Second Prize

Aerial Photographic System Using an Unmanned Aerial Vehicle

Institution: Chungbuk National University

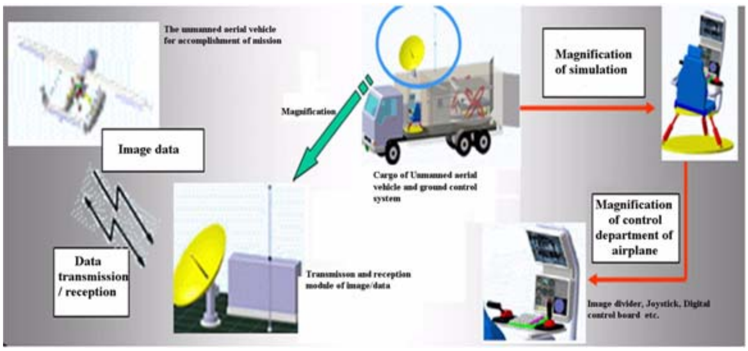
Participants: Hyuk Joong Kwon, Woo Joong Kim, Jang Geun Kim, Sang Bae Park

Instructor: Professor Jung-Kwan Seo

Design Introduction

While many software applications support JPEG image compression, few FPGA applications support it, and those that do are very expensive. Currently, software implementations of JPEG compression use many operations, and adding an algorithm slows performance. Our system implements JPEG compression using an FPGA, which improves the compression rate and operation speed. We implement JPEG compression using the Nios® II soft-core processor, which also performs aerial photography on an unmanned aerial vehicle (UAV). Figure 1 shows the overview of the project.

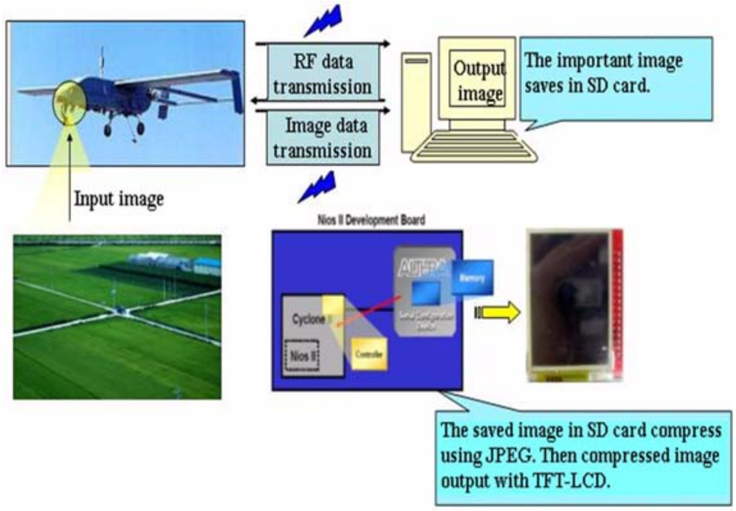
Figure 1. UAV Project Overview



Function Description

The aerial photography is saved onto a secure digital (SD) card using a personal computer. The Nios II processor running on the development board compresses the saved image. Then, the compressed image is output onto a thin film transistor liquid crystal display (TFT-LCD). See Figure 2.

Figure 2. Design Detail



Performance Parameters

The most important performance aspect of the system is tracking the images of a specific object saved in the SD card and keeping the image data. The saved image in the SD card is output onto a TFT-LCD via the Nios II development board. See Figures 3 and 4.

Figure 3. Object Tracking

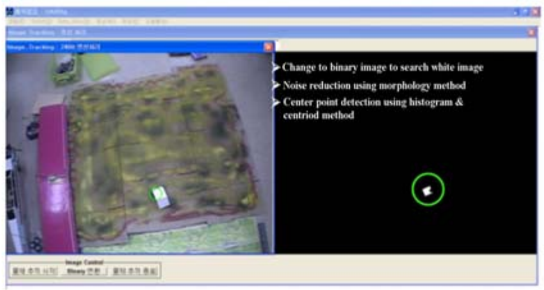
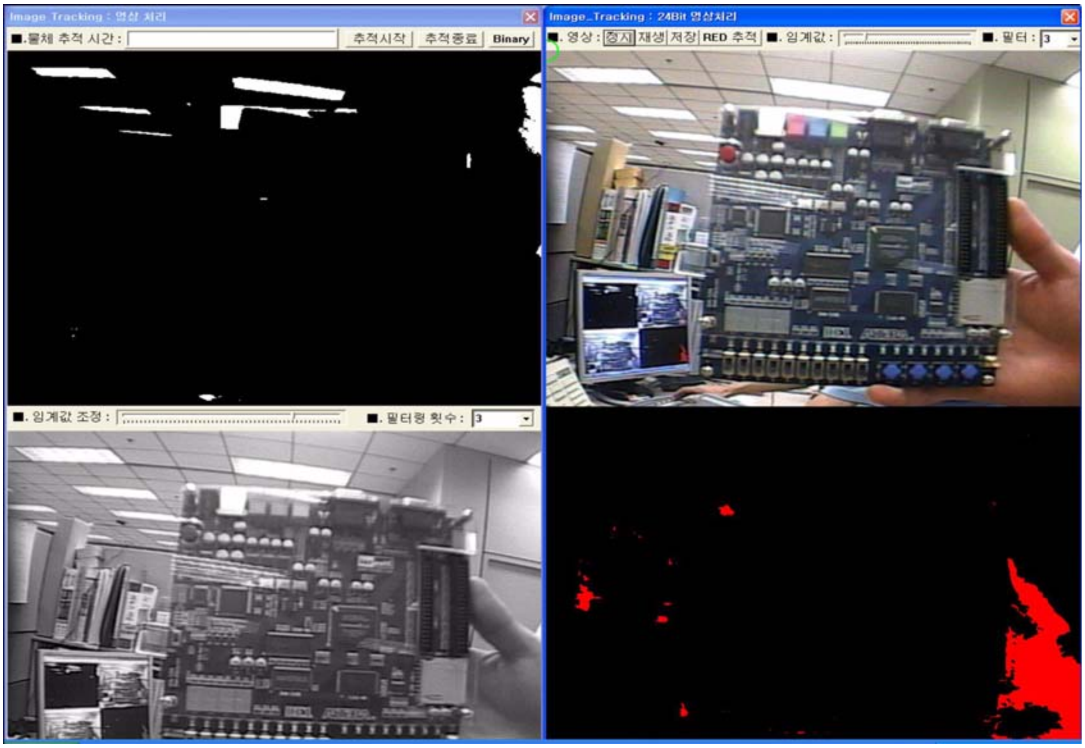


Figure 4. User Interface and External Pilot Image Processing



Design Architecture

This section describes the hardware design block diagram, software flow chart, and JPEG algorithm.

System Structure

Figure 5 shows the aircraft's internal system.

Figure 5. Aircraft Structure

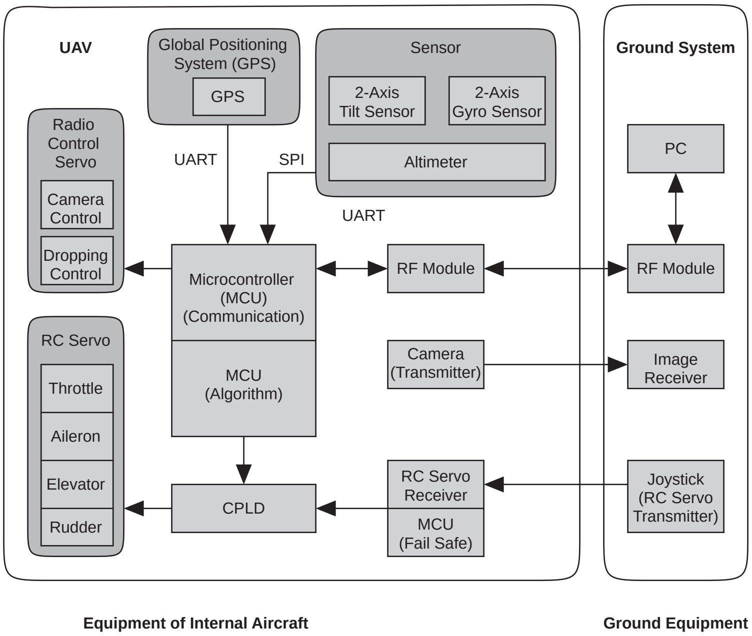
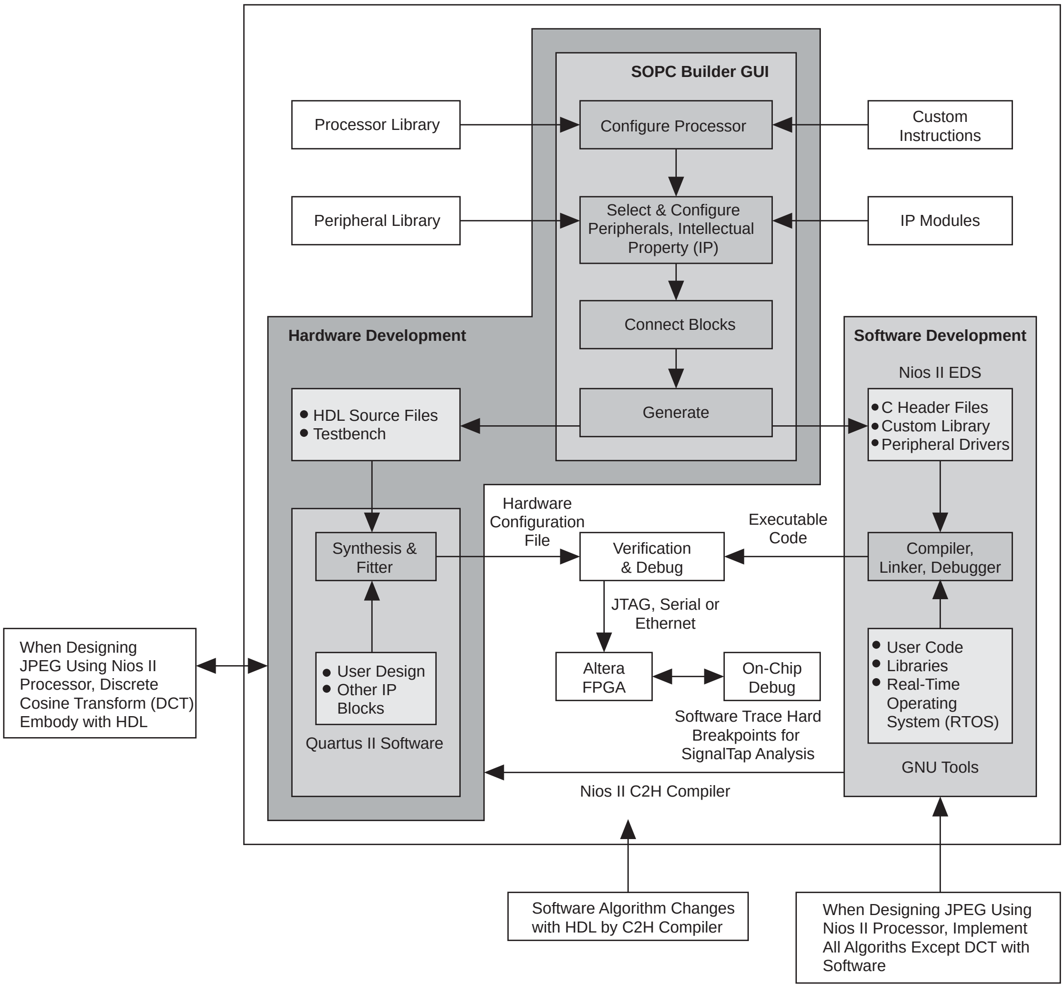


Figure 6 shows the Nios II system.

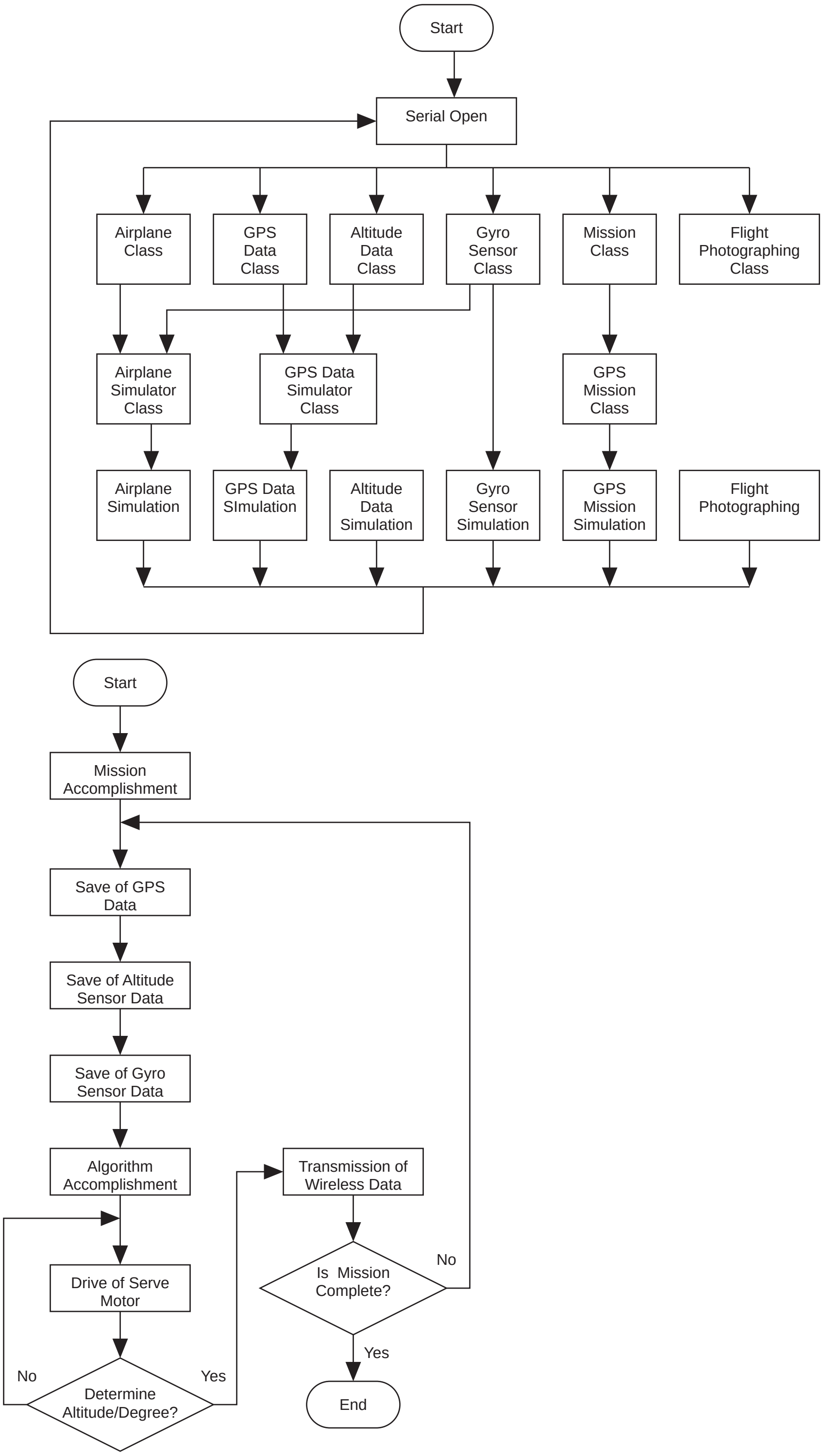
Figure 6. Nios II System Flow Chart



UAV Software Flow Chart

Figure 7 shows the aircraft's internal and external programs.

Figure 7. UAV Software Flow Chart



JPEG Software Flow Chart and Algorithm

JPEG compression has a basic method and an expanded method. The basic method uses 8 bits for each color in a pixel and consists of a sequential mode and Huffman encoding. The expanded method supports larger applications with 8 or 12 bits for each color in a pixel, sequential mode or progressive mode, Huffman encoding, and arithmetic code. The user can select which mode to use according to the application.

Figure 8 shows the JPEG encoding process. First, the converted input image is divided into 8 x 8 pixel blocks. A discrete cosine transform (DCT) operation is executed on each block to obtain the DCT coefficient. The DCT coefficient consists of direct current (DC) and alternating current (AC) components. Each component is independently quantized and the quantization table is individually created. The DC part of the DCT coefficient is encoded by the difference between the DC coefficient of the current block and the DC coefficient of the previous block. The AC components form an array by zig-zag scanning every block, after which the AC component is encoded. Figure 9 shows the DCT algorithm.

Figure 8. JPEG Encoding Flow Chart

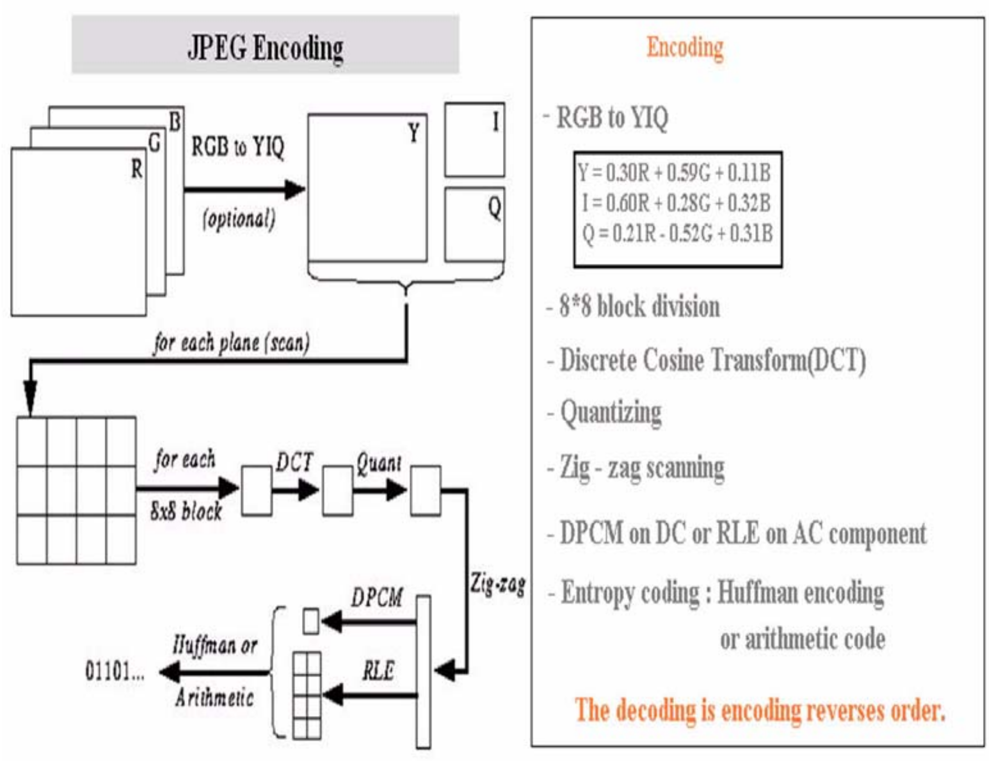
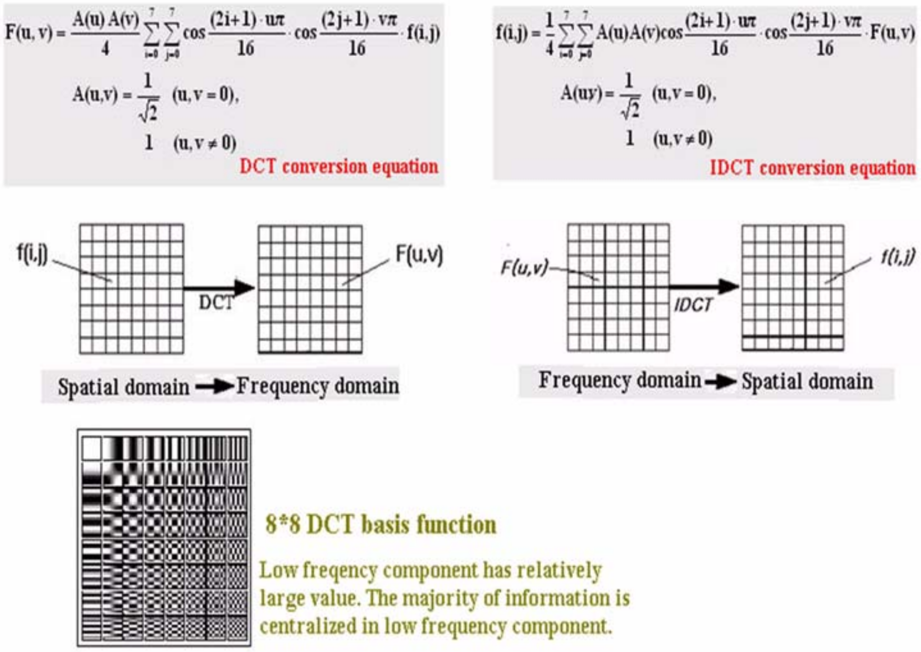


Figure 9. DCT Algorithm



Design Description

This section provides a detailed description of our design.

UAV Hardware Development Environment

Figure 10 shows a photograph of the aircraft and Table 1 shows additional details.

Figure 10. UAV Photograph

Table 1. UAV Details

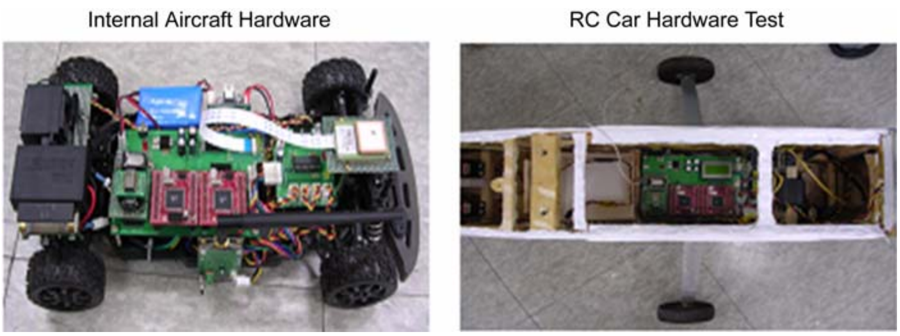
Plane Component	Description
Place Class	High Wing Plane
Body Material	Balsa and Plywood
Wing Span	2,040 mm
Fuselage	1,635 mm

Table 1. UAV Details

Plane Component	Description
Flying Weight	5,500 g

Figure 11 shows the internal hardware of the aircraft. We tested the hardware on a remote control (RC) car before using it on the aircraft.

Figure 11. Aircraft Internal Hardware



Nios II Development Environment

Figure 12 shows the system in SOPC Builder and Figure 13 shows the system generation result. Figure 14 shows the overall system schematic. We used the speed of the Nios II/f processor core to perform the compression quickly. Table 2 compares the features of the Nios II processor variants.

Figure 12. Configuration of SOPC Builder

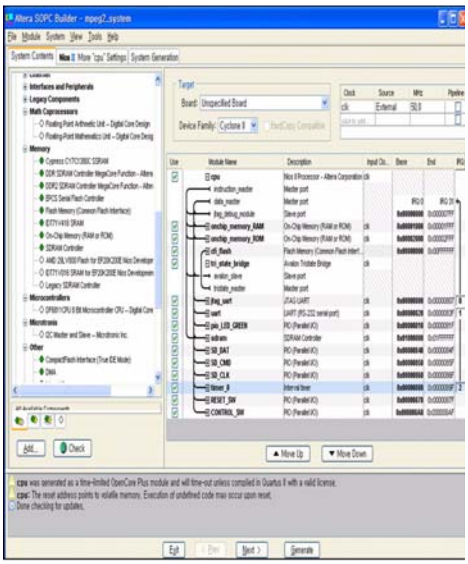


Figure 13. System Generation Result

Figure 14. Overall System Schematic

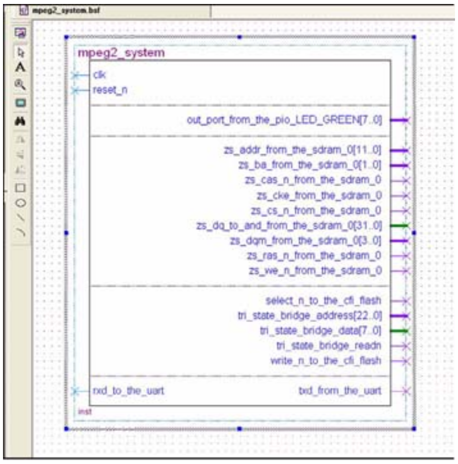


Table 2. Nios II Processor Features

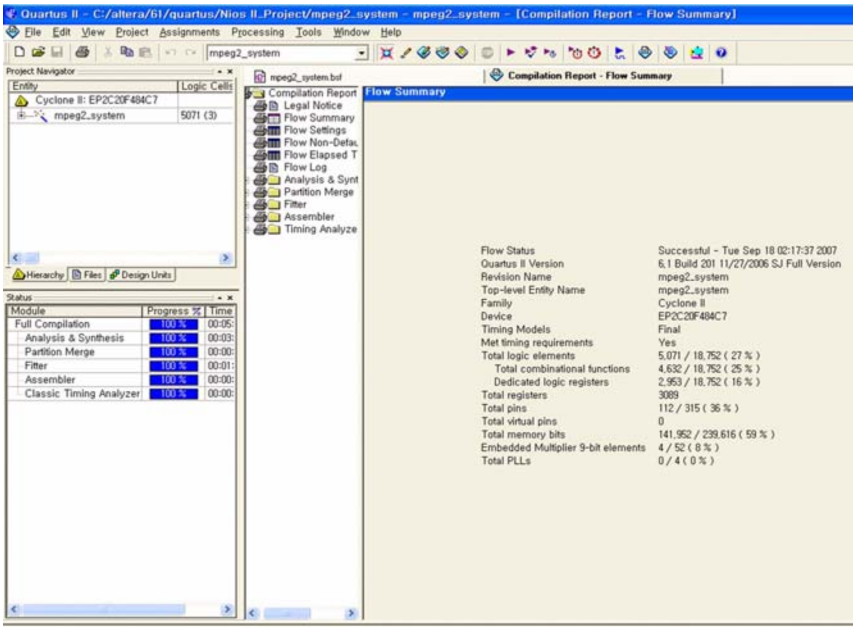
Feature		Core		
		Nios II /e	Nios II/s	Nios II/f
Objective		Minimal core size	Small core size	Fast execution speed
Performance	DMIPS/MHz	0.15	0.74	1.16
	Max. DMIPS	31	127	218
	Max. f_{MAX}	200 MHz	165 MHz	185 MHz
Area		< 700 logic elements (LEs) < 350 adaptive logic modules (ALMs)	< 1,400 LEs < 700 ALMs	< 1,800 LEs < 900 ALMs
Pipeline		1 stage	5 stages	6 stages

Table 2. Nios II Processor Features

Feature	Core		
	Nios II /e	Nios II/s	Nios II/f
External address space	2 Gbytes	2 Gbytes	2 Gbytes

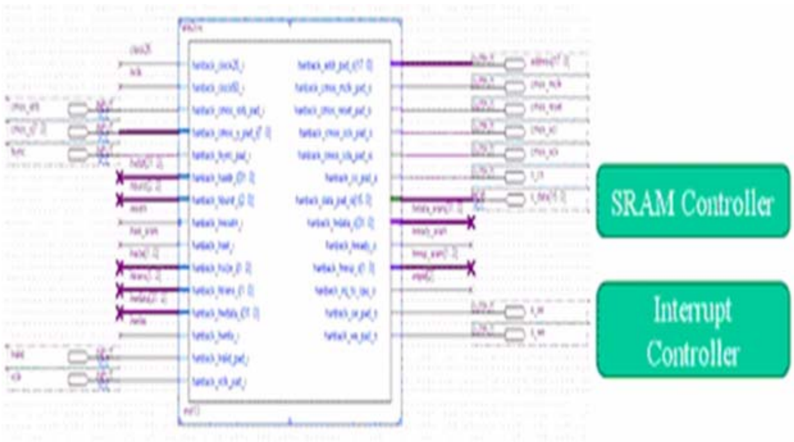
We compiled the generated system in the Quartus® II software. Figure 15 shows the compilation report. The system used 27% of the total available logic, 36% of the available pins, and 59% of the memory bits.

Figure 15. Quartus II Compilation Report



Figures 16 and 17 show the SRAM controller and TFT-LCD controller, respectively.

Figure 16. SRAM Controller



Input Ports	Output Ports
`fsync` = Synchronization of input image `vclk` = Data output clock `lvalid` = Line value	`ntpld[2]` = Interrupt generation `address[17..0]` = SRAM save address `S_data[15..0]` = SRAM save data `S_cs` = SRAM chip select `S_oe` = SRAM output enable `S_we` = SRAM write enable

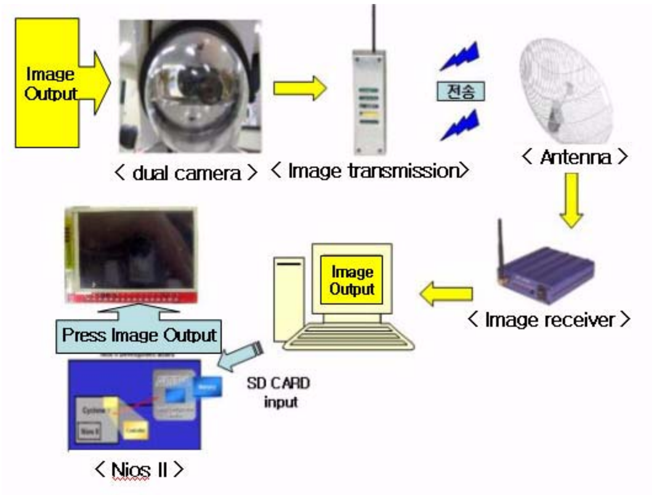
Figure 17. TFT-LCD Controller

Output Ports
`bled_o` = Back light on/off `lcd_de` = LCD data enable `lcd_data[15..0]` = RGB[5:6:5] `lcd_mclk` = LCD clock

Image Processing Development Environment

Zoom cameras have many constraints when used in aircraft because of their size and weight. Using two camera lenses, we reduced this problem. Figure 18 shows the image processing using the dual camera lenses.

Figure 18. Image Display Processing



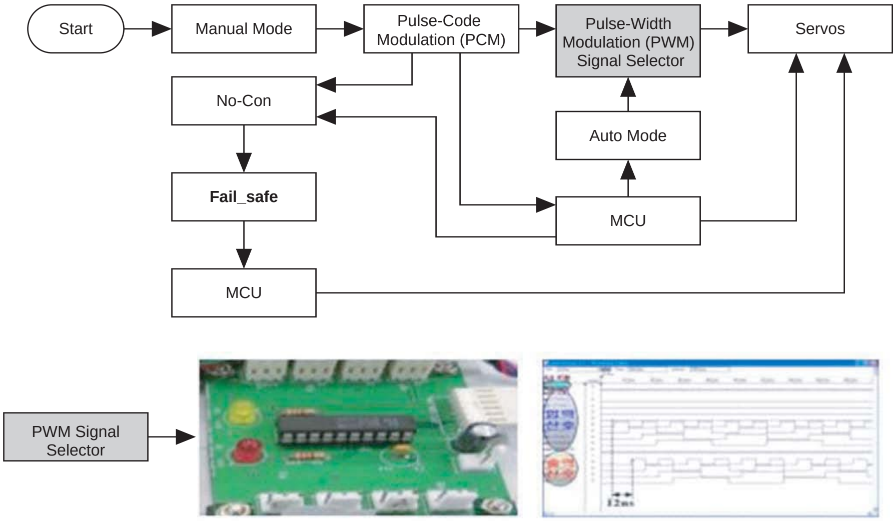
Design Features

This section describes the features of our design.

Flight Safety Test

To convert between automatic and manual piloting safely and rapidly, we built a switch using the GAL16V8 logic device. The device is simple to use. We implemented the conversion between automatic and manual piloting using channel five in the ratio controller. Two LEDs display whether the aircraft is on manual or automatic pilot, and can be seen easily. Figure 19 shows the setup of this switch using the logic device.

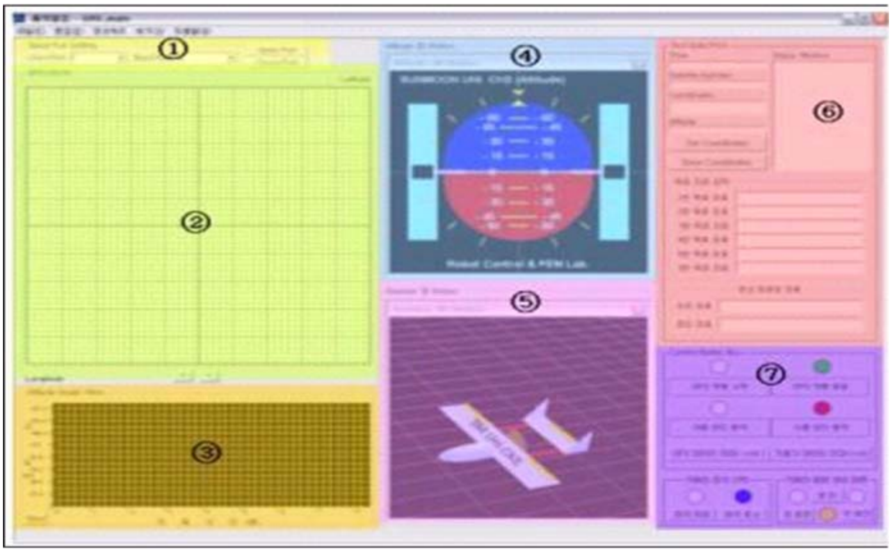
Figure 19. Automatic and Manual Pilot Switch Hardware and Flow



User Interface

A user interface provides communication with the aircraft (see Figure 20). See the following discussion for more information on the numbered areas in Figure 20.

Figure 20. Aircraft Communication User Interface



1. This part of the interface displays data from the hardware system. The application uses serial communication to connect to the hardware system and data is sent asynchronously.

2. This area represents the aircraft movement with a line. The user can judge any objective errors by enlarging or reducing the scale. Additionally, the user can determine whether the mission was executed properly by marking the position of the objective location.

3. This area displays the altitude using a line. The aircraft's altitude is shown in real time.

4. The 3-dimensional (3-D) motion area determines the aircraft's altitude and inclination using a 3-D program.

5. An OpenGL program shows the aircraft's inclination and direction.

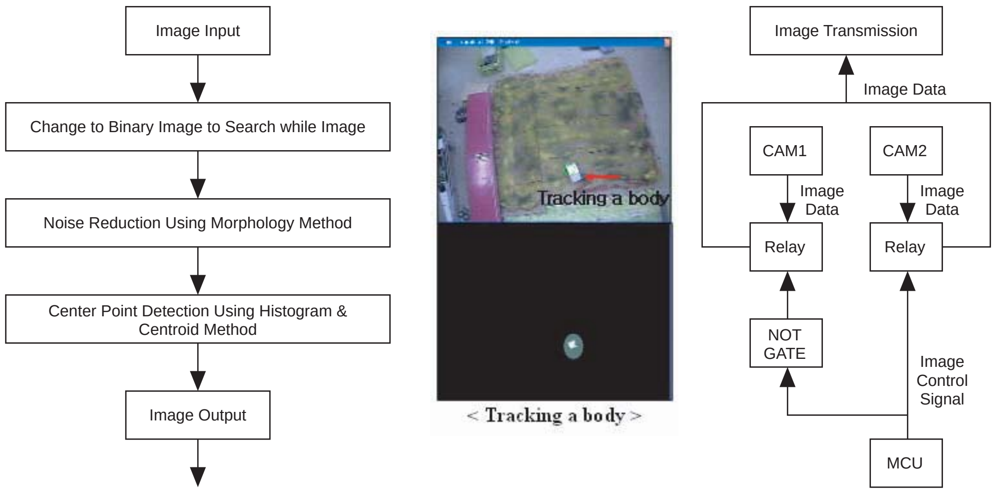
6. This area shows the GPS data received from the aircraft. The application also transmits the objective location to the aircraft. Additionally, it shows the tracking information obtained by the aircraft.

7. This area contains the aircraft control buttons for functions such as turning the GPS module on and off, the camera's position control, the field of view, etc.

Image Processing Algorithm and Camera Interface Hardware

Figure 21 shows the image processing flow as well as the camera's hardware interface.

Figure 21. Image Processing Algorithm and Camera Hardware Interface

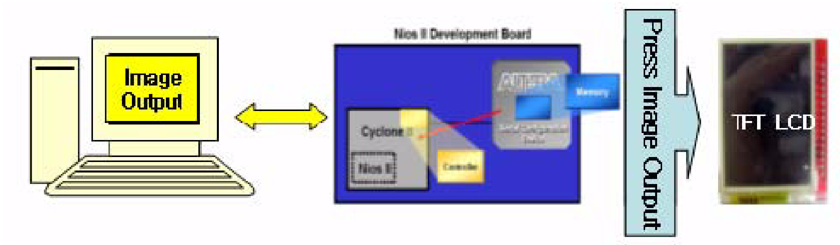


Nios II Processor Role

The Nios II processor compresses the aerial photography using JPEG compression. The camera collects a lot of image information to ensure that the mission has executed correctly. The image compression system codes the aerial photography and then decodes the collected images. Figure 22 shows how the Nios II processor fits into the system.

The system collects real-time image data and saves it to an SD card. The saved images are JTAG compressed by the Nios II processor running on the development board. Finally, the compressed image is output onto a TFT-LCD monitor.

Figure 22. Nios II Processor Role



Conclusion

We learned a variety of things while working on this project, such as:

- We used the JTAG module to perform many experiments with the Nios II processor and the development board.
- We could download the program to memory, and easily start and stop the program execution. The breakpoint and watchpoint features made it easy to debug any problems. It was useful to reference data by analyzing registers and memory.
- The Nios II Integrated Development Environment (IDE) example code and project templates made it easier for us to build the system.
- We used the µC/OS-II (real-time kernel) that was provided with the development kit.
- When we started the project, the on-line demonstrations, such as "Creating a Nios II System" on the Altera web site, were helpful.

The development board has built-in flash memory, but it is very small. We needed to change the scope of our project from using MPEG compression to using JPEG compression, which is relatively small in size. In the future, we will plan to use external memory, which will allow us to perform aerial photography using video compression.