First Prize

High-Speed Image Evidence Collector Based on Dual Nios II Soft Core Processors

Institution: School of Communication and Information Engineering, Shanghai University

Participants: Lu Xiaofeng, Wu Bainian, and Huang Yan

Instructor: Lu Hengli

Design Introduction

Currently, the laser velocity measurement forensics system used in road transport systems includes a velocimeter, charge-coupled device (CCD) camera, image card, and PC. Among these, the CCD camera, image card, and PC are components of car image forensics systems. Because car information collection and the forensics of road transport systems are carried out on site, timely communication with the traffic authority and cooperating institutions is important. Therefore, the system must be able to send road transport system data accurately, in real time. A system's real-time performance enables quick data collection and processing. Accurate data ensures clear images, which can be used by law enforcement agencies, and the convenience of data exchange enables effective cooperation between agencies. Because the system needs a human/machine interface, the system is too complex for portable, real-time operation. The solution is a system that makes it easy to collect and transmit transport system data in real time.

Keeping in mind the need for a compact system, we focused on the miniaturization of the original car image forensics system comprising a CCD camera, image card, and PC. We wanted to design a single image processing system with functions like multipath image collection, intelligent processing, intelligent display, and wireless transmission with a laser velocimeter to fulfill the need of monitoring forensics tasks for a road transport system. By adopting Altera's system-on-a-programmable-chip (SOPC) solution, our forensics system contains the functions of dual-way asynchronous image collection, pre-storage, ASIC high-speed JPEG compress/uncompress, compact flash (CF) card picture storage, picture searching, multiple display modes such as single-way image and picture-in-picture, real-time character overlay and OSD, and remote wireless transmission.

With the Nios® II soft core processor, we can overcome design issues such as low-speed motion compensation unit (MCU) processing, limited peripheral resources, difficult I/O configuration, complex hardware design, and software programming. Using two Nios II soft core processors, we can distribute control and access to multiple peripherals logically, so that communication between them are coordinated. This design also meets the requirements of time sequence and functions, makes the best use of the processor's resources, and greatly improves the overall operational efficiency of system. Because the system has external memory and I/O, memory access is frequent. Through the Nios II processor's user-defined peripherals, user-defined logic, and direct memory access (DMA), memory access and data movement are simplified when accessing SDRAM, SRAM, and flash memory. Additionally, we can implement real-time data overlapping and OSD. In practice, the road transport situation is subject to frequent changes requiring constant system upgrades. However, while making system upgrades, you cannot change the system hardware due to cost and time issues. By combining the requirements of both software and hardware in a coordinated development process, Altera's SOPC solution offers the best choice for the team because this solution can fully showcase the advantages of a multiple soft core processors in combination with an FPGA's logic control and data processing capabilities. This design approach allows for a flexible system configuration and simple and convenient development, supports various processing modes, and offers powerful data processing capacity at low cost.

With China's auto industry forecasted to post high growth, a highly integrated and portable forensics system for cars has a bright market future. Because the system can fulfill mobile random forensics and offer a real-time position fix, it is very useful for public security and road transport departments. Additionally, the system can perform functions such as information collection, processing, and transmission of data on legally registered automobiles, thereby enhancing timely and safe operation of the road transport system. Our design results in an effective monitoring and forensics system for cars, which is a crucial component of an intelligent transport system.

Function Description

Our system collects, synchronizes, displays, and stores dual-camera asynchronous images. These functions include the collection of images based on long shot and close shot distances, and close shot images with detailed features. All recorded images are transformed based on modulus 8-bit YUV (CCIR-656) format.

Customized I^2C bus peripherals with the Nios II soft core processor control the two-way video ADC's initialization. The Altera® FPGA's on-chip hardware logic elements handle image collection, synchronization, and memory read/write functions. The two-way asynchronous images are synchronized using two blocks of 8-Mbyte SDRAM cache. These SDRAMs store each field of two-way images within the three-field image time, respectively, and then alternate read-write of two blocks' memory cache. At the same time, the program advances to the next memory location, that is, by writing memory cache B while reading memory cache A, and writing memory cache A while reading memory cache B. While writing into memory cache, each field of two-way images is written to within a three-field time, but only one-way image data is read while reading the cache.

We can display these two-way images separately or within a picture-in-picture (PIP) configuration. We display these images in real time and skip caching image data when it is one-way image display. Data is updated and displayed in real time while it is being written into storage memory cache by the method previously described. When the desired main image display format is displayed as PIP, the image data of the main picture is not cached and the sub-picture data is the field extraction image by alternate read operations from the SDRAM cache memory. According to the control mode, if you consider the close shot image as the main picture, the sub-picture data is read from cache B and the main picture image is computed from all updated fields while the sub-picture image is updated for every three fields. In contrast, if you consider long shot image as the main picture, the sub-picture data is read from cache A using the mode previously discussed.

Because each sequential burst maximum read/write capability is limited to 512 bytes, the external SDRAM cache selected cannot write the interlace field data of 720[x]288 resolution into SDRAM in real time. We solved this problem by designing two dual-port RAMs (using FPGA logic) to act as a cache memory block of external SDRAM. This scheme ensures that SDRAM continuously reads/writes a whole-line of image field data in real time.

We implemented the dual-port RAM design through logic control elements in the Nios II soft core processor and the first block of the FPGA.

Real-Time Character Overlapping

When the system monitor starts to capture image data manually, the system starts processing data on the spot. If the system monitor uses an automatic process, the system performs a fixed-site execution. The system monitor communicates with the laser velocimeter through an RS-232 interface. When a speeding vehicle is monitored, the laser velocimeter provides information such as vehicle speed, distance from the velocimeter, and current time by series. The monitoring system compares this information in real time against the stored images database for evidence purposes used by law enforcement agencies. All of this data is displayed on the LCD.

Before storing the compressed images for display purposes, we used the Nios II soft core processor to translate it into a regional code character library and stored it in the external flash memory. We then stored the images in the dual-port RAM of the FPGA. Next, using suitable logic circuits, we were able to read this character lattice from the RAM and overlap it on the image to be compressed and stored. This form of the DMA mode improves the operational efficiency of the system. During overlapping, the image color is pre-defined and overlapping characters are changed into grayscale (because only letters containing one type of color are overlapped), and 1-bit character information is read every time. The character information corresponds to a byte of image data. Due to the overlapped grayscale image, Y is assigned the least grayscale value if some characters must be overlapped. If they do not overlap, Y is reduced by a 32-bit grayscale grading to highlight overlapped characters.

We created this grayscale overlapping function using logic control elements of the Nios II soft core processor in the first FPGA. In doing so, we bypassed the costly traditional character-overlapping function module. After this operation, the system passes image data that have been synchronized and overlapped with necessary characters to the FPGA for further processing and display.

Early Image Memory

The system monitor must support an early image capture memory function because of the laser velocimeter instrument and the time delay generated during RS-232C communication between the laser velocimeter and the system, as well as requirement of law enforcement in traffic system. In other words, the system must be able to store image information of a T-1 period or even earlier according to T-period requirements. Keeping this crucial design requirement in mind, we designed our system to control the image cache using a large external SDRAM to ensure enough image storage precision to meet the law enforcement requirement when storing images. The system applies cache A and B during a two-way asynchronous synchronizing process, during which time the early memory storage function is also completed. The early image capture memory process is related to the size of the cache and the address position data read from the cache. In theory, the system can implement an intense early memory function if the cache is large enough. For the current design, because the image time comprises three fields of storage on cache plus the expected execution time of the Nios II soft core processor during control compression and read/write into CF card memory, the system can guarantee an early memory storage requirement of greater than 0.3 seconds.

Image Compression/Decompression

Because the images are available in the CCIR-656 format, we directly adopt a scheme turning one line into two lines, instead of computing direct interpolation values for mobile images to change the image format from 640[x]240 to 640[x]480 to ensure image-memory quality. Although, the scheme of one line turning into two lines adds extra edges to the image, it keeps the definition of the compressed image. In addition, the size of every image can be up to 600 Kbytes or more, which is a great load for memory and radio transmission. Hence, we must compress the images in advance. The system adopts JPEG compression based on the ZR36060 device with about 83% compressibility (affected by image data). The compressed JPEG picture occupies only 100 Kbytes per image or more with higher compressibility requirement. However, the reserved images comprise one data field of the interlineations scan odd/even field. These compressed images, improve memory speed and efficiency of image processing and meet the requirement of image radio transmission. The chip also supports a decompression function.

The Nios II soft core processor performs initialization and compression/decompression timing, and commands the ZR36060 compression chip by way of customized peripherals. The compression chip operates under the control of the Nios II processor and logic elements, and the compression function is realized by two blocks of external SDRAM (cache A and B) as well as RAM within the FPGA. Cache A and B, respectively, store a field of distant and close shot images during asynchronous image synchronizing. If a close shot image is required for compression, the system reads data of the close shot image from cache A to FPGA RAM, and then the logic elements send data to the ZR36060 for JPEG compression. In this case, cache B only stores one field of distant image and close shot image in a six-field data time. By contrast, if a distant image is compressed, the system reads the distant image data from cache B to FPGA RAM for compression. If PIP image is required for compression, e.g., a close shot image of the main picture, the system first reads the close shot image from cache A to interior RAM, and then reads the distant image of extraction frame from cache A to the specified place of RAM. When logic elements read and compress data from the dual-port RAM, the system reads the scrambled image in PIP so as to implement the PIP image compression. In this case, cache B only stores one field of distant image and close shot image data within six-field data time.

Note that image decompression of stored images in the CF card is a reversible process of the previously described compression process.

When the system is compressing images, no real-time image data is sent to the FPGA and cache C and D for processing. This process enables the FPGA to repeatedly send image data in cache C to cache D and LCD to freeze the display picture. Then, the system decompresses the latest data from CF card, delivers it to the FPGA by data bus transfer, and writes to cache C. The FPGA later reads the latest image data from cache D and freezes them onto LCD. So every time image memory data is processed manually or automatically, cache C sends the latest decompressed image data to cache D. The images are frozen on the LCD from the latest image data of cache D to avoid frame-skipping and black screen during compression/decompression and image storage processes, thus guaranteeing a continuous and smooth image display. In addition, this operation also involves updating and intercommunication with the FPGA RAM when the Nios II processor operates cache C and D.

The function was realized by the control and peripheral access functions of data and command communication between two Nios II soft core processors in two blocks of the FPGA. Accessing the compressed chip was performed through the customized parallel input/output (PIO) mode of the Nios II processor.

CF Card File Management System

Because images must be stored in real time during image collection, and a great number of images need to be stored while monitoring speeding instances, the system needs substantial memory. Therefore, the monitoring system must support the right kind of memory medium featuring large capacity, low cost,

and easy operation. Our system uses a commonly available CF card with a 256-Mbyte capacity, which can store more than 1,000 groups of pictures. Combined with the LCD display, the CF card can be used on our system for browsing and searching images or on a PC for easy operation.

The Nios II processor starts initializing both the CF card and compression chip, and builds the catalog contents and file allocation tables (FAT) of the currently stored files in the CF card. Next, compressed images can be stored onto to the CF card under the control of Nios II logic elements. However, the Nios II processor must update catalog contents and FAT of the second picture before the second picture is stored.

CF card file management involves communication between the two Nios II soft core processors and the CF card during the storage and read of CF card data. The data channel is connected directly, and the command channel depends on I^2C bus. The communication mode responds according to the compression process for access and display of external memory data. All keystrokes used by the control process adopt the I^2C bus access mode.

According to the requirements of the traffic system, the monitoring system must be able to store data similar to the size of an image under question to track aberrant behavior on the spot. This feature helps the inspector working with the system to track down an over-the-limit speeding vehicle with high-quality pictures. Using this principle, we designed two memory modes in our system: one mode for two pictures and the other for three pictures. The system can randomly select two or three pictures to store in a single image or PIP image under these two memory modes. In addition, the monitoring system can browse, search, cancel, and format the stored pictures on the CF card. The system LCD display is user friendly in that it hints all possible operations (including OSD display mode features), its visualization, and possible shortcuts.

File management on the CF card is realized through control and peripheral accessing functions of the Nios II soft core processor as well as the command communication function of the FPGA. We have implemented the I^2C bus and CF card interface through a customized PIO mode.

GPRS Wireless Transmission of Images

Because a road traffic system involves intense communication between law enforcement units and related locations, our system features a peripheral general packet radio service (GPRS) module. This module communicates with the RS-232 serial port and the Nios II soft core processor of the first FPGA, and enables wireless transmission of images involving traffic violation (venues and vehicles). The module allows users to browse the pictures stored on the CF card directly, and transmit them as forensic evidence to fixed IP addresses on the Internet through a GSM network. This function was mainly realized through accessing serial port peripherals by the Nios II soft core processor in the first FPGA device.

OSD Display Mode

Because we need to display status information of the CF card, GPRS module, and user menu on the LCD screen along with the image, we adopted the OSD display mode for easy operation.

The OSD overlapping character comprises two parts. The first part deals with the law enforcement location and the enforcement code number on the upper screen. This data is sent to the first FPGA via a serial port connected to the front-end laser velocimeter to obtain information. Then the main processor in the first FPGA sends this data to the subordinate FPGA via the I^2C bus to implement the OSD display. The second part deals with the status and menu prompt information. Information in this mode is obtained using direct data transmission between on-chip and off-chip memory under the Nios II soft core processor control. This operation is similar, in theory, to DMA mode, but it does not use DMA peripheral control directly. Instead we use the external SRAM's custom-defined write logic. When data

is translated into RGB format from YUV format in the second FPGA, the Nios II processor begins to write OSD content stored in the external flash memory to an external SRAM chip via the internal RAM of the FPGA. Then, the logic control unit takes in OSD data read in the external SRAM to the transmitted RGB image, and displays it in OSD mode. The Nios II processor modifies the OSD content character bitmap in the external SRAM via the internal RAM of the FPGA during each data access.

The OSD data identifies an overlapped word data with a 16-bit word. Because the text displayed in our current system is black and white, we only use three bits. If necessary, we can use all 16 bits to display the text in color. The Nios II processor uses custom logic to write the external SRAM. We have used a DMA-like mode when data in the external SRAM is transmitted to internal RAM and overlapped under Nios II processor control.

The logic unit translates the image format, from YUV to RGB, before OSD affected in the second FPGA. During this process, we implemented line interpolation to create smoother images for easy viewing of the display.

Performance Parameters

These parameters are the actual performance parameters of the system, which were obtained after implementing the system design. The actual system performance parameters are:

- *Power supply*—DC voltage: 9 – 12 V; operating current: 600 mA (motherboard current is about 200 mA); power consumption: 10 W
- *Environment*—Operating temperature: 0° C – 40° C; relative humidity: 8% – 95%
- *Input image*—Mode: two-channel asynchronous image; data format after digitalization: CCIR-656
- *Display*—LCD resolution: 640[x]480; LCD display area: 16 cm² or 6.4 inch²; LCD display color depth: 6-bit/RGB
- *Storage*—Storage image resolution: 640[x]480; storage image color depth: 8-bit, JPEG; storage image capacity: about 100 Kbytes/image; image storage total cycle: < 0.5 s/2 images
- *Transmission time*—About 1 minute/image (depending on local network condition)

We have used two Nios II soft core processors in this design to manage the FPGA's internal resources, define the time sequence requirements for data processing, and handle display and control of the system. In addition, we needed to make frequent access to multiple peripherals from the main system and the Nios processor helped us to improve the overall system operational efficiency. The dual-core system fully utilizes all features and performance of the Nios II soft core processor. The following functions were included in the design:

Functions of the Nios II soft core processor in the first FPGA:

- Main processor, controlling logic unit for the whole system.
- Performs intra-soft core communication with Nios II soft core processor in the second FPGA via user-defined I^2C bus peripheral.
- Handles data and command communication with front-end laser velocimeter through RS-232 serial port peripheral.

- Handles all operation instructions to FPGA internal logic circuits through user-defined PIO peripherals.
- Acts as the main component of I^2C bus in the whole system, controlling all subcomponents on the I^2C bus in the system, such as image analog-to-digital (A/D) conversion chip, user buttons, and Nios II soft core processor in the second FPGA.
- Communicates with the GPRS module via RS-232C serial port peripheral.
- Converts the standard region code character library in flash memory into character dot matrix.
- Issues partial control instructions during image compression and CF card storage while initializing the compressed chip and CF card at the same time through user-defined peripherals.
- Performs file allocation table (FAT) file management system of CF card via user-defined peripheral.

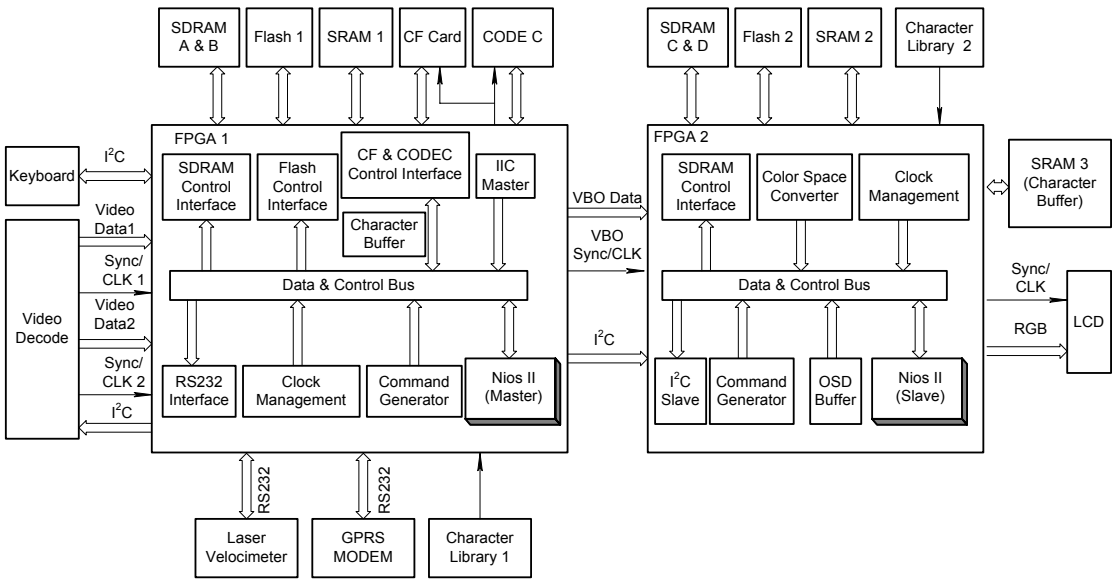
Functions of the Nios II soft core processor in the second FPGA:

- Subprocessor.
  - Handles intra-core communications with main Nios II soft core processor in the first FPGA via I^2C bus.
  - Issues all operation instructions to FPGA internal logic via user-defined PIO peripheral.
  - Implements self-defined logic to write into external SRAM.
  - Handles DMA data access mode between external SRAM data and on-chip logic.
  - Converts standard character library in Flash memory into character dot matrix.
- Implements OSD display.

Design Architecture

Figure 1 shows the system block diagram.

Figure 1. System Block Diagram



Design Methodology

This section describes the design methodology we used for the system.

System Function Design & Test

Using Altera's UP3 development board, we were able to design most of the system functions, test, and simulate all functions. In addition, we designed a few circuit boards for design and test. We did our system design and testing as outlined in the steps below:

1. Design the technical parameter and function indexes that need to be implemented by high-speed image storage.

2. Refer to examples and testing documents on UP3 development board, understand Nios II soft core architecture, software programming using C language under IDE environment, and online program debug and programming for the device.

3. Adopt SOPC Builder to implement peripheral storage access, RS-232C serial port, and PIO on UP3 development board.

4. Implement access of hardware through IDE interface and setup file management system on UP3 development board (the IDE interface on UP3 development board also defines the sequence error of address pinout definition).

5. Implement user-defined peripherals and logic on UP3 development board.

6. Implement collection, compression, decompression, and CF card storage for a single processor image.

7. Implement real-time character overlapping for single processor image.

8. Realize LCD OSD display.

9. Realize synchronization of dual-processor asynchronous images.

10. Realize picture in picture, collection, compression, decompression, and CF card storage for dual-processor asynchronous images.

Hardware Implementation

Our hardware design task was to combine the above-mentioned testing methods realized on UP3 development board and our circuits, then design the necessary hardware modules, build a high-speed image forensics system based on dual Nios II soft core processors, and implement the above functions.

Using the PROTEL 99SE tool, we were able to partition the design using the UP3 development board and our own circuit modules. This enabled us to decide upon the usage of two Cyclone® EP1C6Q240 FPGAs as controlling devices. All other functional modules in the system were designed based on these two FPGAs, which represents a high SOPC-integrated design concept and principle in hardware. Because the Nios II soft core processor is available to be designed in both FPGAs, only peripherals such as SRAM and flash devices needed to be configured. Due to the design requirements of image buffer and freeze, a high-capacity SDRAM needed to be configured as a buffer. We also needed to perform JPEG image compression; so we decided to use an ASIC to perform compression. Our design treats CF card as storage media in order to implement image storage, and we chose the CF card to make the storage media easy to use. The dual-processor simulated image had to be digitized to achieve effective processing. Therefore, the system had to configure the A/D module for both single and dual-processor images. Other system peripherals include a power management unit, LCD interface, two serial ports (for communication with front-end laser velocimeter and GPRS module), and operational buttons.

To make it easy to understand the system hardware design process, we have split the design into the schematic diagram, functional verification of the schematic diagram, component purchasing, and PCB development, as described below.

1. *Schematic diagram design*—Because most of the functional testing and simulation has been completed on UP3 development board and self-designed circuits, the schematic diagram design mainly refers to our own circuit designs that effectively combine these into the most representative of the circuit system schematic diagram. Thanks to the adoption of Altera SOPC solution, all processors and functional control units were integrated into two FPGAs, further simplifying the design structure.

2. *Functional verification of the schematic diagram*—Although most of the functional testing has been completed on testing board, hardware integration needs functional verification. The functional verification of the schematic diagram is primarily to demonstrate the proof-of-the-concept in consideration with the guidance of the tutor. This process led to the confirmation of a complete circuit design.

3. *Component purchase*—We purchased the necessary components while designing the schematic diagram and its verification. All component packages were clearly marked on the schematic, which made PCB development easier.

4. *PCB development*—Our circuit board design uses a four-layer PCB scheme, which is capable of handling multiple signal wires and earth wire to ensure the normal functioning of the circuit board and perform all system functions.

Software Implementation

The software design task is to migrate the VHDL and C language program of the above-mentioned functions realized on UP3 development board and self-designed circuit to two Cyclone EP1C6Q240 FPGAs with two Nios II soft core processors. The design of all software modules was based on highly integrated hardware modules. We used the Altera Quartus® II software, SOPC Builder, and Nios II IDE to build the Nios II soft core processor and to develop the controlling program of the system, highlighting the SOPC solution's highly integrated and programmable concepts. Our software design was based, but did not completely rely, on hardware modules so that we could complete core modification and be able to make upgrades. The design also made it possible to easily implement, add, and remove multiple peripherals, access user-defined peripherals, and design user-defined instructions. Generally, the system is under control of the dual Nios II soft core processors. However, most of the software modules were implemented based on the cooperation between the Nios II soft core processor with the FPGA logic.

We used the VHDL and C languages for the software design. We wrote the logic control and data processing program in VHDL, and used C language routines for the control program of the main and sub Nios II soft core processors. Based on the functional tasks, the system software is divided into system initialization, direct image display, character overlapping of image data, image compression and decompression, image storage, GPRS wireless transmission, and OSD display. The implementation method and steps are shown below:

1. *System initialization*—The system is initialized for all parts in the system by the main Nios II soft core processor via the I^2C bus, which includes video A/D module and user buttons, initialization operation for the CODEC chip during compression and decompression, FAT file management system during CF operation, and directory entry and FAT modification.

2. *Image display*—The image is displayed on the LCD screen before it is captured and stored. The image can be displayed in single-processor mode or picture-in-picture mode. Although the image data seems to be directly available, the system is writing this data into external SDRAM cache unceasingly and repeatedly to meet the requirements of compression and storage in advance. It also refers to the access to external storage operations in user-defined peripheral mode by the Nios II soft core processor. Meanwhile, logic is needed to do a great deal of work for clock management, controlling command generation, and data channel handling to carry out functions set by the system.

3. *Real-time character overlapping*—The character overlapping software design includes translating standard characters into binary lattice information by the Nios II soft core processor, the access time sequence design of the lattice information, and reading/writing of external storage with internal RAM in the FPGA by the logic unit, which is performed by the Nios II processor and FPGA logic.

4. *Image compression and decompression*—Image compression and decompression refer to the initialization of a compression chip by the Nios II soft core processor, generation of control signal for compression and decompression is also provided. Logic handles the processing and transmission of other data.

5. *Image storage*—The Nios II processor plays an important role during the access of the CF card by the system. It creates a complete FAT file management system, initializes the CF card, and issues

control commands for storing and reading processes. Other data processing is performed in cooperation with the FPGA logic.

6. *GPRS wireless transmission*—Pictures stored in the CF card need to be transmitted wirelessly to a fixed IP address at any time based on the users' requirements. In this way, all law enforcement units of the whole road traffic system can share the latest data resources. The Nios II serial port peripheral accesses the GPRS module and to transmit the image data. This implementation provides more convenience for operating and programming the system.

7. *OSD display*—This function is mainly implemented by the sub Nios II soft core processor, including bitmap translation from OSD data, writing of bitmap data to the external SRAM, reading to the internal RAM of FPGA from data of external SRAM, and overlapping of OSD data to real-time image. We used the Nios II soft core processor to access the peripheral, write to the external SRAM in user-defined logic mode, and read/write external SRAM data in DMA mode. In addition, FPGA logic translates the image format, converges image and OSD data, and interpolates the image and data.

Design Features

This section describes the system's design features.

Dual Nios II Soft Core Processors

The master and slave Nios II soft core processors were implemented in two Cyclone FPGAs. The processors implement communication with Nios II-defined I^2C bus peripherals. This design takes full advantage of the dual-core cooperation to coordinate system data processing and display timing requirement, and utilize FPGA internal logics and storage device resources to share data processing, peripheral access, peripheral and internal logic control, enabling CF card file management system, user-defined peripheral access and control, user-defined logic, DMA, and OSD display. With this implementation, we could add extra system functions and control methods, while reducing the overhead of extra control units such as MCU, hardware circuits, and design complexity. This implementation led to lower software and hardware design costs.

Synchronization of Dual-Camera Asynchronous Images

We used I^2C bus peripherals defined by the Nios II processor to control the dual-channel video ADC initialization and schedule the FPGA logic to control image collection, synchronization, and storage. While synchronizing the two asynchronous images, the off-chip and on-chip cache can be used to synchronize the dual-channel asynchronous images, store image information in advance, and realize multiple display methods such as dual-channel and PIP. The direct access to multiple storage devices using FPGA often makes the programming task difficult and takes plenty of on-chip resources. We solved this design issue by using the Nios II processor's fast access to multiple storage device peripherals, which helped to reduce the design cycle time and expense.

Real-Time Character Overlapping

During the image capture process, the capturing system and laser velocimeter communicate through the RS-232 serial port. When detecting a speeding vehicle, the laser velocimeter provides the current speed of the vehicle, its distance from the velocimeter, and the precise time to the capturing system through serial ports. Then, this information is displayed through OSD on top of the screen by the system. Meanwhile, the character information, such as the execution site and force number, are overlapped into the saved pictures in real time to be used as evidence for police action. This design makes full use of the Nios II-defined peripheral functions to enable the system to communicate with the front-end

velocimeter in a timely fashion to obtain character information and instructions needed without having to develop a dedicated serial port driver.

ASIC Compression & Decompression

The system adopts an ASIC-based, ZR36060 device JPEG compression, delivering 83% compression ratio. The device can compress JPEG images down to 100 Kbytes, which not only improves the image storage speed and efficiency but also meets the requirements for image wireless transmission. The chip is also equipped with a decompression function. Its implementation relies mainly on a user-defined peripheral of the Nios II software to initialize the compressed chip and issue control instructions for compression and decompression tasks in coordination with the logic unit.

CF Card File Management System

The Nios II processor starts to initialize the CF card while initializing the compressed chip, and further creates a directory entry and FAT for the files being stored. Then, the compressed images can be saved to the CF card. The Nios II processor configures the CF card for saving every image. We have designed an image storage format of 2 or 3 images in a group. The Nios II software user-defined peripheral function is fully utilized when accessing or storing images onto the CF card, which allows the Nios II processor instructions to complete the complex operations on CF card.

Image GPRS Wireless Transmission

We devised externally connected GPRS modules to connect with the Nios II software and the first FPGA through serial ports to wirelessly transmit speeding vehicle images to the law enforcement, which facilitates communication between the execution units and sites. Users can view the images stored in CF card directly and then transmit these images to a fixed IP address on the Internet through a GSM network. The monitoring execution sites can communicate with each other through GPRS and quickly read the information on condition of the scene and vehicle in question. This communication allows the police to intercept these vehicles quickly, and further monitor the road traffic in real time.

OSD

The system uses OSD to facilitate user operation. It is needed because there are many prompts during system operation, status information of the CF card, GPRS, and various menus that need to be synchronized and displayed on the LCD along with images.

The characters overlapped on OSD include two parts. One is the execution site and police number on the top of the screen. The information is sent by the front-end laser velocimeter through the serial port to the first FPGA. The FPGA's master processor then sends the data to the FPGA for OSD. The other is the status and menu prompts. This information is obtained by transmitting data directly between on-chip and off-chip storage devices under the control of the Nios II soft core processor. This operation follows the same principle as DMA but does not use DMA peripheral control directly. Additionally, this function writes to the external SRAM using user-defined logic. When the data format is transformed from YUV to RGB in the second FPGA, the Nios II processor starts to write the OSD contents stored in external flash into an external SRAM chip via the RAM in the FPGA. The logic control unit overlaps the OSD data read from external SRAM into the RGB image whose format has been transformed for OSD. Then, the Nios II processor modifies the OSD content character matrix in external SRAM through the RAM in the FPGA during each blanking interval of the data field.

OSD data indicates an overlapped data point with one character (16-bit data). Now, only three conditions are used for black-and-white display. If necessary, all 16 bits can be used for color display. The Nios II processor uses user-defined logic to write to the external SRAM chip, while the data in external SRAM under the control of Nios II soft core processor is transmitted into RAM for overlapping through a DMA-like method.

Highly Integrated SOPC Solution

The whole system has been designed around two FPGAs. Making full use of multiple functions and features of two Nios II soft core processors, the control and data processing are managed by Nios II soft core processors in the FPGAs. While the design is processed with the FPGA software, the hardware system and software system can be combined, casting off the traditional reliance of software development on hardware. Additionally, the multiple soft core processors can be used to select appropriate peripherals, storage devices, and I/O interface to build a system that is well-tailored to match customer demands. This design approach also yields low price, simple design, high integration, and low risk.

Easy System Upgrades

Due to the nature of the user requirements and road traffic conditions, the system requires quick and frequent upgrades. The Altera SOPC solution frees users from upgrading hardware for software upgrades. Even if the product has been delivered to a customer, the software can be updated regularly. Users can add new features to the hardware continuously to reduce risks possibly caused by changes in standards and add functions, thus simplifying hardware repairs. They can also avoid processor obsolescence.

Collaborative Software/Hardware Development

Because the hardware and software of an SOPC system can be combined, the design of both can be jointly performed during the system development process. The development of software and hardware can begin and end almost simultaneously, saving a lot of time. Additionally, the SOPC design approach accelerates the pace of product launches and enables a longer product life cycle. You only need to generate a new Nios II kernel if you need to modify some definitions during development process, exerting no impact on other peripherals or other Nios II programs.

Lower System Cost

The Nios II soft core processor solution can help to achieve a good balance between system performance and system cost when designing systems involving higher integration, optional CPU (high-speed, economical, and standard), a multi-CPU system, and no fixed requirements for a processor implemented on an FPGA. Using the Nios II soft core processor, we designed an economical and practical monitoring solution for road traffic systems, while also taking into consideration the need for flexible system functions.

Conclusion

Before participating in the contest, we had some prior experience with Altera devices and possessed some development experience. When we started using Altera's SOPC solution incorporating FPGAs and the Nios II soft core processor, our original understanding of the FPGA changed. The FPGA functions that are single, parallel, and high speed with strong logic, simple time sequence, and complex operation were greatly expanded. We integrated designs that could be implemented by an external control unit into one FPGA, and removed/added the peripherals and interfaces of an internal Nios II soft core processor. Additionally, the Nios II soft core processor can be adapted easily for low-speed control, awkward peripheral access, and massive storage peripheral devices, all of which need frequent interface modifications. Furthermore, in accordance with many features of the FPGA I/O, there is always a problem of operation coordination and interface between the FPGA and control units. You need to take this coordination into account during software/hardware design, e.g., for data processing, command control, time sequence coordination, parallel/serial, and high/low speed. Between the Nios II software processor and FPGAs you can simplify system design and software programming. By migrating the OS into the Nios II soft core processor, you can combine the FPGA and traditional embedded CPU. At the same time, the kernel of the SOPC solution is the most effective.

Accordingly, an integrated SOPC solution and the Nios II soft core processor have totally overthrown the old embedded system design and greatly boosted the hardware/software design gains. Based on this evolution, an embedded system's hardware circuit is simpler and more effective and the software design can be visualized and migrated easily. It is certain that hardware circuit design is not as complex as before and "processor + memory + peripheral" is easy to understand. However, the software design is more complex than before. The hardware/software coordinated development technology of SOPC solution has solved the problem and enabled the synchronous development of FPGA logic and the Nios II soft core processor inner program to enhance the design efficiency.

In the contest, we created the dual Nios II soft core processor control, but did not migrate a real-time OS into the processor due to time limitations and difficult technical challenges. Additionally, our Nios II evaluation is deficient in this aspect. According to the current requirements of embedded systems, the realization of real-time OS migration shall be based on the improvement of system integrated efficiency and stability, which is also the direction of our efforts.

Because the SOPC solution and Nios II soft core processor were launched not long ago, some problems occur in real applications such as process speed deficiency, shortage of IP cores, an internal resource insufficiency of general mid- to low-level FPGAs, and expensive high-level devices, all of which have restrained the application of SOPC solutions in some fields. But we believe that with the growth of related technology, as well as innovations from Altera, system designers will be sharing many practical SOPC solutions before long. When that happens, by virtue of advanced tools, we will then design the most profound system in the most concise way.