First Prize

Passive Digital Camera

Institution: **Hanyang & Yonsei University**

Participants: **Ji Won Kim, Doe-Hoon Kim, and Shin Seung-Chul**

Project Leader: **Min-Chul Kwon**

Design Introduction

Today we live in a society where digital devices and related software tools enable us to capture videos. Thanks to advances in semiconductors, it has become affordable to own personal video equipment such as video and digital cameras. People use video cameras to store and share their experiences, and blogs on the Internet to share their thoughts.

Because of popular interest in image acquisition and its storage, a unique set of problems in managing these images has been created. Users often need to selectively access images; therefore, they need tools that can consistently enable them to select, sort, and manage the images. In the past, image collections were small, and therefore easy to manage, but with the ever-growing library of images, it is becoming increasingly difficult to classify and locate these images. To solve this problem, you need a tool that can classify and search images automatically. The image compression standard, MPEG-7, was developed based on this need. However, it is not useful for personal video users because digital camera manufacturers supply software that provides only the acquisition date and time stamp, and classifies images based on these parameters. Unfortunately, searching for an existing image without the time stamp is impossible based on this technique.

To solve this problem, we would like to propose a device called the passive digital camera featuring image-classification software. We call this device a passive digital camera because the digital camera, not the user, decides the shot time. The passive digital camera determines the acquisition time of the image by using various sensors. The camera recognizes the local conditions such as temperature, velocity, and tilting angle.

We have developed a software module for image analysis based on the image classification descriptor found in the MPEG-7 standard. The HW-IP module, which analyzes images, does so by considering the ability of the device and transferring it to a PC. The module also makes software control easy by loading the uCLinux real-time operating system (RTOS). Our digital camera is compact and lightweight. Being a portable device, it is best if it can be implemented as a one-chip solution. Altera's FPGA and Nios® II embedded processor provide a huge advantage in this type of design.

Function Description

Our Personal Black Box (PB$^{2)}$ design collects image data that is beyond the user's cognition capability. We tried to use a grouping algorithm, which sorts collected images automatically, but we had to discard this approach because it needed a more powerful computing device. Instead, we tried to perform image classification in hardware. To enable this methodology, PB^2 would include a global positioning system (GPS) module in which it stores location information for individual images. We had to discard this approach also, and try a different method.

Using an acceleration sensor and an infrared ray sensor, we thought that we could fix the shaking noise in the image acquisition. This method worked partially because it enabled us to determine how long it took to acquire images. The image module in our design features more than a 300-Kbyte pixel capability and can control 640 x 480-size 24-bit color images. For this project, however, we used 320 x 240 image sizes.

Our PB^2 supports 1 Gbyte or more of flash memory and an SD card interface, using serial peripheral interface (SPI). We have also implemented the FAT32 file system. Because the PB^2 operates in real-time, it offers enhanced reliability in image acquisition timing. To achieve this reliability, we developed two software modules: the first module features single-threaded firmware, and the second features a multi-threaded program on µC/OS-II. These modules work identically.

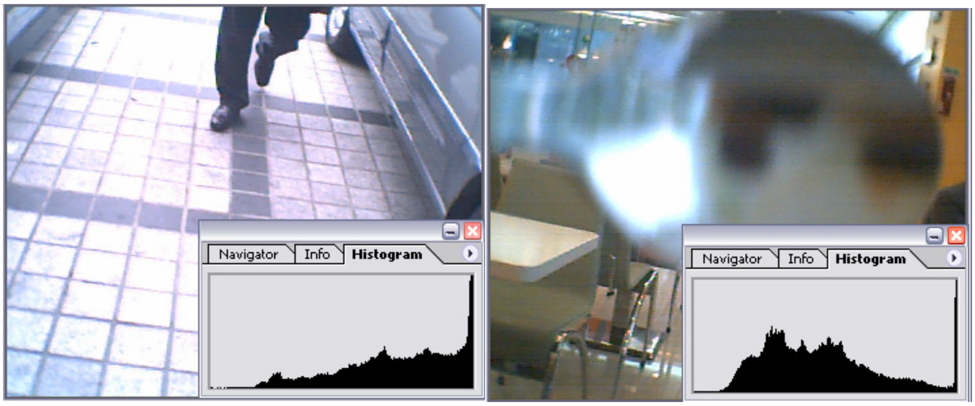
With our power management circuitry we thought we could guarantee up to 16 hours of operation. (however, we have not been able to meet this expectation). Our product was implemented on the UP3 development board, but power management needs its own full custom design. On the UP3 development board the 1.2-V 900-mAh Ni-MH x 5 starts at 6.6 V, and an hour later it falls to 5.4 V and disables the UP3 board. We suspect this may be due to the power consumption of the LCD backlight.

Performance Parameters

The image control module uses a probability algorithm, the analysis of which leads to the dispersion comparison of images.

To refrain from acquiring unnecessary images, the module compares the head of the most recent two images. It then extracts the sample from the center of the image and compares it with the other image. To make the operation more efficient, it extracts the minimum sample for a comparison. See Figure 1.

Figure 1. Distribution of Each Image



The statistical tests were carried out on the PC as shown in Figure 2. We then used the Nios II processor to perform the tests and found that our operation efficiency was equal to the PC environment by a significant level (0.10).

Figure 2. Testing of Statistical Hypothesis

$$F = \frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} = \frac{\frac{1}{n_1-1}\sum_{i=1}^{n_1}(X_i-\overline{X})^2/\sigma_1^2}{\frac{1}{n_2-1}\sum_{i=1}^{n_2}(Y_i-\overline{Y})^2/\sigma_2^2} \sim F(n_1-1, n_2-1)$$

Testing of statistical hypothesis :

Null hypothesis $H_0 : \frac{\sigma_1^2}{\sigma_2^2} = 1$ and

Testing statistics : $F = \frac{S_1^2}{S_2^2}$

Alternative hypothesis $H_1 = \sigma_1^2/\sigma_2^2 > 1$

Significant level : 0.10

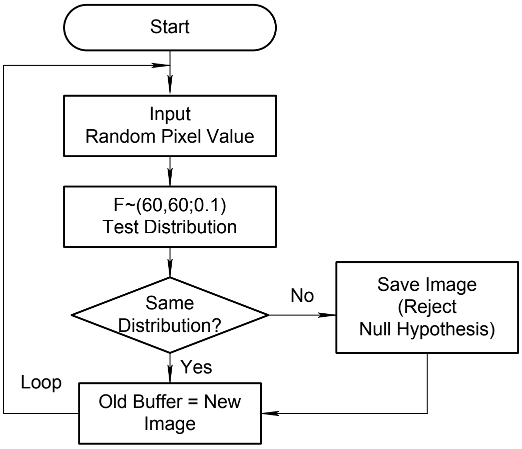
Critical region : $F \geq F(n_1-1, n_2-1; \alpha)$

Design Architecture

While developing our design, we found that the MPEG-7 standard implementation posed many constraints that were too big to handle. We had a very small-capacity FPGA device, and we needed to integrate several peripherals. Hardware acceleration was almost impossible to achieve under these conditions. Also, we needed to develop many device control software routines, which required a simpler algorithm. When we have a more powerful FPGA available to us, we can reconsider hardware acceleration.

The image matching algorithm flowchart is shown below in Figure 3. We contrasted and compared two images based on a randomly chosen pixel's distribution ratio. If the ratio is over a specific limit, we deem the two images to be different, and then store the new image.

Figure 3. Testing Algorithm Flow Chart

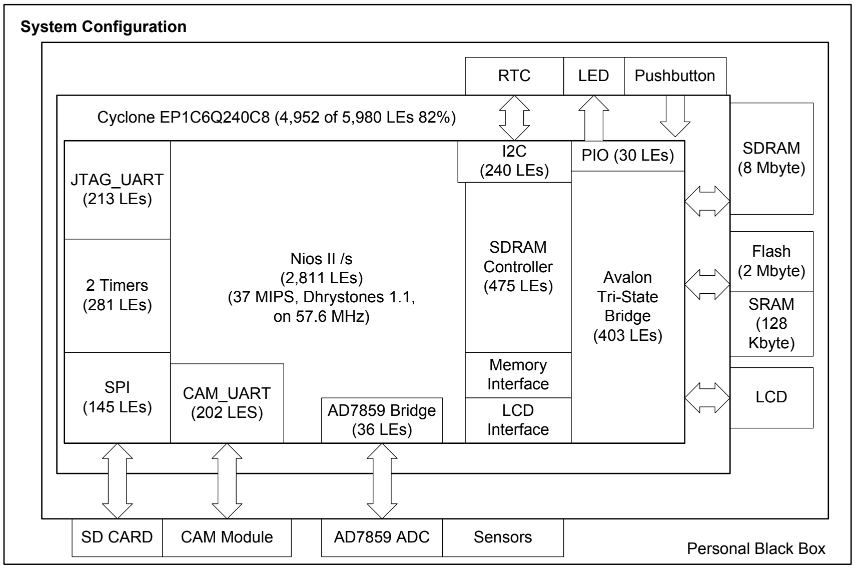


Hardware Design

We used an Altera® FPGA in our system design, where it functions as the main controller, handles all sensor functions, and manages all peripherals. The sensor module determines the image acquisition time. The peripherals in our design consist of a mass storage device and a battery module. Figure 4 shows the PB^2 hardware block diagram, including logic element (LE) usage. Later, we plan to add a battery module to the design.

We designed the image analysis function using the Nios II processor. The image analysis function's performance can be enhanced with statistical data analysis software such as the MATLAB software; or you can use a generic C program module.

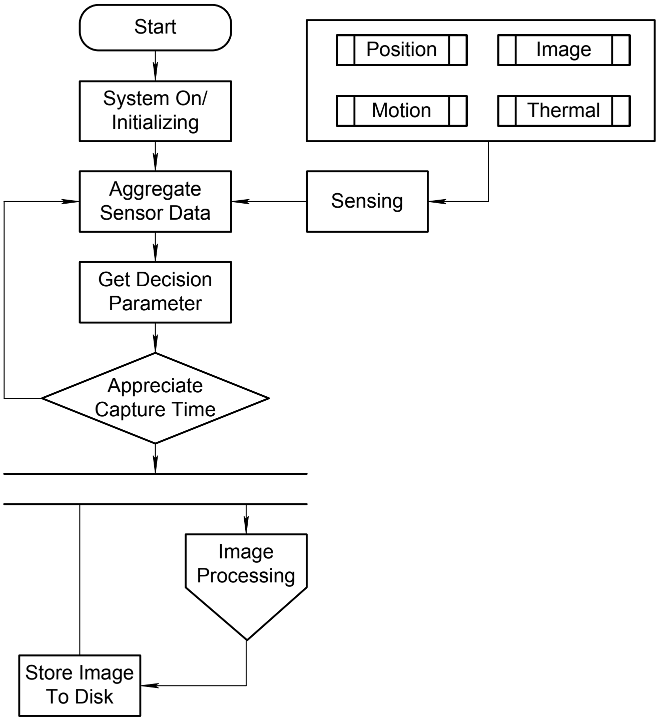
Figure 4. Hardware Block Diagram



Software Design

Figure 5 shows the software flow chart for image acquisition, excluding the image processing. Each sensor helps determine the image acquisition time. The system uses a heat sensor, speed sensor, and tilt sensor. The image processing function starts before image storage and is determined by the available power and storage parameters. The image processing module comprises two parts: an image description extraction and classification of captured images. We use a PC to perform image classification. Therefore, each processing module task can be performed independently of the other.

Figure 5. Software Process Flowchart



Figures 6 and 7 show the single-threaded and multi-threaded state diagrams, respectively.

Figure 6. Single-Threaded Software State Diagram

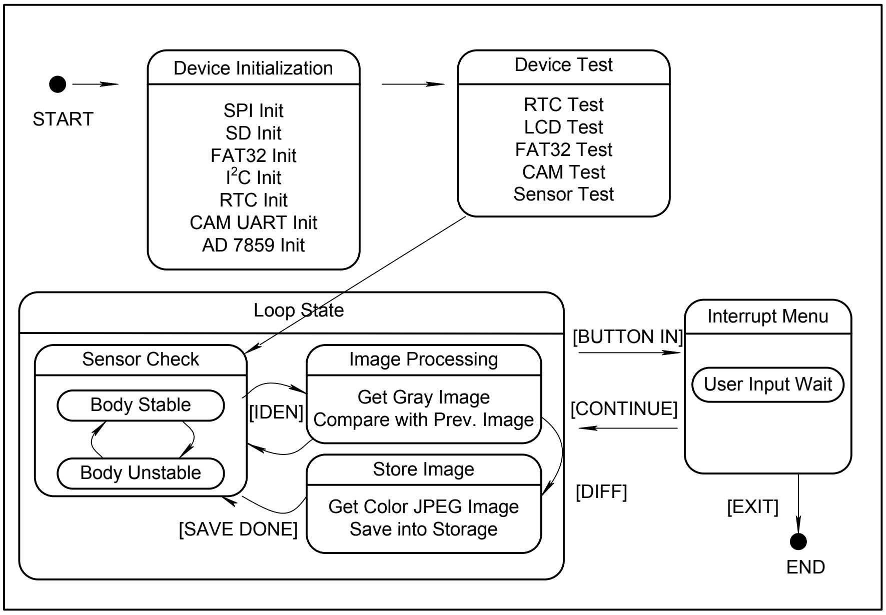
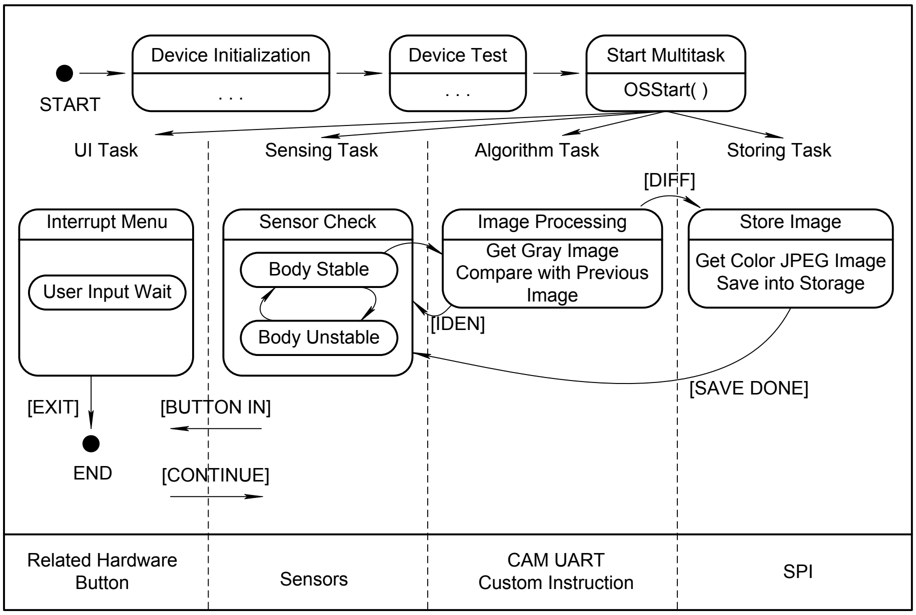


Figure 7. Multi-Threaded Software State Diagram



Although the multi-threaded software module looks good, it was not much help in our design application. A slow image sensor and a fast processor for the image algorithm made it difficult to partition tasks. We feel, however, that the multi-threaded software module works well when all of the peripherals are reliable and the processing speed is uniform.

Design Methodology

Because our system clock was set at 57.6 MHz, it caused peripheral communication errors, especially with the UART baud rate. By keeping the board clock at 48 MHz, we managed to get 57.6 MHz using a phase-locked loop (PLL) adjustment. The main modules in our system were:

- 57.6 MHz Nios II/s processor with JTAG level 1 and a 14.4-MHz SPI bus for SD card control (with wide clock range support). We used SOPC Builder to calculate the clock latency, which was very helpful.

- I^{2}C Core (from opencores.org) for real-time clock (RTC) implementation. We found that Altera's OpenCore® Plus I^{2}C core required too many constraints for our system.

- 115.2-kbps UART for CAM module communication. Because this was a critical module, we changed and tuned the system clock. The CAM integrated circuit (IC) supports SPI mode for up to 3 MHz, but had some restrictions when used as a UART-type module. We could have selected the raw CAM module without using the UART interface, but at 115.2 kbps it was too slow to obtain the image.

- SDRAM memory controller. This module used a 57.6-MHz clock, but needed some phase-shift adjustments. We set this shift to -82.0 degrees, which was determined by board-specific characteristics.

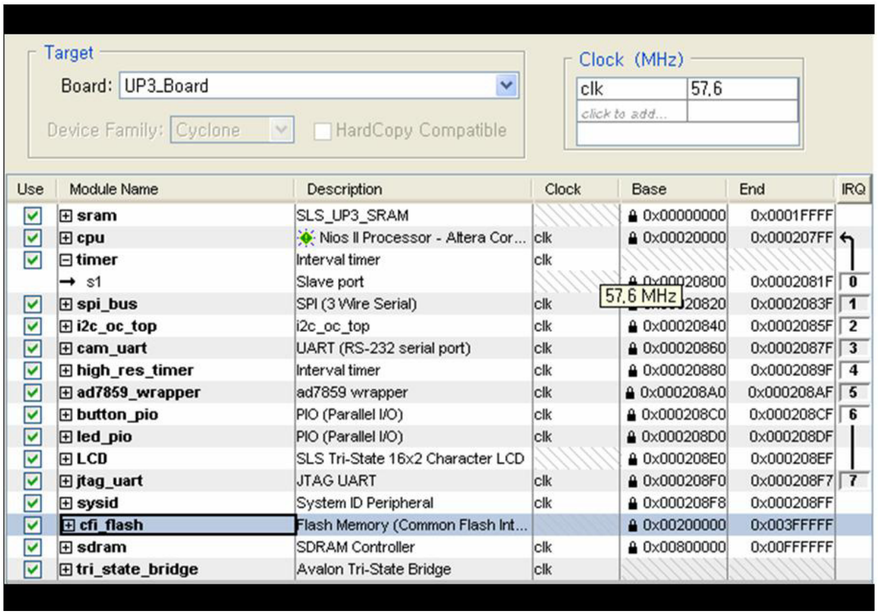
- SRAM and CFI. This module is a tri-state bridge between memories.

- AD7859 user logic bridge. The AD7859 has control and data pins. We made AD7859 a bridge component using the software's new component feature. Because we had good knowledge of the Avalon bus interface, connecting peripherals was easy.

- Keypad and LED parallel (I/O) (PIO). This module comprised a 16 x 2-character LCD interface, buttons, and LEDs.

The SOPC Builder settings are shown in Figure 8.

Figure 8. SOPC Builder Settings



Target

Board: UP3_Board

Device Family: Cyclone

☐ HardCopy Compatible

Clock (MHz)

clk	57.6
click to add...	

Use	Module Name	Description	Clock	Base	End	IRQ
☑	⊞ sram	SLS_UP3_SRAM		0x00000000	0x0001FFFF	
☑	⊞ cpu	Nios II Processor - Altera Cor...	clk	0x00020000	0x000207FF	
☑	⊟ timer	Interval timer	clk			
	→ s1	Slave port		0x00020800	0x0002081F	0
☑	⊞ spi_bus	SPI (3 Wire Serial)	clk	0x00020820	0x0002083F	1
☑	⊞ i2c_oc_top	i2c_oc_top	clk	0x00020840	0x0002085F	2
☑	⊞ cam_uart	UART (RS-232 serial port)	clk	0x00020860	0x0002087F	3
☑	⊞ high_res_timer	Interval timer	clk	0x00020880	0x0002089F	4
☑	⊞ ad7859_wrapper	ad7859 wrapper	clk	0x000208A0	0x000208AF	5
☑	⊞ button_pio	PIO (Parallel I/O)	clk	0x000208C0	0x000208CF	6
☑	⊞ led_pio	PIO (Parallel I/O)	clk	0x000208D0	0x000208DF	
☑	⊞ LCD	SLS Tri-State 16x2 Character LCD		0x000208E0	0x000208EF	
☑	⊞ jtag_uart	JTAG UART	clk	0x000208F0	0x000208F7	7
☑	⊞ sysid	System ID Peripheral	clk	0x000208F8	0x000208FF	
☑	⊞ cfi_flash	Flash Memory (Common Flash Int...		0x00200000	0x003FFFFF	
☑	⊞ sdram	SDRAM Controller	clk	0x00800000	0x00FFFFFF	
☑	⊞ tri_state_bridge	Avalon Tri-State Bridge	clk			

Using SOPC Builder, we designed a general microcontroller like AVR or ARM. This approach saved us additional hardware design work. Then we developed device drivers. We found that the Nios II Hardware Abstraction Layer (HAL) is good enough for normal operation, but we needed to customize it for better performance. We found some compilation problems as well. The memory alignment for structures was not documented.

Design Features

Our design had the following features.

Performance

Image processing requires many functions, and image processing resources are limited in mobile devices. Consequently, hardware/intellectual property (IP) implementation can help improve image processing efficiency.

Portability

The design must be flexible. Because the whole system is complex, you can get around this requirement by using a PC-based camera system to perform software conversion easily. We used the Nios II embedded processor for a hardware implementation of software functions, which is more convenient.

Low Power Consumption (Abandoned)

Our original design had low power consumption as one of the performance parameters, which we could not accomplish. Power consumption is determined by the operation time of the hardware module. Also, mobility considerations and related sensor operations have to be taken into account to create a low-power design.

Integration

The image processing unit put a strain on integration. From image acquisition to sorting to storage, we needed to have the hardware and software modules working together in the design. Thanks to the Nios II processor, we managed to accomplish that in our design.

Conclusion

Using Altera's system-on-a-programmable-chip (SOPC) solution, we learned a new way to solve system design problems. By employing an SOPC design in our system, we learned its advantages and disadvantages. Because SOPC designs use multiple IP modules to optimize the hardware design, this technique allowed us to simplify the system revision and debug process. This approach also allowed us to simultaneously design software and hardware modules.

In this design contest, we acquired hands-on experience and were able to use some excellent hardware development tools. The Quartus® II and SOPC Builder software made it easy for us to modify and change hardware, depending on the application. For example, we could easily add and change the SPI peripherals that are needed by the SD card control and transmission modules. We also feel that taking this design approach is economical—you do not need to buy additional hardware; you just change the Nios II soft core configuration. You can easily build new functions by adding related hardware based on the changed Nios II processor.

However, we do feel that the Nios II software development tools can be further improved. For example, the coding, compiling, and debugging tools are separate, making it difficult for the designer to work efficiently. Integrating these tools would greatly improve the design efficiency.

Using the Nios II development kit, we found it very easy to connect the custom logic components. Additionally, using the tool, we could easily tune several clocks in the system. However, we found it difficult to find a matched system clock. We think it would be a good idea to provide a table of several matched clocks. The Avalon bus interface was simple to understand. The UP3 Education Kit was great. We could do many things on UP3 board. But we hungered for a small LE. – *Jiwon, Kim*

We found that using the Nios II development kit in our design made it easy for us to develop our design with greater efficiency. – *Doehoon, Kim*

We had planned a bigger design at first, but were not able to implement it given time and resource constraints. We hope to do better next time - *SeungChul, Shin*