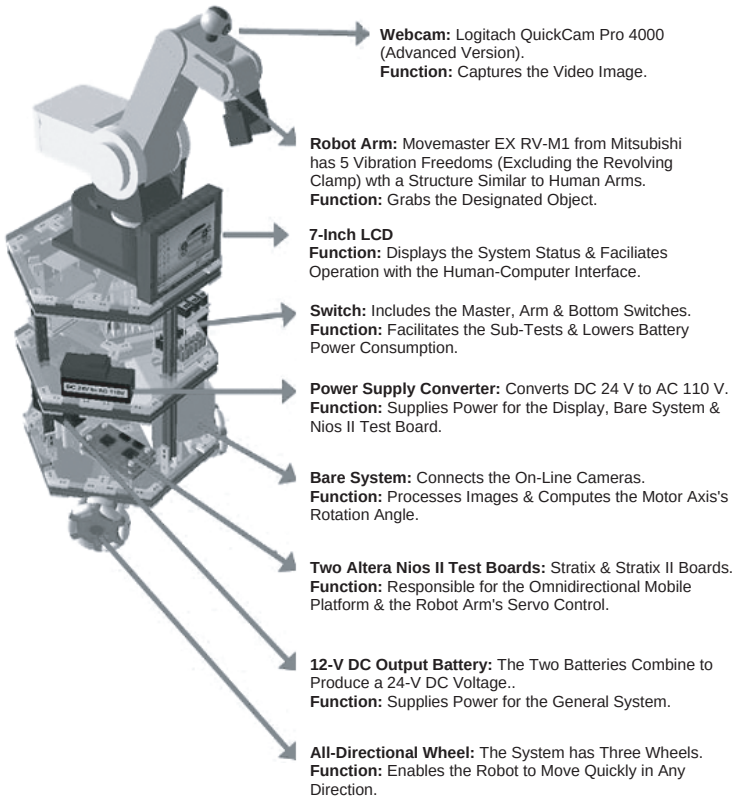
Unlike ordinary chips, the Nios II processor's peripheral I/O provides an I/O interface Ethernet port for communication. We implemented communication between the Nios II processor and the PC via the TCP interface, greatly improving integration of the robot's subsystems.

Programmable logic devices (PLDs) enhance the Nios II processor's capabilities. The robot arm has five servomotors and the platform has three. The hardware design control circuit architecture is so complex that ordinary digital signal processing (DSP) chips cannot meet the system's requirements. We used the Nios II processor and system-on-a-programmable-chip (SOPC) design techniques to create the smart robot hardware circuit. With the Nios II processor's calculation capability and the Quartus® II development software, we could implement the overall system using the Altera Stratix II and Stratix development boards.

Function Description

The omnidirectional mobile home care robot uses two Altera development boards; one controls the arm and the other controls the platform. The PC processes the image and calculates the object's position. Figure 2 shows the overall system architecture.

Figure 2. System Architecture



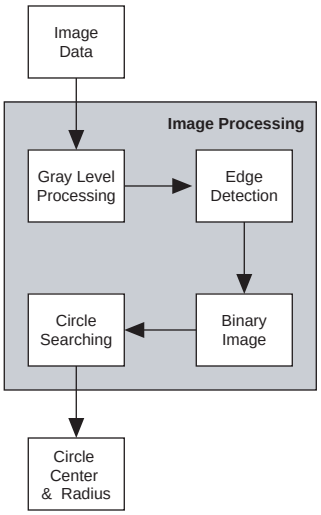
A PC controls the image processing and genetic algorithm and the Nios II processor (running on two development boards) controls the robot arm and the platform. The Nios II processor performs all control algorithms. Our test results show that the Nios II processor can act as the servo controller, while combining the hardware circuit and the software control program code in an FPGA accelerates research and development (R&D). Altera's Nios II processor performs important tasks and the development boards help us perform R&D.

The development board and PC designs are described as follows.

- *PC-based image processing core*—The user uses the webcam network camera to select image information, which is delivered to the PC for coordinate processing. Figure 3 shows the processing flow. We used Borland's C++ Builder to create the image processing design, and used the video for Windows (VFW) software development kit (SDK) to implement Gray-level processing, edge detection, and binary imaging. The edge detection design uses the Sobel algorithm and the binary imaging boundary values are determined using the Otsu algorithm. When image processing finishes, the system determines the object's three-dimensional (3-D) coordinates compared to the robot's. Then, the system substitutes the coordinates in the genetic algorithm to obtain the rotation angle of the robot arm's five-axis motor as well as the platform's best rotating X and Y coordinates. The system transmits the control signal to the hub via TCP and

Winsockets and delivers the rotation angle and the XY coordinates to the development boards, which control the speed and position of the robot arm and platform.

Figure 3. Image Processing Procedure



- *Stratix II development board*—The board receives the five-axis motor's rotation angle from the PC. It uses the built-in lightweight IP (LWIP) interface to connect to the PC, reducing the time spent on R&D. After the system determines the rotation angle for each axis of the robot arm, the board smoothly moves the arm to its destination. The Nios II CPU calculates the trajectory. A photo encoder installed on the arm motor gives the motor position (we wrote this circuit in VHDL). The Nios II CPU controls the motor position and speed with a PI controller. For the robot arm, we use a double PI control system to achieve the best performance. Finally, we used pulse width modulation (PWM) to output the control command to the motor driver. We wrote the PWM circuit in VHDL, which is implemented in the FPGA. The arm motor control circuit hardware and the software are integrated onto the Stratix II development board. This imeplementation greatly reduces the circuit size, provides design flexibility, and gives excellent performance.

- *Stratix development board*—The board receives the XY coordinates from the PC. The Nios II CPU, implemented on the board, controls the robot arm and the platform's point-to-point motion. The programmable I/O (PIO) outputs the digital control signal to the digital-to-analog converter (DAC), which converts it into an analog signal. The DAC outputs the analog signal to drive the platform's motor. The Nios II CPU integrates the circuit with the FPGA to control the platform.

Performance Parameters

Our system performs as described below.

- *Image processing and genetic algorithm*—During image processing, the system determines the object's position. The genetic algorithm determines how to control the arm and the platform compared to the object. We created the design using the Borland C++ Builder development interface. The system displays the results on screen, which makes the interface work more smoothly. In the future, we plan to implement these functions in the Nios II processor.

- *Communication between the PC and Nios II processor*—Our design uses one PC and two Altera development boards. Therefore, communication between the PC and the Nios II processors is very important. The LWIP interface, embedded in the Nios II processor, is the TCP interface for the system. On the PC side, we used a socket application programming interface (API) to write the TCP communication program, which achieves stable and fast information transfers. Based on our

test results, using the Nios II processor helped us to achieve our design goal for this part of the design.

- *Robot arm subsystem control*—The subsystem design integrates hardware and software in the Altera Stratix II development board. The design performs as follows:
  - *Trapezoidal route planning*—The built-in LWIP interface effectively receives control commands from the PC. The Nios II CPU calculates the trapezoidal route planning to facilitate smooth robot arm movement. Our tests show that the Nios II CPU has sufficient performance to ensure smooth arm movements.
  - *Five-axis servo motor control*—After performing route planning, the Nios II CPU propels the motors on each axis to rotate the arm. The Nios II CPU calculates the PI motor position and speed control. The Nios II processor is useful for this part of the design.
  - *PWM output, servo motor drive*—The development board's PIO drives the arm, which grasps the object. This test was successful.
- *Subsystem platform control*—The subsystem design integrates hardware and software on the Altera Stratix development board. The design performs as follows:
  - *Point-to-point algorithm*—The built-in LWIP interface receives the platform control command from the PC. Then Nios II CPU calculates the point-to-point algorithm, so that the platform can move to the object's correct position. Our test showed that Nios II calculation helps the robot make a successful point-to-point motion to reach the object.
  - *Three-axis servo motor control*—The platform is controlled by three servo motors that use a PI control rule. According to our tests, the Nios II processor does well for both position and speed control.
  - *DAC*—The Stratix development board PIO outputs a 12-bit digital control signal, and converts it in the DAC into an analog signal that controls the platform's servo motor. Our testing shows that the digital analog conversion is accurate.

Design Architecture

This section describes our design architecture. Figures 4 and 5 show the hardware design block diagrams for the arm and mobile platform, respectively.

Figure 4. Arm System Hardware Block Diagram

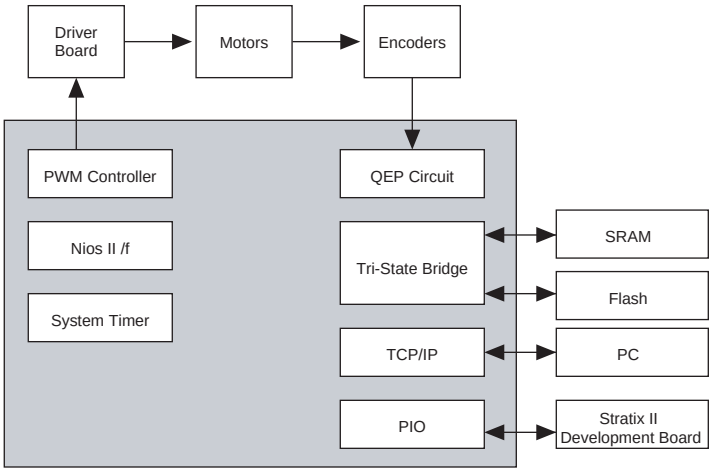


Figure 5. Mobile Platform System Hardware Block Diagram

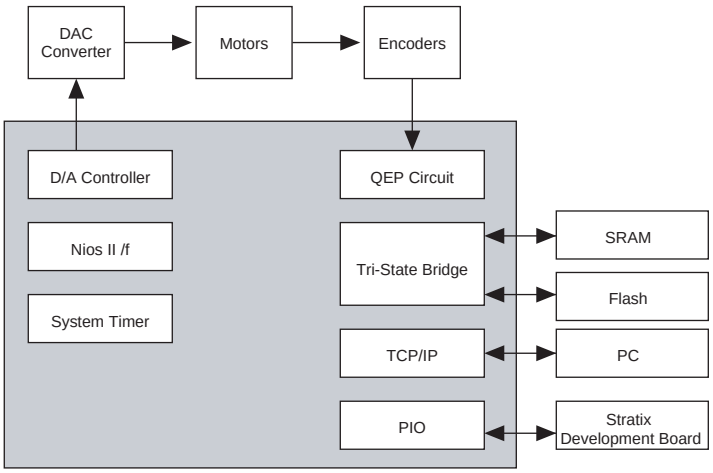
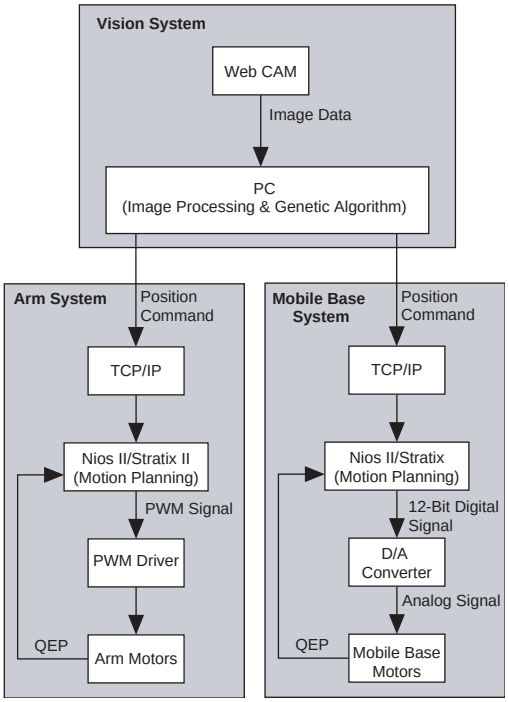


Figure 6 shows the software flow chart, which describes the software and the control signal flow for the visual imaging system, robot arm control system, and platform system.

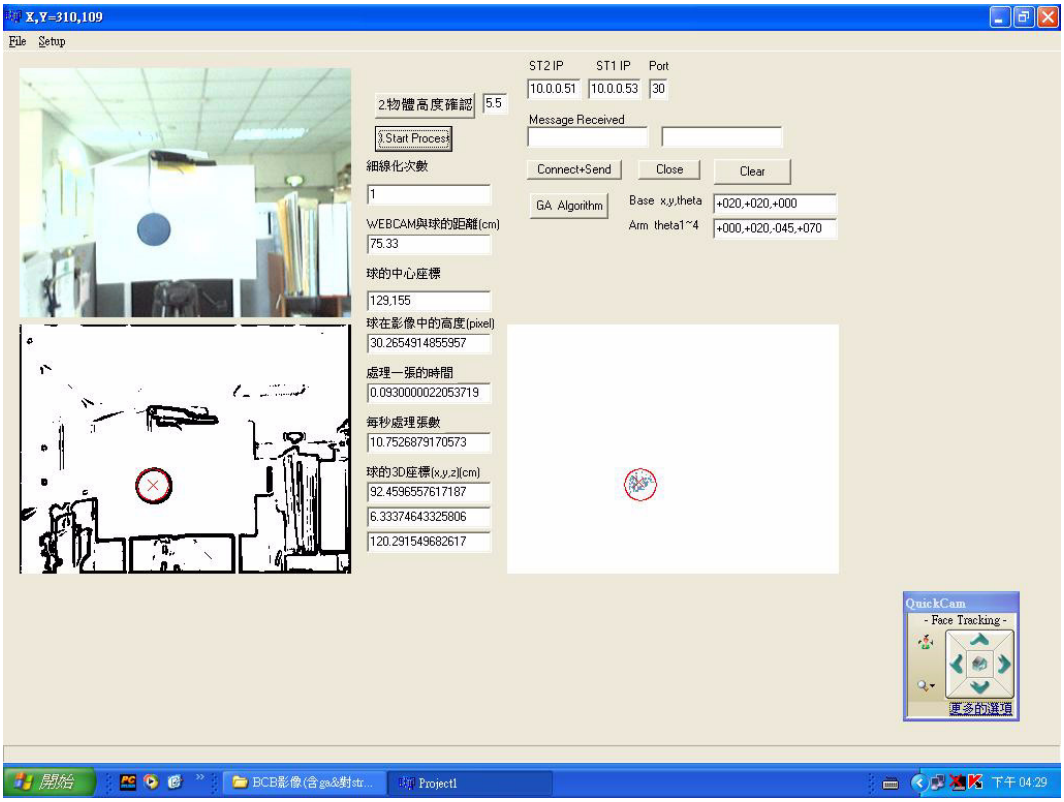
Figure 6. Software Flow Chart



The robot's software has four parts:

- *Image processing*—After selecting the object's image, the smart robot's visual system (webcam) delivers the image to the PC. Then, it uses C++ Builder to analyze and process the data, and read the object's 3-D position. Figure 7 shows the C++ Builder program interface.

Figure 7. Borland C++ Builder Program Interface



- *Robot arm and the platform's genetic algorithm*—The PC uses the C++ Builder program to convert the XYZ coordinates into the rotation angle axes and the platform's rotation coordinates. The PC uses the Winsocket API and TCP communication to deliver the signal to the Nios II processor on the Stratix II development board (for the arm) and the Nios II processor on the Stratix development board (for the platform).

- *Robot arm route planning*—The Stratix II development board receives the rotation angle axes from the PC. We used the Nios II Integrated Development Environment (IDE) to develop the arm's joint motion algorithm. The Nios II processor calculates the point-to-point route planning according to the axis's largest acceleration and speed. Then, it determines the joint's angular speed and acceleration, and outputs the PWM signal to control the arm's rotation. The quadrature encoding pulse (QEP) produced by the motor rotation is sent to the Nios II processor to control the PI servo motor feedback. Thus, the robot arm reaches the specified position and grabs the object.

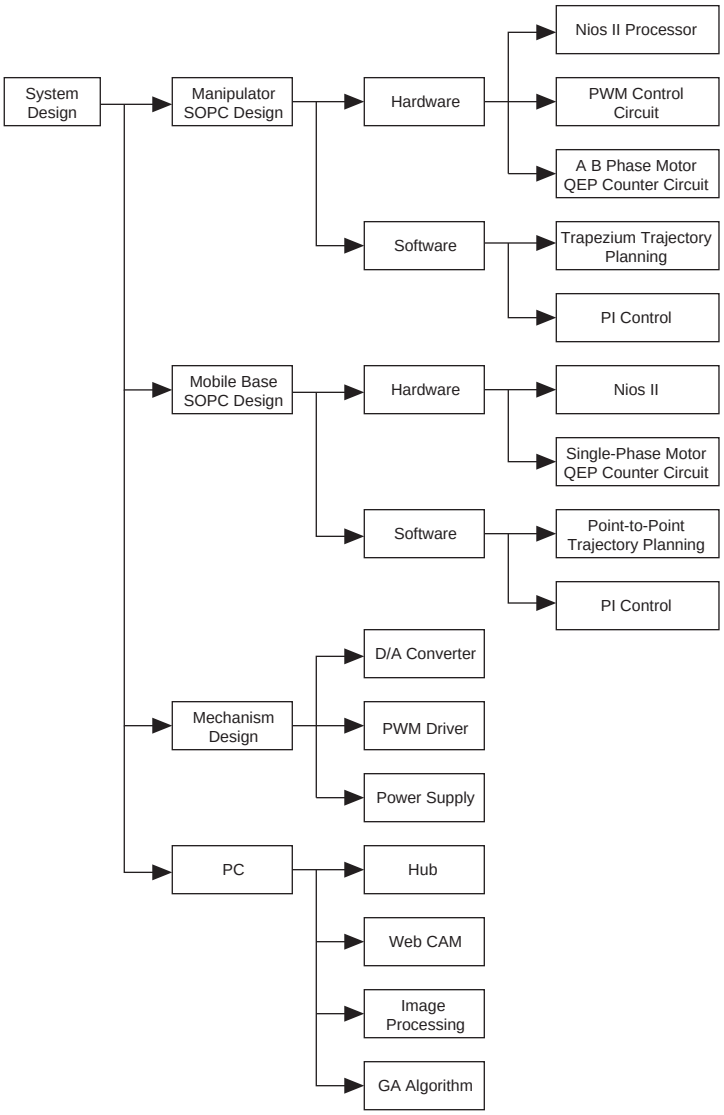
- *Omnidirectional mobile vehicle route planning*—This subsystem controls the omnidirectional mobile vehicle's Stratix development board and receives the object position command from the PC. Then, it determines the kinematic equation according to the physical structure of the vehicle. It uses the Nios II IDE to write the point-to-point route-tracking rule, determine the motor speed, and output the 12-bit digital signal that controls the vehicle motor rotation. The QEP produced by the motor rotation is sent to the Nios II processor to control the PI servo motor feedback.

After simulating the controls on a computer, we implemented the design and confirmed its accuracy using the Stratix II and Stratix development boards.

Design Methodology

Figure 8 shows the system design methodology flow chart. Our methodology has four parts, as described in the following sections.

Figure 8. System Design Methodology Flow Chart



Manipulator Design

The robot arm controller is the Stratix II EP2S60 development board and the Nios II processor is the microcontroller. We implemented the arm's five-axis motor PWM circuit (see Figure 9) and the A, B phase QEP counting circuit (see Figure 10) using hardware circuits. The circuit design lets us implement the arm's servo control system on an FPGA; therefore, it has low cost and low power, and is digital and small.

Figure 9. PWM Circuit Diagram

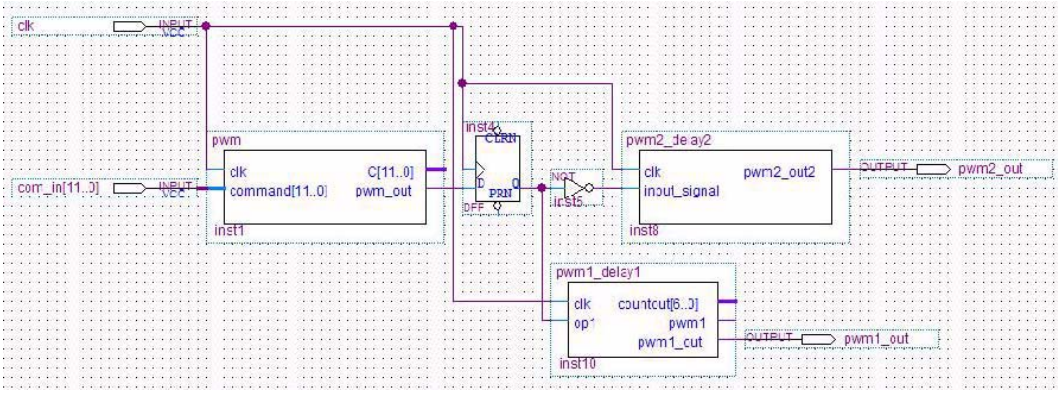
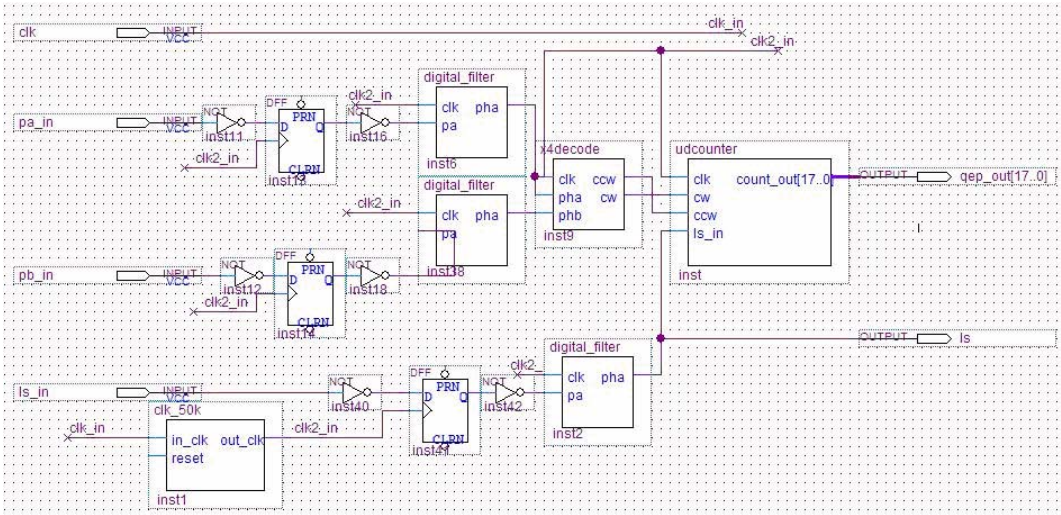
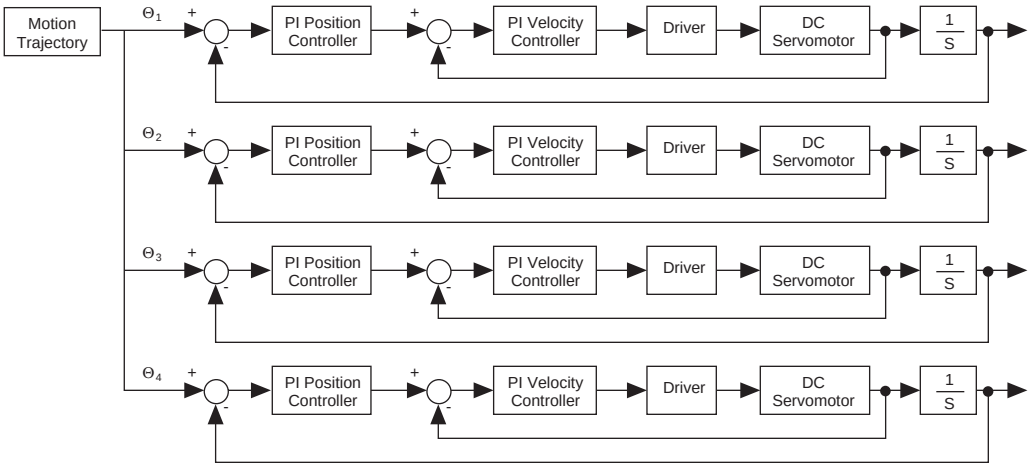


Figure 10. A, B Phase QEP Counting Circuit Diagram



The control rules are written into the Nios II IDE and are implemented in software, including the PI servo motor feedback control (Figure 11) and the trapezoidal route planning. We used software because it facilitated adjusting the K_p and K_i control rule parameters, and we could add program interrupts, which helped save debugging time.

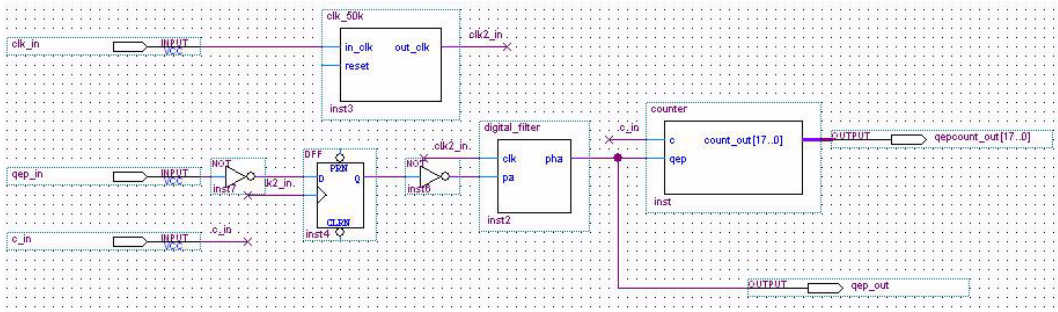
Figure 11. Robot Arm PI Control Diagram



Mobile Base Design

The platform uses the Stratix EP1S10 development board as the controller and the Nios II processor as the microcontroller. We implemented the platform's three-axis motor QEP counting circuit (see Figure 12) using a VHDL hardware circuit. The control rules are written into the Nios II IDE and implemented in software, including the platform's point-to-point route planning and PI servo motor feedback control. We used software because it facilitated adjusting the K_p and K_i control rule parameters as well as the platform's rotation speed.

Figure 12. Single-Phase QEP Counting Circuit Diagram



Mechanism Design

The part of the design concerns the analog-to-digital converter (ADC) used to drive the platform's motor, the PWM driver for the arm's motor, and the robot's power supply.

PC

The PC retrieves image information via the webcam. It determines the object's 3-D coordinates compared to the robot, the robot arm axis's rotation angles, and the platform's moving distance. It sends the information to the Stratix II and Stratix development boards through the hub using TCP/IP.

Design Features

We integrated many functions on the Altera development boards, which greatly reduced hardware requirements and system development time. The two development boards and the PC performed image

processing. The Stratix II development board controls the robot arm, including the control rule calculations and command output. The other Stratix development board calculates the omnidirectional mobile vehicle's control rules, route planning, and control command output. We implemented the control rule calculations in the FPGAs using VHDL and software.

Traditionally, designers use the RS-232 port to communicate between a PC and the Nios II processor. To increase transmission speed and accuracy, we used TCP to transmit the information. With the Altera development boards, we did not need to add additional hardware, but could directly embed the TCP interface into the FPGA. Because we used TCP, the communication between the PC and development board was secure.

The Nios II processor performed well in the system and was outstanding at performing the control calculations and generating the control command output. Because the system is complex, we needed to use two development boards. If we had used a DSP or some other chip instead of the Nios II processor, we would have spent much more time developing the system hardware.

During system development, debugging is necessary. Altera provides good HDL simulation tools to help with this task. Additionally, the LED and seven-segment display on the development boards were very helpful for debugging. With the Nios II IDE hardware debugging interface, we could simply use the Quartus II software and Nios II IDE to develop the arm and platform. These tools are very easy for beginners to learn. For image processing, we used Borland C++ Builder.

Conclusion

Our R&D process shows that Altera's Nios II processor is a very useful soft-core embedded processor system. If we had not used the Nios II processor, we would have spent more time on R&D and would not have achieved such excellent performance. With the Nios II processor as the system core, users can use the Quartus II software and SOPC Builder to create custom Nios II processors and peripherals. The resulting embedded system satisfies the user's requirements for hardware structure, functional features, and resource consumption. For the debugging users encounter in R&D, both the Quartus II software and Nios II IDE provide convenient tools, enabling the user to find problems quickly.

Our design uses the Quartus II software to plan and design the system. The Quartus II software's compete multi-platform design environment satisfies designer's demands and is an all-around tool for SOPC design. Its diversified functions fully support the VHDL and Verilog HDL design process, which eases hardware circuit design. Additionally, the user can simulate and test the HDL designs using the same tool. The Quartus II software provides a simulator and also supports third-party simulation tools such as the ModelSim® software. Simulation checks whether the design circuit is correct before the design is impemented in hardware, shortening debugging time. In our project, we integrated both the software and hardware design into the FPGA, which would have been impossible in other development systems. Using Altera tools and FPGAs greatly reduced the number of wires and electronic components our design required.

The Quartus II software also includes the library of parameterized modules (LPM), which helps the user create complex, advanced systems. The LPM can be used widely in SOPC designs, or used together with ordinary Quartus II designs. The Quartus II software's MegaWizard® Plug-In Manager helps the user create function modules, LPM functions, and intellectual property (IP) functions.

For the three parts of our design, only the image processing part does not use the Nios II CPU. We plan to work on this aspect of the design and implement the image processing in the FPGA as well. Working on this project gave us a deeper understanding of the Nios II processor, and we plan to use it to develop new designs in the future.