Second Prize

Real-Time Driver Drowsiness Tracking System

Institution: School of Electronic and Information, South China University of Technology

Participants: Wang Fei, Cheng Huiyao, Guan Xueming

Instructor: Qin Huabiao

Design Introduction

Our real-time drowsiness tracking system for drivers monitors the alertness status of drivers during driving. The system judges whether the driver is distracted or dozing, and monitors the driver's continuous working hours, vehicle speed, and other information. When the system finds extreme fatigue or other abnormal conditions with the driver's behavior it alerts with a voice warning.

Drowsy driving is an invisible killer for drivers especially while driving on highways. An amazing fact derived from a large number of traffic accidents is that about 20% of traffic accidents are due to drivers' drowsy driving. In addition, drowsy driving is the reason for 22%-30% of severe traffic accidents resulting in death, ranking it as the top of the cause list. So drowsy driving is undoubtedly very dangerous for traffic security. Chinese and international vehicle manufactures are all busy in designing drowsy-driving monitor devices. Most of them have deployed advanced test technology combining embedded systems and intelligent control technology to design their drowsy driving monitor systems, so as to reduce traffic accidents caused by drowsy driving. Our system was designed for the same purpose.

This design adopts the Altera® Nios® II soft core embedded processor and combines image processing and mode identification functions to complete the design of the whole system. We have also used Altera's Cyclone® FPGA to build the necessary peripheral circuits. Taking advantage of the unique self-defined instruction mode of the Nios II processor, we have deployed pure hardware mode to realize image processing algorithms, which greatly improve the system's real-time performance, accuracy and reliability.

The system uses a camera to capture the driver's image, and makes a collaborated analysis to the image information using the processor and an image-processing module before dispatching data to the Nios II processor. The system judges whether the driver is distracted or dozing, and takes appropriate action. For example, it issues a voice warning immediately when the driver is about to doze off. In addition, the system can monitor the driver's continuous driving hours and driving speed. In this case, if the driver has been driving continuously, for example, for five hours or more and/or the driver is over the speed limit set by road management, the system will send appropriate warning signals. Further, the system can also record the driver's status, driving hours, speed and vehicle status and store this information as evidence in case of a traffic accident. In this way, the system can monitor the driver's status in real-time, restrict driver's mistakes while driving, and effectively prevent serious traffic accidents caused by drowsy driving.

The system, which is based on the Altera system-on-a-programmable-chip (SOPC) concept, utilizes the rich logic resources of the Cyclone FPGA to implement complex system-level functions while integrating the Nios II soft core processor. When compared to traditional DSP designs, our SOPC design approach simplifies design complexity, simplifies debugging, and derives a scalable system. The SOPC approach also makes system maintenance and upgrades easy when required.

The system can be used on long distance coaches, trucks, and cars.

Function Description

The following functions comprise the real-time drowsiness tracking system for drivers:

- Judge driver alertness based on the inputs received during day and night driving conditions and deliver the information to the Nios II processor for further processing

- Capture the driver's image in real time, then process and analyze it to judge whether the driver's eyes are open or closed or if he is dozing. If the driver is found dozing while driving, a warning voice signal will be sent.

Due to the strict real-time performance requirement of the system, we implemented the image-processing algorithms in hardware. The implementation difficulty and robustness of the algorithm will have to be comprehensively studied when choosing the algorithm. The basic image processing algorithm can be realized in the following three steps (for a detailed algorithm flow, see Figure 1):

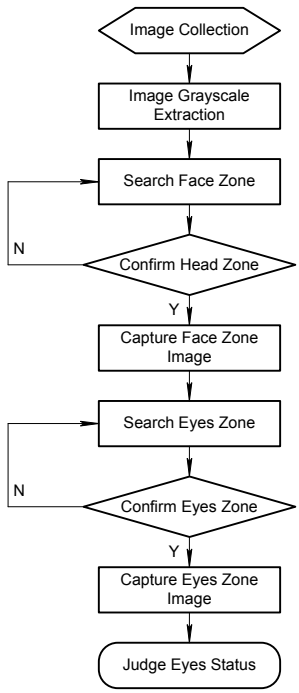
1. *Face-Zone Positioning*—Pre-process the original image of the driver to determine the zone of face and capture it. In this way, the image zone, which needs further processing, can be shrunk to speed up the processing. In addition, this technique also lightens the influence of a complex background for face feature judgment.

2. *Eye-Zone Positioning*—Determine the position of eyes in face zone, and capture driver's image eyes zone for further processing.

3. *Eyes Open/Close Judgment*—Analyze the open/close status and winking frequency of the driver's eyes to determine whether the driver is distracted or dozing.

Figure 1 shows the image processing algorithm flow.

Figure 1. Image Processing Algorithm Flow of Real-time Drowsiness Tracking System for Drivers



Performance Parameters

The following table shows the correct judgment rate testing.

Test Scenario	Sample Number	Correct Judgment	Threshold Value	Correct Judgment Rate	Notes
Day	80	54	1000	67.5%	
Night	60	43	1100	71.7%	
Uniform Background	80	54	1000	67.5%	Day
Complex Background	80	50	1000	62.5%	Day

The following table shows the program response time.

Transmitting Time of One-Frame Image	Head Check Module Processing Time	Head Check Module Processing Time	Head Check Module Processing Time	Total Response Time
33.3 ms	2.56 ms	2.56 ms	0.27 ms	80 ms

The image collection module includes camera and external SRAM. We have used the digital, color CMOS image sensor OV7620 from US-based OmniVision as the image collection device. The OV7620

is a 300k pixel image sensor that is compact, affordable, powerful, and easy to use. The sensor provides high quality digital images with a 10-bit dual-channel A/D converter, which meets the system's resolution requirements. In addition to being a CMOS image sensor, the OV7620 also features the following items:

- 326,688 pixel, 1/3-inch sensitive area, VGA/QVGA format.
- Supplies color and black/white image data in interlace and line scan mode.
- Supports output digital format: YCrCb4: 2: 2, RGB4: 2: 2 and RGB original image data.
- Outputs 8- or 16-bit image data in CCIR601, CCIR656, and ZV interface mode.
- Configurable through internal registers to meet the application image requirements.
- Output window can be adjusted from 4 x 4 to 664 x 492.
- Automatic gain and white balance control, 1 - 500 times automatic exposure range.
- Multiple adjusting functions such as brightness, contrast, gamma correction, and sharpness.
- Operates from a single 5-V supply, power rating of < 120 mW, standby power loss < 10 uW.

Because the OV7620 outputs 8-bit wide data, the external SRAM is based on the IDT71V424 device that has the same data width as the OV7620. The IDT71V424 buffers the driver's image. The IDT71V424 is a 512-Kbit SRAM device with a read/write period of 10 ns, which meets the system requirement.

Design Architecture

Figure 2 shows the system hardware design.

Figure 2. System Architecture of Real-Time Drowsiness Tracking System for Drivers

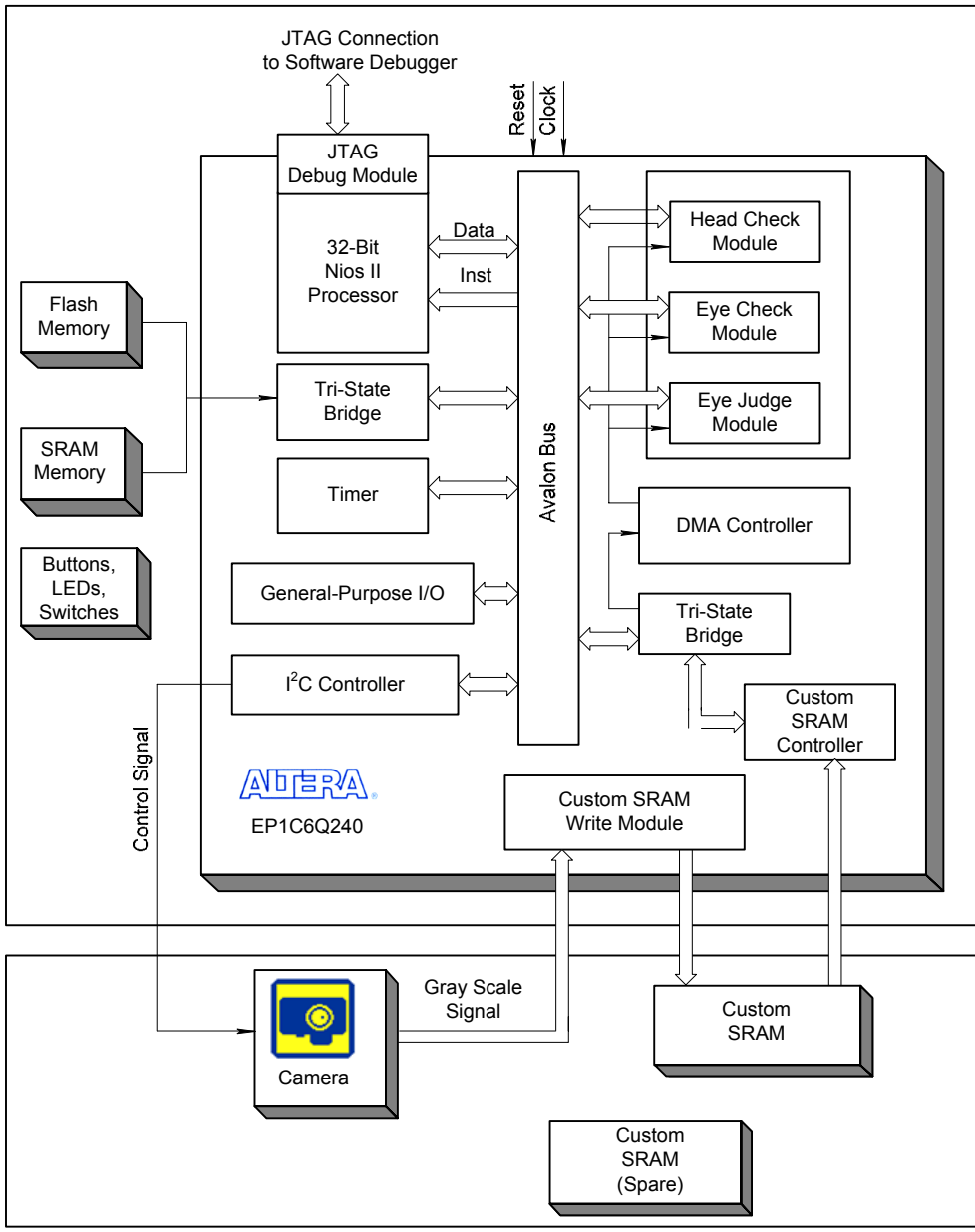
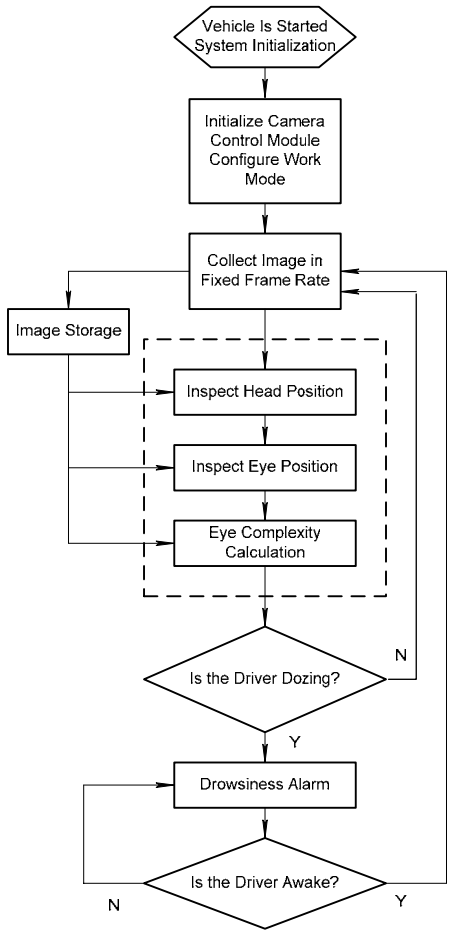


Figure 3 shows the software flow.

Figure 3. Software Flow of Real-Time Drowsiness Tracking System for Drivers



Design Methodology

This section describes the hardware and software design.

Hardware Design of the System

The hardware design consists of the OV7620 control module, the read/write control module of the external SRAM, and the hardware realization of the relevant image processing algorithm (core part).

OV7620 Control Module

We configured the OV7620's internal register with the serial camera control bus (SSCB), to change the camera-operation mode. The SCCB bus, which complies with I^2C Bus specifications, is a popular bus mode used by image sensors, currently. The SCCB bus includes two signal lines, SCL and SDA, which represent the clock and serial data I/O, transmit start, address, stop and read/write data commands.

The OV7620 outputs a 27-MHz synchronization clock with a 320 x 240-resolution image data at 30 frame/second speed. Therefore, it is quite difficult to implement a system at such high data speeds when extracting images by the Nios II processor directly. To solve this problem, we developed an independent OV7620 control module with I^2C Bus control using the C programming language to

configure the camera and read/write the OV7620's control registers. In this way, we can avoid using the SCCB bus. As far as the Nios II processor is concerned, image collection can be realized just by sending start and configuration commands, waiting for the completed signal return from control module, and storing data in external SRAM.

First, the OV7620 is initialized and configured by camera control module via the SCCB bus to operate as a sensor with QVGA format (320 x 240), 30 frame/second, 16-bit, YCrCb4: 2: 2 digital video signal output, default image saturation, brightness and contrast, AGC, and white-balance control. Then the frame synchronization signal VREF, line synchronization signal, and pixel clock signal PCLK, generated by OV7620 are received and are referred to when writing data to external SRAM.

External SRAM Read/Write Control Module

Saving the image data collected by the camera to the external SRAM is a major system design challenge. The image-processing algorithm used by the system has a strict specification with regard to image quality. Further, when we save images in external SRAM, due to cable crosstalk and noise from other components, many transmission errors could occur during SRAM read/write operations.

To solve these problems and enhance system precision, we reduced the system clock from 48 MHz to 30 MHz. Our tests have shown that the system makes almost no mistakes when reading and writing to SRAM at lower clock rates.

As referred to earlier, the pixel clock output by the camera is at 27 MHz. Since the PCB is compact and mounted on a vehicle, even pixel data observed with a 500M-sample logic analyzer appears to be unstable under such a high frequency, let alone the pixel data extracted using the FPGA. Therefore, we decide to configure the camera as a QVGA sensor, 16-bit output, ignoring color difference signal and extracted brightness signal only. In this way, the actual pixel output frequency of the camera gets reduced to 6.75 MHz, and then we can extract stable pixel data using the FPGA.

The SRAM control module, written in Verilog HDL, judges when the data signal is valid according to field synchronization, line synchronization, and pixel clock signal output by the camera, generates the CS, WE, and address signals needed in SRAM writing, and writes pixel data to the appointed SRAM address. Because the Nios II processor and our control module control SRAM, we wrote a multi-path noise suppressor program to assign control signals to SRAM.

Hardware Realization of the Relevant Image Processing Algorithm (Core Part)

The image-processing algorithm of the design was realized in hardware. Using Verilog HDL, all hardware modules were developed and attached to the Avalon® bus as the Nios II processor's peripherals. Each module has its own register and task logic sub-module. The Nios II processor calls the DMA controller to transmit different image data to the three SRAM-based algorithm modules using the following process:

1. The Nios II processor starts the DMA controller, transmits the whole image to the first module (the face check module), responds to the interruption signal from the module after it starts running, and writes operating result from the module.

2. The processor enters the result of the face check module into the second module (the eye check module), starts the DMA controller again, inputs face data, writes the operating module data, finds line position of the smallest point and judges the position of eyes.

3. The processor enters eye data to the third module (the eye-complexity calculation module) via DMA controller. This module outputs the ultimate eye-complexity calculation results to the Nios II processor for judgment.

In the above operation, the threshold values of several operating data in algorithm hardware module are entered by the program in the Nios II processor. These threshold dates were obtained through actual tests while running the system.

System Software Design

The software design comprises camera control and invoking of algorithm modules. The code flow is shown in Figure 3. The flow is as follows:

- The system software first invokes init_button_pio() function to set up a user interrupt service to start the camera control program after a button-interrupt response.
- The function init_camera() initializes the camera and sets its working mode by writing data to camera register.
- The function camera_sent() sends control signals to the camera to send image data.
- The program writes camera data to SRAM in SRAM control module, and then enters the algorithm module.
- The function DayCheck() judges whether it is day or night according to the external input and modifies the hardware input parameter with this value.
- The function HeadCheck(), EyeCheck() and JudgeEye()are three key hardware modules that invoke the following subfunctions, respectively.
- PhEnable(address) is the write reset signal for entered module address parameter.
- Sram2Dma_io(sram_start,data_length,ph_address) invokes the DMA controller, and enters data_length long data to algorithm module address of ph_address from the SRAM address of sram_start.
- Values exchanged between each module are selected according to the output result of DayCheck() function. When set to night conditions, the output MAX_DATA of head check module is selected as the driver's head starting position, and MIN_DATA as the driver's head ending position. This is reversed in a daytime situation. The EyeCheck() function selects the smallest point as eye position based on the day/night entry value, and it is set to the second point at night, and third point at daytime. In addition, the complexity threshold in JudgeEye() is also modified according to the day/night entry.
- The function SendWarning() controls the lamp according to the judging value. The lamp is on and sends warning if the driver's eyes are closed, and it will be off when the driver's eyes are open.

Our design debugging process proceeded as described below:

Early-term—System programming, algorithm research. We made the plan, specified the functions realized in this plan, and partitioned the functions separately as to be realized in hardware or software modules. We collected the camera materials, researched image identification algorithm, and adopted the most appropriate scheme for the design.

Mid-term—Write program, implement tests. This area was divided into three parts:

1. *Camera control*—Read and writer registers via I^2C Bus using the Nios II processor and enter the camera output image to PC for observation and analysis. Adjust the focus, analyze the image quality and brightness, and make preliminary plan on algorithm parameters and threshold figures.

2. *SRAM read/write*—By means of image comparison by the camera and recovered image from SRAM, verify the read/write logic and final clock frequency.

3. *Algorithm modules*—The algorithm plan adopted in this design comprised three modules. A certain number of images were selected, and were divided into two parts: eye open and eye close. MATLAB simulations were made on each modules to confirm the operation results and generate ModelSim-simulated **.txt** document and hardware processed **.bin** document. Compile HDL files and make time sequence simulation on documents generated by MATLAB in a ModelSim environment. Then download **.bin** document to flash storage, read image data (content of **.bin** document) to SRAM by invoking the HAL function, simulate the output data of the camera, invoke DMA to transmit data to run in hardware, and use SignalTap® II to observe the resulting data.

Then, we compared the two simulation results with hardware data output, debugged, and selected the appropriate thresholds.

Late-term—Combined adjusting. Because each part provides image data to the next module in mid-term, each module ran under normal mode under the condition that the image data has met the requirements of the lower-level module. We debugged the system hardware under bright and dark light, day and night conditions in a complex environment to determine the threshold parameters.

Design Features

The design features are as follows:

- System image processing adopts a pure hardware approach that relies on parallel processing and pipelined technology. Compared to the DSP system that completes corresponding processing algorithms based on serial instructions, our system greatly reduces the speed of image data collection and processing, which meets the real-time requirements of the system.

- During the processing and judging for the image information, multiple modes of identification algorithm were adopted comprehensively, which made image processing flexible. This approach enhanced system accuracy and reliability.

- SOPC Builder not only provides abundant intellectual property (IP) resources and parameter selection, but also supports user-defined IP. This design approach helped us to realize the algorithm using the hardware description language method and enabled us to define the IP of the algorithm as special instruction via self-defined instruction mode. We used this approach to invoke software modules when implementing repeated operations during image processing and thus reduced system processing speed.

- The Nios II system based on an FPGA offers extra design capability and extensibility, which can be used to adjust the algorithm complexity and balance the robustness and real time performance according to application requirements. We can also use this flexibility to load or remove input and output modules and other peripheral devices according to application requirements, such as an on-board entertainment module, anti-theft module, and GPS positioning module.

- This design adopts DMA as transmission mode, thus ensuring data input data of one pixel in a clock cycle, which remarkably speeds up data transmission. DMA transfers save almost 75% of transmission time when compared with using the Nios II processor for transmission. In addition, DMA does not utilize any bus resource. Therefore, we could use the Nios II processor to implement other functions at the time of DMA transmission.

Conclusion

The Nios II soft core processor represents a brand new concept in embedded design and overturns many of the traditional embedded system design concepts. The three-month contest provided us with a better understanding of this design approach.

Based on the Nios II processor as its core element, the embedded system design platform features remarkable flexibility. SOPC Builder provided us with abundant peripheral resources. The fact that we could write our own peripheral modules freely by adding self-defined logic was of crucial importance to us. For instance, in our real-time drowsiness tracking system for drivers, the I^2C bus module that controls the camera and the module implementing the image processing algorithm were all integrated in the Nios II processor and managed by the Nios II processor to be the peripheral of Avalon bus. Using self-defined logic, instead of writing logic modules that are independent of the Nios II system, could speed up the design flow and improve system compatibility.

The advanced debugging tools embedded in the Altera system impressed us a lot as well for it give us a big help in system design. Compared with the general logic analyzer instrument, the SignalTap® II embedded logic analyzer is more flexible and stable. The key difference was in that we could observe the floating situation of internal signal in the FPGA, and debug the logic compiled by ourselves in an intuitive mode. Besides, we often found a certain logic module worked normally in a single synthesis but failure occurs when it was synthesized with other logic modules. LogicLock™ technology solves this problem well. The powerful design and debug tool made the designing easier and more convenient. For us, the biggest improvement in the Nios II processor is the introduction of the integrated development environment (IDE) which provides a unified visualized interface for program, debug, and download operation. Thanks to this IDE, there was no need to memorize the complicated compile, link, and download instructions.

It is a big challenge for us to write the image-processing algorithm operated on the PC as a hardware module and integrate it into the Nios II system. Although the Real-time Drowsiness Tracking System for Drivers in this text is only a rudimentary application that needs further enhancements, it is a basic design that inspects driver's alertness in driving. In addition, the excellent upgradeability of the Nios II processor will make it easy for us to improve the algorithm and optimize the system in the future.