Second Prize

Set-Top Box Capable of Real-Time Video Processing

Institution: **Xi’an University of Electronic Science and Technology**

Participants: **Fei Xiang, Wen-Bo Ning, and Wei Zhu**

Instructor: **Wan-You Guo**

Design Introduction

The set-top box (STB) played an important role in the early development of digital TV. As the technology develops, digital TV will undoubtedly replace analog TV. This paper describes an FPGA-based STB design capable of real-time video processing. Combining image processing theories and an extraction algorithm, the design implements real-time processing and display, audio/video playback, an e-photo album, and an infrared remote control.

Background

The television, as a home appliance and communications medium, has become a necessity for every family. Digital TV is favored for its high-definition images and high-fidelity voice quality. The household digital TV penetration rate in Europe, America, and Japan is almost 50%.

In August 2006, the Chinese digital TV standards (ADTB-T and DMB-T) were officially released. In 2008, our country will offer digital media broadcasting (DMB) during the Beijing Olympic games, signaling the end of analog TV. However, the large number of analog TVs in China means that digital TVs and analog TVs will co-exist for a long time. The STB transforms analog TV signals into digital TV signals and it will play a vital role in this transition phase.

There are four types of STBs: digital TV STBs, satellite TV STBs, network STBs, and internet protocol television (IPTV) STBs. Their main functions are TV program broadcasting, electronic program guides, data broadcasting, network access, and conditional access. Most STBs in the market use ASIC chips.

This paper provides an FPGA-based STB design that implements real-time processing, display, and playback of audio/video as well as e-photo album and music playback functions. The system uses system-on-a-programmable-chip (SOPC) technology with a Nios® II processor embedded in an FPGA.

The design enhances the system's flexibility and integrity, shortens the development period, and improves efficiency. Additionally, it is highly scalable, easy to update, cheap, energy-efficient, and highly competitive.

Development Platforms

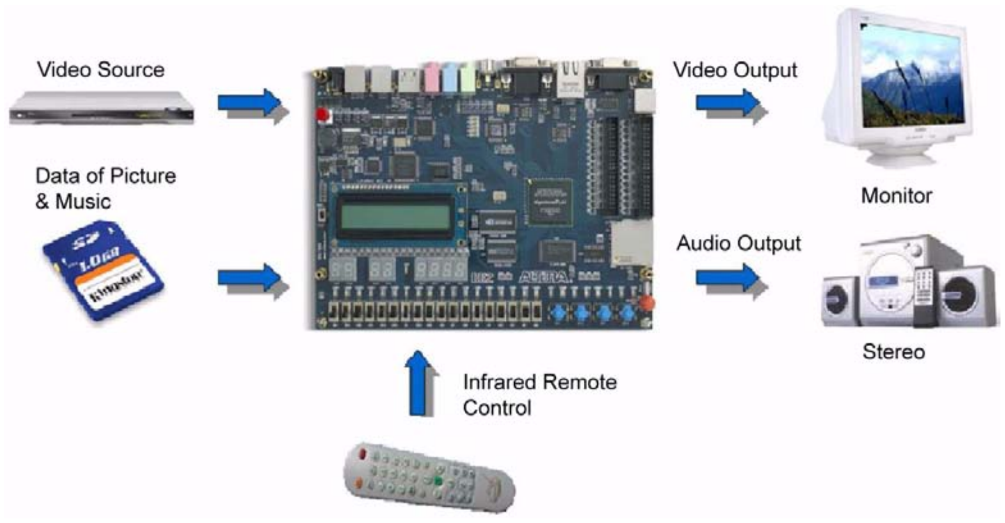
We used the following hardware and software development platforms to build our design.

Hardware Platform

We used the Altera® Development and Education (DE2) board as the core platform. The monitors, speakers, and infrared remote controls are peripherals. The design processes and plays video information input by the video source in real time, and can simultaneously display pictures and play audio/video data stored in the secure digital (SD) card.

The DE2 board provides a variety of resources, including the Cyclone® II EP2C35672C6 device (which has 105 M4K RAM blocks), SRAM (512 Kbytes), SDRAM (8 Mbytes), flash (4 Mbytes), an SD card interface, a VGA digital-to-analog converter (DAC), TV decoder, Infrared Data Association (IrDA) transceiver, etc. Figure 1 shows the hardware platform.

Figure 1. Hardware Platform



Software Platform

We used the Altera Quartus® II software, SOPC Builder, and the Nios II Integrated Development Environment (IDE) to create the logic and program files. The Quartus II software is the fourth-generation development tool for programmable logic devices, providing functions ranging from design entry to device programming. The SignalTap® II logic analyzer, which is provided with the software, is a system-level debugging tool that captures and displays real-time signal behavior. It allows engineers to observe the interactions between hardware and software in system designs.

SOPC Builder is an automatic system development tool. It enables the designer to add processors and peripheral interfaces to the Nios II processor. Additionally, SOPC Builder can automatically generate logic that the on-chip bus and bus arbiter need to connect microprocessor cores, peripherals, memories, and other IP cores, which facilitates high-performance SOPC designs.

The Nios II processor is a configurable, versatile, RISC embedded processor. It can be embedded into Altera FPGAs, and users can adjust the features and performance of the embedded system according to their needs. The Nios II IDE accelerates software development.

Using these software tools and SOPC concepts, we implemented an FPGA-based STB demonstration system that is capable of embedded real-time video processing.

Function Description

This section provides a functional description of our design.

Video Processing and Display

The digital TV resolution and system are different from those of analog TV; therefore, the STB must convert between digital and analog TV resolution. The system converts the resolution and aspect ratio, captures pictures of the original video, and outputs it. During the output, several display effects, such as video-in-picture and picture-in-video, appear.

The original video provided by the video source is input by the audio/visual (AV) interface and is decoded by the TV decoder chip (ADV7123). It passes through the FPGA internal hardware logic for color space conversion, which converts YCbCr signals into RGB ones. Then, the system starts video processing, sends the signals to the video buffer, displays them, and outputs them via the VGA interface.

Video processing includes:

■ *Aspect ratio conversion*—Images convert between the 16:9 and 4:3 aspect ratios in video output. The 16:9 ratio meets ergonomic requirements, and 4:3 satisfies traditional programming needs.

■ *Image freeze*—The user can freeze images in the video output to view them for a long time.

■ *Image capture*—During video output, the user can zoom in on a selected part of the image, and display it on the whole screen.

■ *Picture-in-video*—To enrich the effect, the user can display pictures while video is playing. For example, if an annoying advertisement appears, the viewer can watch pictures as an alternative. Or, the user can adjust the display positions of pictures and videos.

■ *Video-in-picture*—In this case the user can display video with pictures. For example, the user can set pictures in the background when the video resolution is low.

■ *Video resolution conversion*—This function converts the original 640 x 480 video resolution into 120 x 60 or 120 x 96 to meet display requirements.

Menu Display

The menu display helps users learn which information is in the system, makes the system user friendly, and designs the menu's on-screen display (OSD). This function shows the system name, resolution, aspect ratio, system, and other information.

During operation, the Nios II processor determines the video status and generates the menu information according to the system. If the system is in 4:3 mode, it generates the menu information for 4:3 mode; if it is in 16:9 mode, it generates the corresponding menu information. Then, the Nios II processor sends the information to the menu buffer, combines the menu data with the video data, outputs them through the VGA interface, and displays them on the monitor.

E-Photo Album

Most digital devices today use external memories (such as an SD card) to store pictures and music. Our system can read the picture and audio information of external data sources. When there is no video input, the user can view pictures independently; otherwise, the user can view pictures as video-in-picture or picture-in-video. Additionally, users can enjoy music while viewing pictures.

The Nios II processor reads the pictures stored in the SD card and sends them to the SRAM, overlapped with video data in real time, and outputs them through the VGA interface. The Nios II processor also reads the audio data on the SD card while reading pictures. It performs digital-to-analog conversion and outputs the audio through the audio port.

Infrared Remote Control

We designed an infrared remote control to help users control the system. The infrared remote control sends out control signals, which are decoded into relevant pulse signals by the infrared remote control logic module. Signals are then sent to the Nios II processor, which converts them to level control signals that control each hardware module, thus controlling the system.

Performance Parameters

This section describes the performance parameters for our design.

System Performance

This section describes the system performance parameters for video, picture, audio, and the menu.

Video Parameters

The video parameters are:

- *Input video system*—NTSC system
- *Output video format*—RGB format
- *Input video resolution*—640 x 480 pixels
- *Output video resolution*—120 x 60 and 120 x 96 pixels
- *Video color*—18-bit color, 6 bits each for red, green, and blue
- *Aspect ratio*—4:3 and 16:9
- *Frame rate*—60 frames per second (fps)

Picture and Audio Parameters

The picture and audio parameters are:

- *Picture resolution of the e-photo album*—240 x 180 pixels
- *Picture format of the e-photo album*—Bitmap (**.bmp**) format
- *Picture color*—15-bit color, 5 bits each for red, green, and blue
- *Audio format*—**.wav** format

Menu Parameters

The menu parameters are:

- *Menu resolution*—207 x 32 pixels
- *Menu color*—3-bit color, 1 bit each for red, green, and blue. The first line of the menu is the menu; the second line is for information display.

Use of System Resources

This section describes the system resources our design uses.

Use of FPGA Resources

The FPGA resources used are:

- *Logic units*—33%
- *On-chip memories*—79%
- *Pins*—47%
- *Phase-locked loop (PLL)*—50%

DE2 Board Resources

The DE2 development board resources used are:

- *FPGA*—Altera Cyclone II EP2C35 FPGA
- *Dynamic memorizer*—8-Mbyte SDRAM
- *Static memory*—512-Kbyte SDRAM
- *Flash memory*—4-Mbyte flash memory
- SD card reader
- 50- and 27-MHz crystal oscillators
- Switch, button, LED, seven-segment numeral tube, and LCD
- Infrared transceiver
- XSGA video port, XSGA 10-bit digital-to-analog converter (DAC)
- TV decoder chip (supports the NTSC system)
- 24-bit audio CODEC

Auxiliary Resources

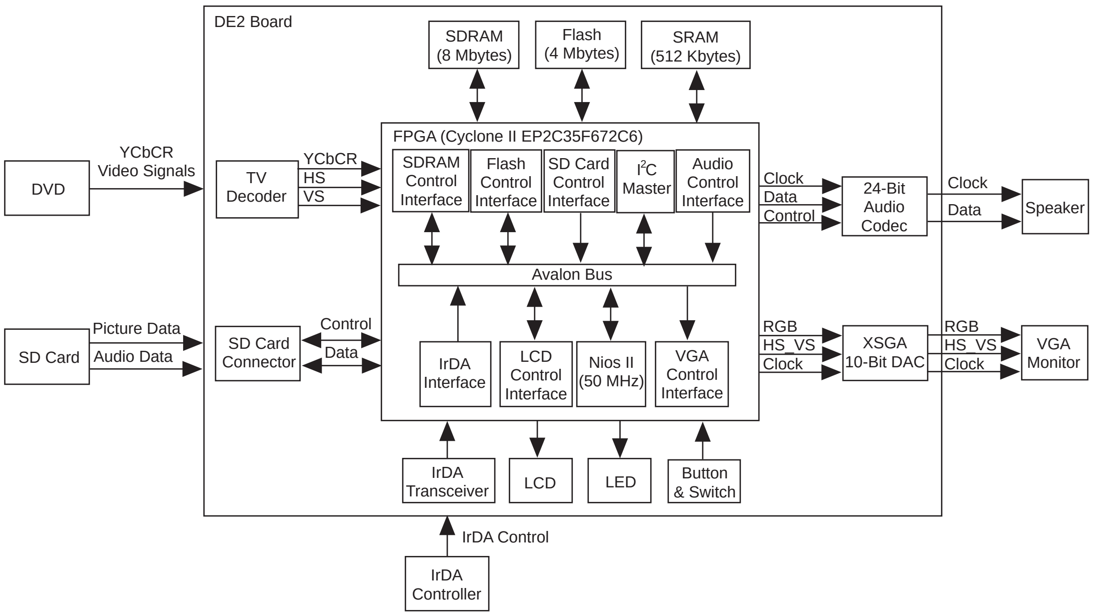
The following additional resources were used:

- *SD card*—1-Gbyte Kingston SD card
- *Infrared remote control*—Epson projector infrared remote control

Design Structure

The system design fully uses the available hardware resources, including the FPGA and the DE2 board. The system includes a DVD, SD card, DE2 board, speaker, and monitor. The DE2 board provides an FPGA, SDRAM, SRAM, flash memory, SD card reader, XSGA 10-bit DAC, infrared interface, and a 24-bit audio CODEC. The system implements a variety of functions, including video processing and display, menu display, an e-photo album, and infrared remote control. Figure 2 shows the system structure.

Figure 2. System Structure



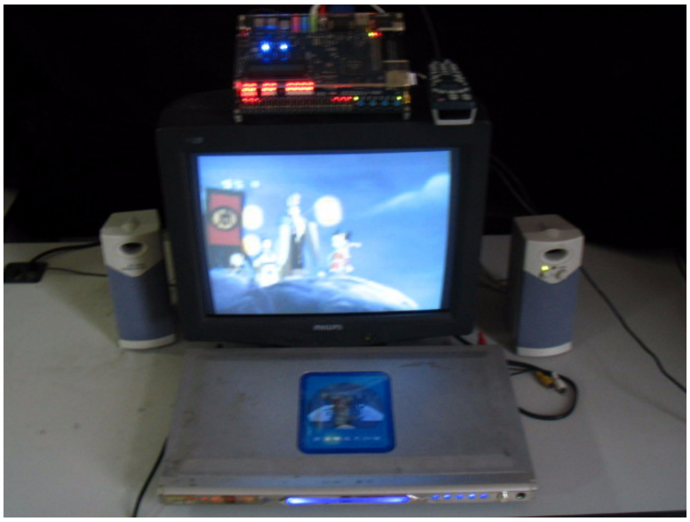
The DVD video source is input through the video in port. It passes through the TV decoder chip (ADV7181) on the DE2 board. The analog video signals input through the video in port are converted to NTSC-format system digital video signals that comply with the ITU-R656 standard. Then, the signals are sent to the FPGA internal video decoding module, which converts the YCbCr color-difference signals into RGB signals. Next, the signals go through the video processing module, which performs16:9 resolution processing, 4:3 resolution processing, or cutting. The system stores the processed video signals in the video buffer module. It reads the data in the buffer module, conducts analog-to-digital conversion with the XSGA 10-bit DAC chip (ADV7123), and outputs the signals through the VGA interface.

The SD card reader obtains picture and audio information and provides it to the system. The Nios II processor embedded in the FPGA reads the picture and music information provided by the SD card and outputs it to the picture buffer module (512 Kbytes of SRAM) and the 24-bit audio CODEC chip (Wolfson WM8731). The picture buffer module stores the picture data, and the display output module controls the display output. The 24-bit audio CODEC chip transforms digital audio signals into analog audio signals, which are output through the audio output interface.

The infrared remote control sends out control signals to the system. The DE2 board's infrared transceiver (Agilent HSDL-3201) receives the remote control's infrared signals, which are decoded by the FPGA's infrared remote control module and sent to the Nios II processor, which generates each module's control signals.

The LCD control module controls the LCD system name. The LED control module turns the LED on and off. The buttons and switchs function like the infrared remote control, generating signals to control the system's operation. SDRAM and flash are as the memory and program memory of the system, respectively. Figure 3 shows the system components.

Figure 3. System Components



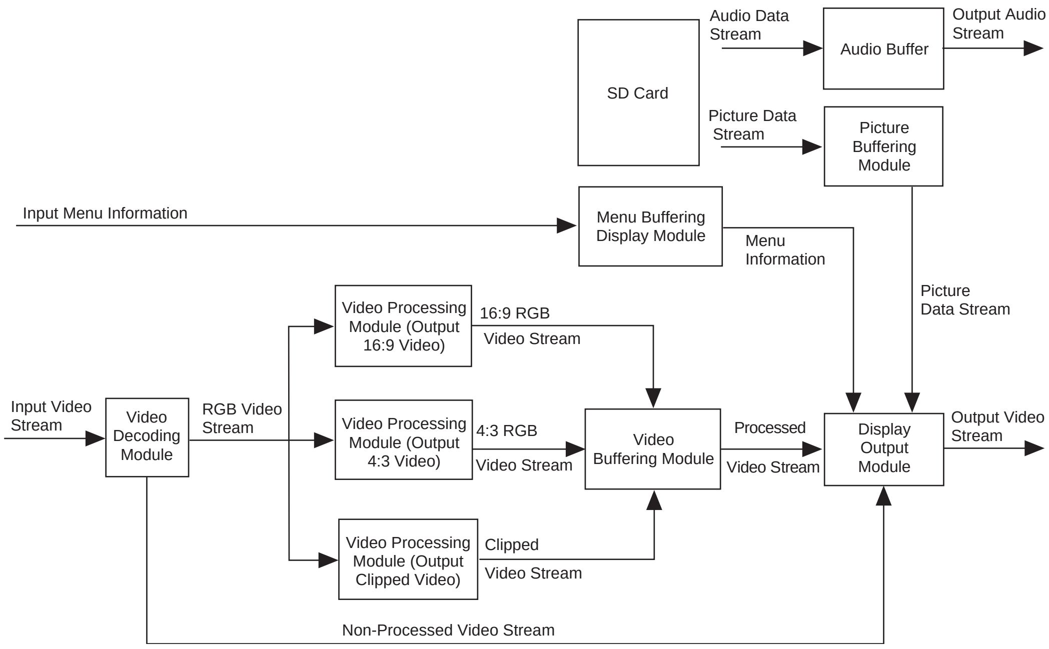
Design Methods

This section describes our design methodology.

Function Design

Our design includes the video decoding module, video processing module, video buffer module, picture buffer module, menu buffer display module, output display module, and process control module. Figure 4 shows the block diagram.

Figure 4. Functional Block Diagram



Hardware Design

This section describes our hardware design.

Video Decoding Module

The video decoding module decodes NTSC video, converting from YCbCr to RGB and from interlacing to a progressive scan of the horizontal synchronizing signals. Meanwhile, it generates the control signals required for the VGA display. It also decodes video and controls the I^2C bus.

The video decoding function outputs RGB three-route color signals and generates control signals such as vertical synchronization, horizontal synchronization, and the VGA clock. It also provides ITU-R656 video decoding and video data exchange. Because the NTSC video signal is for interlaced scanning, the ITU-R656 video decoding section mainly converts the YCbCr video signal and the synchronous signal from interlaced 4:2:2 to progressive 4:4:4. The following formula is for the video data conversion from 4:4:4 YCbCr to RGB:

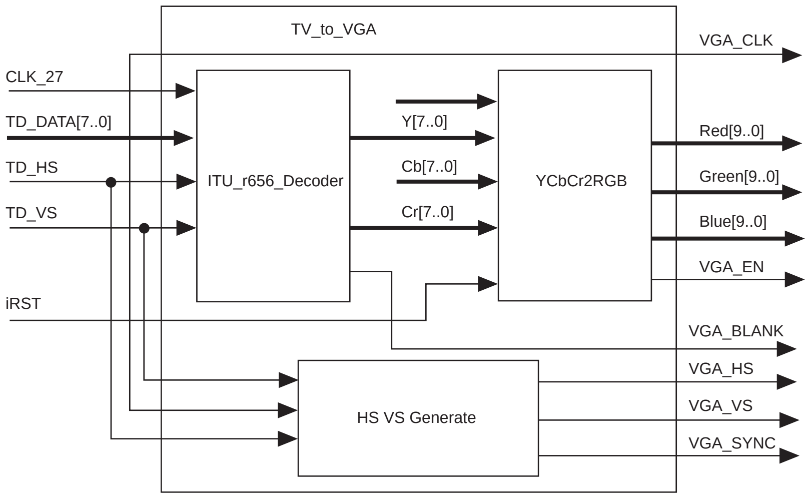
$$R = Y + 1.402(Cr - 128) \qquad (1\text{-a})$$

$$G = Y - 0.34414(Cb - 128) - 0.71212(Cr - 128) \qquad (1\text{-b})$$

$$B = Y + 1.772(Cb - 128) \qquad (1\text{-c})$$

The I^2C bus control section outputs the I^2C bus control signal and the video stream control signal, according to the video data list. Figure 5 shows video decoding block diagram.

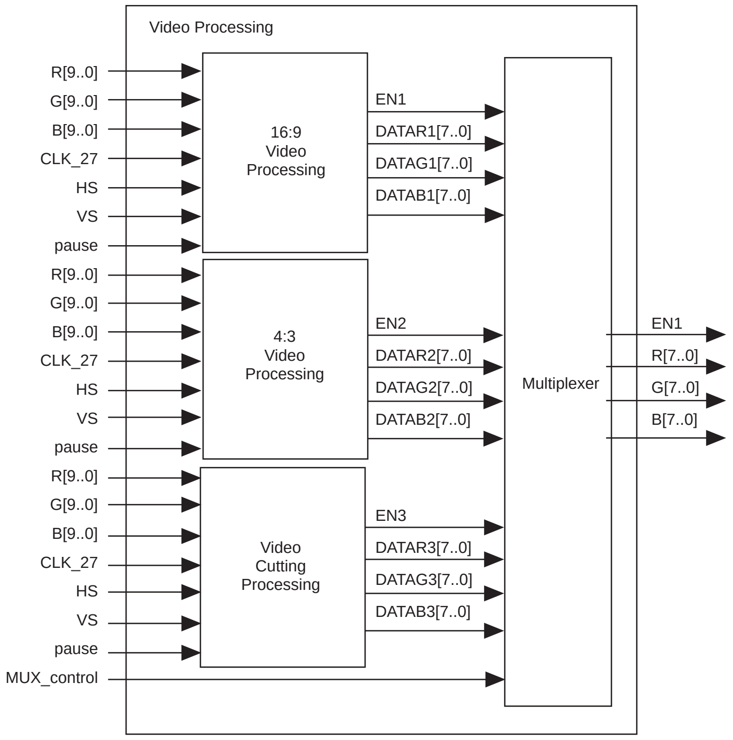
Figure 5. Video Decoding Module Block Diagram



Video Processing Module

Figure 6 shows the video processing block diagram.

Figure 6. Video Processing Module Block Diagram



The video-processing module processes the RGB video data output by the video decoding module. There are three kinds of processing: 16:9 video processing, 4:3 video processing, and video section processing. Therfore, this module includes a 16:9 video processing module, 4:3 video processing module, video clipping processing module, and output control module.

We considered two schemes for the video resolution processing algorithm: linear interpolation and extraction. The linear interpolation algorithm is widely used for video interpolation. The pixels are obtained by weighted calculation of the adjacent domain points. In the following equation we adopt eight adjacent domain sampling weighted calculations to generate the new data.

$$I(x,\ y) = \left(\sum_{i=-1}^{1} \sum_{j=-1}^{1} I(x+i,\ y+j) \times W(x+i,\ y+j) \right) \qquad (2)$$

W(X + i, y + j) are the weighted values of the 8 adjacent domain points.

According to the MATLAB software calculations, this algorithm cannot meet real-time demands. The time complexity is not available for video processing, so we chose not to use this algorithm.

The extraction algorithm is simple and implements resolution compression by dropping part of the original image data. This algorithm does not require mathematical calculations and is very fast. As seen by human eyes, this image is indistinguishable from the one generated using the linear interpolation algorithm. Therefore, to achieve real-time performance and reduce time complexity, we used the extraction algorithm with a hardware module.

The 16:9 video processing module converts the input video to 120 x 60 resolution and 16:9 aspect ratio. During extraction, the module marks a line for every 8 data lines in each frame of the original video image, and then marks a point for every 6 points in the line. It outputs the marked points one by one to the video buffer module, reducing the video resolution and converting the aspect ratio. This module suspends the input terminal to determine whether to output the next video frame. If the terminal is at a high level, it stops outputting data to the buffer module, which freezes the picture. Otherwise, it outputs the data normally.

The 4:3 video processing module converts the input video to 120 x 96 resolution and 4:3 aspect ratio. Then, it marks a line for every 5 data lines in each frame of the original video image and marks a point for every 6 points in the line. It outputs the marked points one by one to the video buffer module, converting to the 4:3 aspect ratio and reducing the video resolution. This module also has a video input suspension terminal that plays and stops the video.

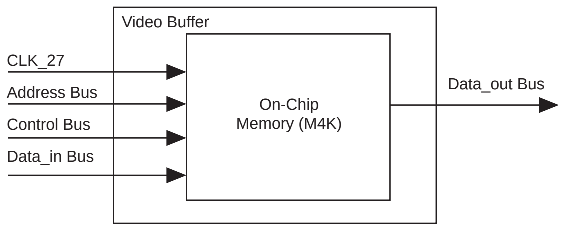
The video clipping processing module captures and displays part of the video, playing at 16:9 or 4:3 aspect ratio. The switch and infrared remote control can determine the captured video's position in the original video. On the basis of the original video, it consecutively marks 96 lines of data in each frame of the image, and then selects 120 consecutive points from the marked lines (column marking). It outputs the selected points one by one to the video buffer module, clipping the video data. The system shifts the clipping position by changing the starting position of the line and point marking. This module also has a video input suspension terminal to control that plays and stops the video.

The output control module switches between the video data output by the three processing modules, which avoids collision.

Video Buffer Module

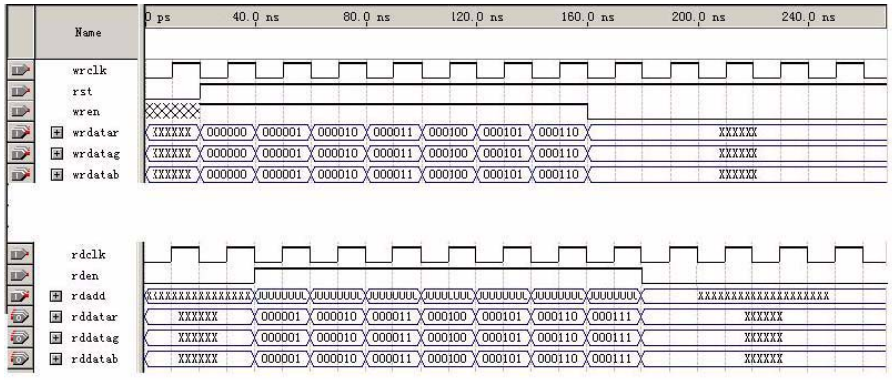
The video buffer storage module implements the video data buffer storage and output control. It stores data as 18 x 11,232 (pixel bits x buffer length) bits in size, and stores a frame of 11,232-pixel video images. This module includes a data buffer storage block and an output address generation block. Figure 7 shows the video buffer block diagram.

Figure 7. Video Buffer Module Block Diagram



After processing video, the video processing module sends a logic high to the video buffer's `WREN` signal. Then, it writes the video data into the data buffer from the starting point; otherwise, it does not write anything. Figure 8 shows the read/write waveform.

Figure 8. Video Buffer Module Read/Write Waveform



After completing video processing, the module waits for 1,000 clock cycles. Then it sends a logic high to the video buffer’s `REN` signal via a counter. The output address generation section uses the buffer length as the module cycle counting and delivers the generated address signals to the data buffer storage. This buffer outputs the stored video data according to the address.

Menu Buffer Display Module

The menu buffer display module stores the menu data output by the Nios II processor and controls the menu storage section output and menu display. It has a menu storage block and a menu display controlling block. The menu storage block’s read/write sequence is the same as that of the video buffer block.

When the switch or infrared remote control sends the menu display signal, the Nios II processor generates menu data and the menu storage block writes the data into the menu storage. Meanwhile, the address generator outputs the address to the menu storage block after it receives the menu display signal. The menu storage block outputs the menu information to be displayed to the menu display control block.

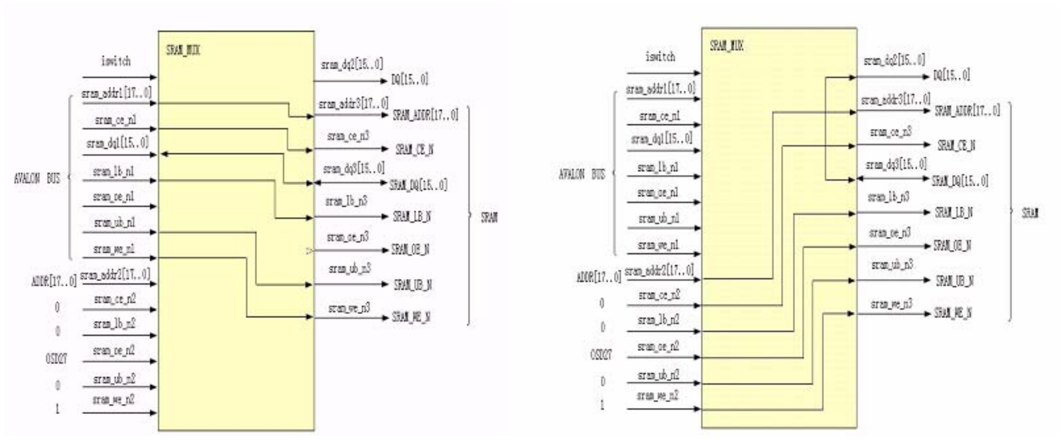
The menu display control block controls the menu display, implementing the video data and menu data’s weighted stacking. After the remote control sends the menu display signal, the menu display control block generates new video data according to the menu data output by the menu storage block. After menu data is sent to the input terminal, if the menu data’s weighted value is much bigger than that of the video data, the system outputs the menu data as video. If the menu data’s weighted value is much smaller than that of the video data, it outputs the original video instead.

Picture Buffer Module

The picture buffer module stores the bitmap data read from the SD card and acts as a VGA control buffer. Because the pictures are stored in the SD card in clusters, the Nios II processor’s picture data is read and written into the picture buffer module according to the picture’s start address and the clusters occupied. When displaying, the address generator generates the output address to control the reading of the data in SRAM.

The picture buffer module has three parts: SRAM, a bus controller, and an address generator. We wrote the bus controller and address generator modules in Verilog HDL. Figure 9 shows bus controller block diagram.

Figure 9. Bus Controller Block Diagram



The SRAM cannot perform dual-port reads/writes, so we designed a bus controller. When displaying the picture, the Nios II processor first sends a control signal to the bus controller, which is connected to the SRAM's data and address lines, to the Avalon® bus. After writing the data, the Nios II processor sends another control signal to the bus controller, thereby connecting the SRAM's data and address lines with the output display module. Meanwhile, the address generator outputs the address signal to SRAM and sets the SRAM to read mode. The system reads the data from SRAM into the output display module.

Output Display Module

The output display module displays the image data (overlapped or not) in each buffer while generating the synchronous signals required by the VGA monitor. It includes an output controller and the overlapped images. Figure 10 shows the output display module block diagram and Figure 11 shows the output display module waveform.

Figure 10. Output Display Module Block Diagram

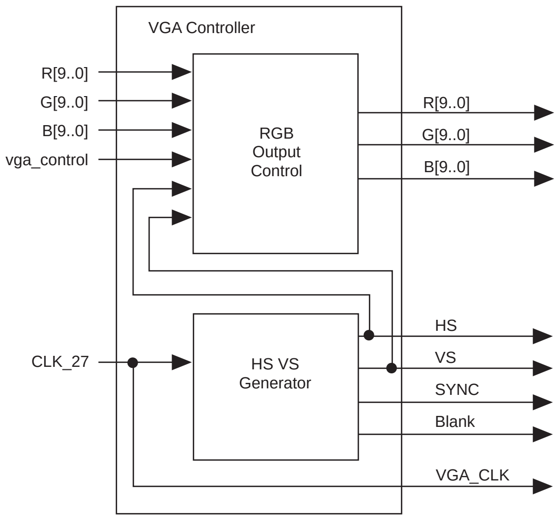
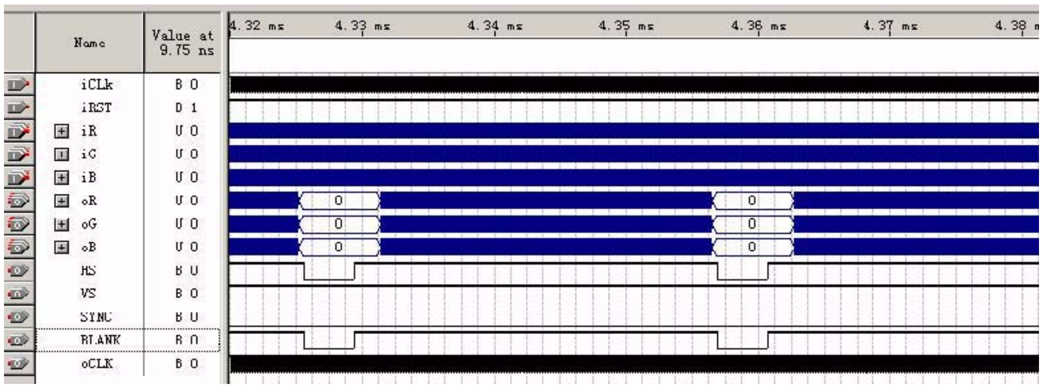


Figure 11. Output Display Module Waveform



The dock clock `iCLK` is 27 MHz and the vertical frequency is 59.94 Hz. As shown in Figure 11, `VS` is the vertical synchronizing signal and the field duration t_{VS} is 16.683 ms. Each field has 525 lines, of which 480 lines are valid display lines and 45 lines are field blanking intervals. Each field of the vertical synchronizing signal `VS` has a pulse, with the low level width t_{WV} of 63 µs (2 lines). The field blanking interval includes the vertical synchronizing time t_{WV}, a field blanking front porch t_{HV} (13 lines), and field blanking back porch t_{VH} (30 lines).

`HS` is the horizontal synchronizing signal, and the line period t_{HS} is 31.78 µs. Each display line has 800 points, of which 640 are valid data periods and 160 are line blanking periods (i.e., non-display fields). Each line of the horizontal synchronizing signal `HS` has a pulse, with the low-level width t_{WH} of 3.81 µs (namely 96 `DCLK`). The line blanking period includes the horizontal synchronizing time t_{WH}, a line blanking front porch t_{HC} (19 `DCLK`), and a line blanking back porch t_{CH} (45 `iCLK`). The composite blanking signal is the logical AND of the line blanking signal and field blanking signal. The composite signal is high during a valid display period and low for non-display periods.

The output controller outputs the RGB video data signal and control signal required by the VGA monitor. According to the input clock, the output controller can generate the control signals, including field synchronization, line synchronization, composite blanking signals, etc., while the output data is the same as the input data. Additionally, it can be controlled to alter the video's position in the monitor.

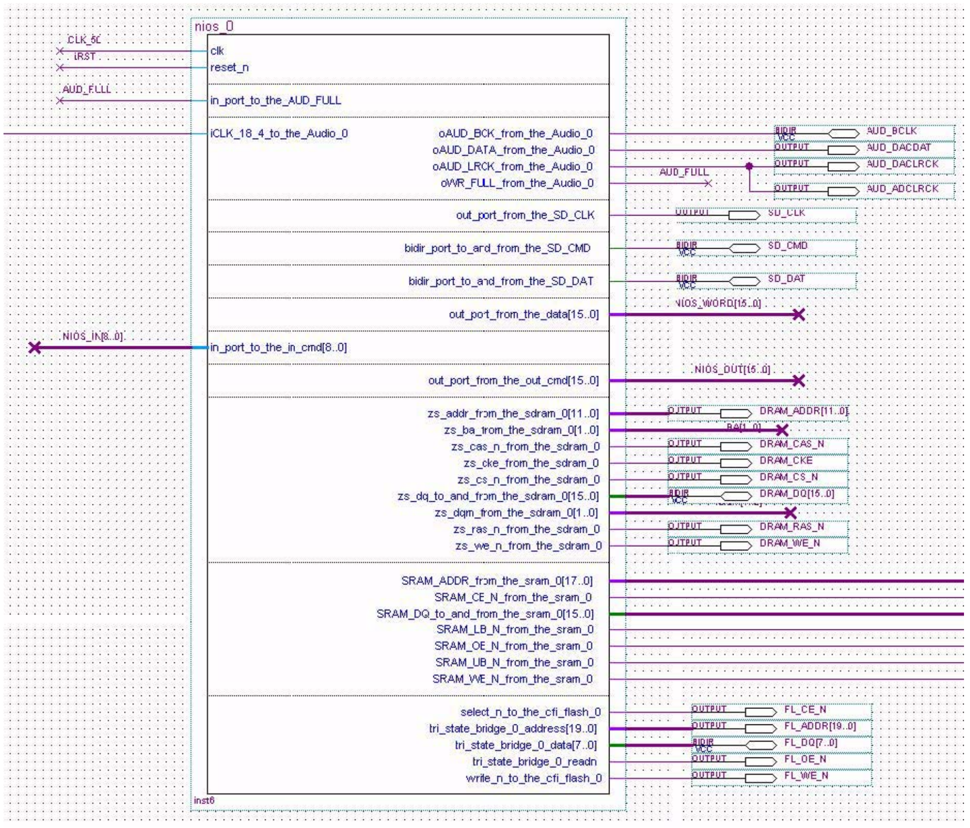
The system can overlap the picture output from the picture buffer module to implement video-in-picture, or overlap the video on the picture to realize picture-in-video. For video-in-picture, the picture is displayed instead of parts of the video image because the picture data is weighted higher than the video data. Video data is shown in locations that do not have input picture data. For picture-in-video, video is shown instead of the picture in some places because the video data is weighted higher than the picture data. Picture data is shown in locations that do not have input video data.

Process Control Module

The process control module generates the modules' control signals, reads the picture and music data in the SD card, outputs the picture data into SRAM and directly outputs the music data, generates menu data, and delivers it to the menu buffer control module. Figure 12 shows the process control module symbol.

The Nios II processor implements the process control module. The module receives button and switch signals as well as signals passing through the infrared remote control module. After analyzing these signals, the module outputs control signals, such as suspension, picture hand-over, motion, menu, etc. Then, it reads the pictures in the SD card into SDRAM and transfers them into SRAM via the Avalon bus. Meanwhile it reads the audio information in the SD card and outputs it through audio interface.

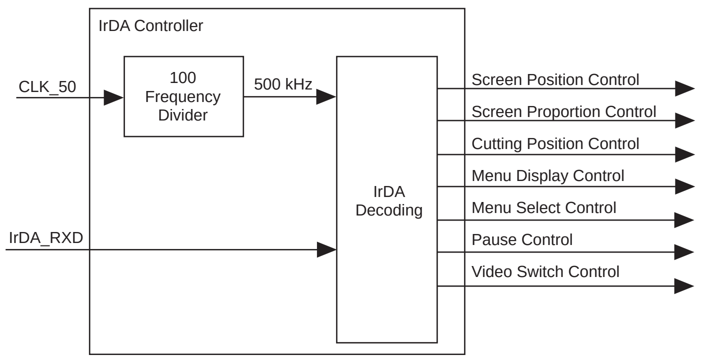
Figure 12. Process Control Module Symbol


nios_0
CLK_50
iRST
clk
reset_n
AUD_FULL
in_port_to_the_AUD_FULL
iCLK_18_4_to_the_Audio_0
oAUD_BCK_from_the_Audio_0
oAUD_DATA_from_the_Audio_0
oAUD_LRCK_from_the_Audio_0
oWR_FULL_from_the_Audio_0
AUD_BCLK
AUD_DACDAT
AUD_DACLRCK
AUD_ADCLRCK
out_port_from_the_SD_CLK
SD_CLK
bidir_port_to_and_from_the_SD_CMD
SD_CMD
bidir_port_to_and_from_the_SD_DAT
SD_DAT
out_port_from_the_data[15..0]
NIOS_WORD[15..0]
NIOS_IN[8..0]
in_port_to_the_in_cmd[8..0]
out_port_from_the_out_cmd[15..0]
NIOS_OUT[15..0]
zs_addr_from_the_sdram_0[11..0]
zs_ba_from_the_sdram_0[1..0]
zs_cas_n_from_the_sdram_0
zs_cke_from_the_sdram_0
zs_cs_n_from_the_sdram_0
zs_dq_to_and_from_the_sdram_0[15..0]
zs_dqm_from_the_sdram_0[1..0]
zs_ras_n_from_the_sdram_0
zs_we_n_from_the_sdram_0
DRAM_ADDR[11..0]
BA[1..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_RAS_N
DRAM_WE_N
SRAM_ADDR_from_the_sram_0[17..0]
SRAM_CE_N_from_the_sram_0
SRAM_DQ_to_and_from_the_sram_0[15..0]
SRAM_LB_N_from_the_sram_0
SRAM_OE_N_from_the_sram_0
SRAM_UB_N_from_the_sram_0
SRAM_WE_N_from_the_sram_0
select_n_to_the_cfi_flash_0
tri_state_bridge_0_address[19..0]
tri_state_bridge_0_data[7..0]
tri_state_bridge_0_readn
write_n_to_the_cfi_flash_0
FL_CE_N
FL_ADDR[19..0]
FL_DQ[7..0]
FL_OE_N
FL_WE_N
inst6

Infrared Remote Control Module

The infrared remote control module receives signals from the infrared remote control and decodes them according to its defined coding rules. Figure 13 shows infrared remote control module block diagram.

Figure 13. Infrared Remote Control Module Block Diagram



According to the coding rules, the module determines whether the signal is 0 or 1 according to the infrared signal's high-level length. It is 0 if the high-level duration is within the sampling period of 220 to 370 samples; it is a 1 if it is within 520 to 670 samples. Other lengths are invalid.

The infrared remote control module translates the instructions from the remote control and sends them to the Nios II processor. This module has two parts:

- Frequency division sampling, which divides the 50-MHz clock into 500-kHz to sample the remote controlled infrared signals.
- Instruction translation, which translates the sampling signals.

Software Design

The system software runs on the Nios II processor. It responds to the user and meanwhile sends control signals to the external hardware modules. The system software uses an inquiry method, cyclically auditing the state of the input ports upon initiating the software. It executes the corresponding operations if pulse signals are sent to the input port.

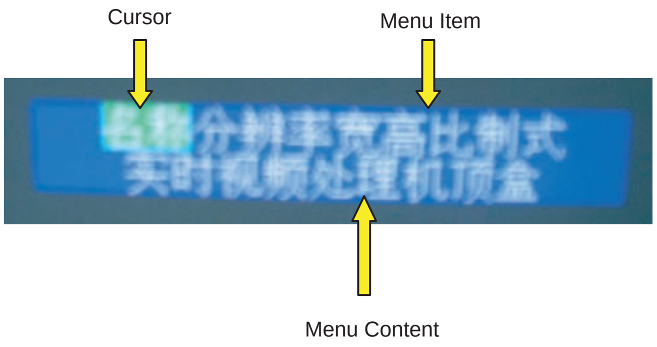
The system menu has three parts: the menu item, cursor, and menu content. The menu item includes the system name, resolution, aspect ratio, and mode, and the cursor indicates the selected item. The menu is generated by the following functions:

- `void draw_menu(int cur_position)`, where `cur_position` is the current position of the cursor.
- `void draw_item(int row, alt_u16 *title, int len)`, where `row` refers to the line of the menu, `title` is the content string of the menu to display, and `len` is the content length.
- `void draw_cursor(int xstart, int w)`, where `xstart` is the starting x coordinate and `w` is the cursor width.

While displaying the menu, the system uses the `draw_menu` function to generate the menu content and update the menu buffer according to the cursor position. The system uses the `draw_item` for drawing the menu items and and `draw_cursor` for drawing the cursor.

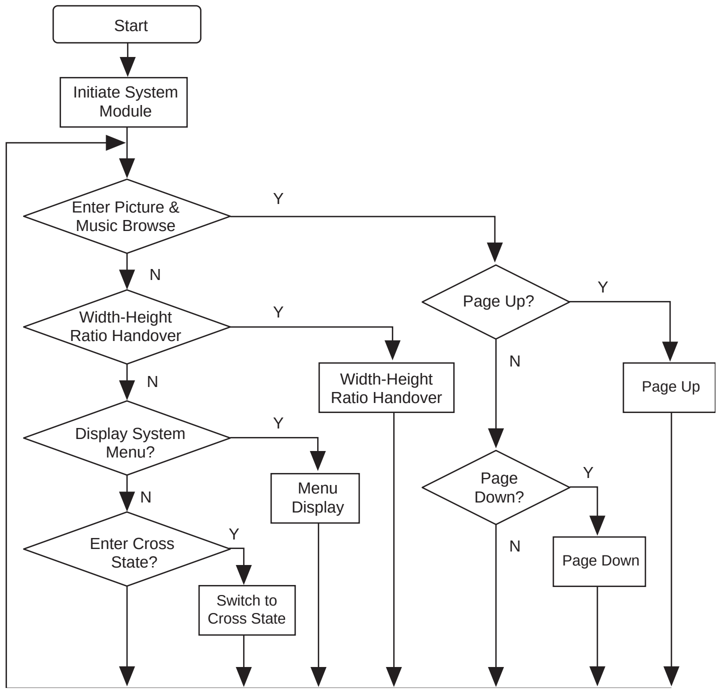
The menu has two lines. The first line displays the menu items, including name, resolution, aspect ratio, and mode. The second line display the menu content according to the cursor position. Figure 14 shows the menu graphics.

Figure 14. Menu Graphics



When displaying the video's aspect ratio and resolution, the Nios II processor automatically checks the current video aspect ratio against the value of the state variable `ratio_state` (0 refers to the 16:9 state, and 1 refers to the 4:3 state) and displays the corresponding information. Figure 15 shows the system software procedure.

Figure 15. System Software Procedure



Design Features

The system has the following features.

- *SOPC concepts*—The system uses SOPC concepts to combine the Nios II processor, modules, and other peripheral control logic. With the custom peripherals we added using SOPC Builder, the system selects processors, buses, memories, and I/O interfaces to build a SOPC system that matches the STB design. This process reduces volume, enhances the reliability, lowers power consumption, and allows the system to be upgraded and scaled.

- *User-defined logic*—We used the user-defined logic in the system design. The user-defined logic consists of an audio analog-to-digital converter (ADC), FIFO buffer, and PLL. The logic reads audio data from the SD card and outputs it to the speaker. The user-defined logic is added into the Nios II processor; therefore, we simply control the audio data reading and output with software.

- *Bus-switching*—In the picture display function, we use the off-chip memory SRAM as the picture buffer as the on-chip memory is stuffed. SRAM is a single-port memory, when the system outputs one picture, the SRAM must be refreshed. We used bus switching for this function. When the picture data is updated, the system connects the SRAM to the Avalon bus; after the update, the SRAM disconnects from the Avalon bus and connects to the external bus. The display output module uses the external bus to read data in the buffer and outputs the picture data.

- *Fully uses development board resources*—The system fully uses the development board resources, such as the FPGA, SDRAM, SRAM, flash, SD card reader, XSGA 10-bit DAC, infrared interface, and 24-bit audio CODEC. The design uses 79% of the on-chip memory (because of the huge volume of information required for video processing) and only about 33% of the logic.

- *Hierarchical display*—The system uses hierarchical design for the menu layer, video layer, and picture layer. The video layer contains the original video layer and processed video layer. The menu layer contains the menu word layer, menu background layer, and menu cursor layer. Every point of the final image is the result of real-time overlapping of corresponding points in those layers according to the image's needs. The hierarchical design lets the system control each layer separately without affecting the display of other layers. It also facilitates data processing of the display layers and system updates.

- *On-screen display (OSD)*—Human-machine interaction requires many notification messages. The state of the video must be shown synchronously with images on the screen; the system displays them using OSD to facilitate user operation.

  Due to limited resources, the system does not add a bitmap character library. Instead, it stores several fixed combinations of the information to be displayed to save storage space. When the system works, the Nios II processor first determines the user's operation, generates the menu information through the Nios II processor, writes the menu information on the menu buffer module, and adds it to the display module.

Conclusion

Using an FPGA design, SOPC concepts, bus switching technology, and hierarchical display, the system implements video processing and display, menu display, an e-photo album, and an infrared remote control. Video processing must be real-time, so the hardware circuit module implements the video decoding, processing, and display. The Nios II processor controls peripherals, generates the menu, and performs human-machine communication. The Nios II processor improves system control flexibility and simplifies the read/write memory operations.

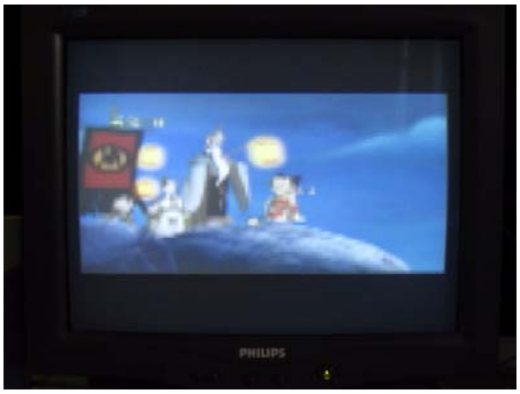
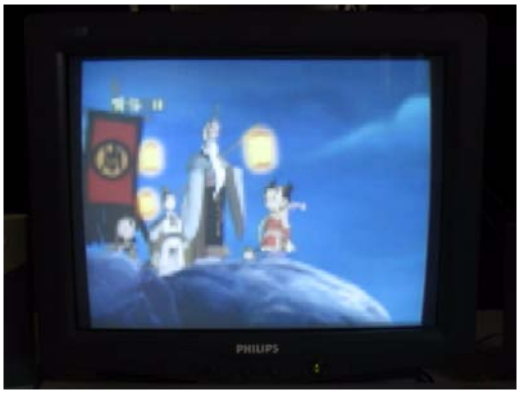
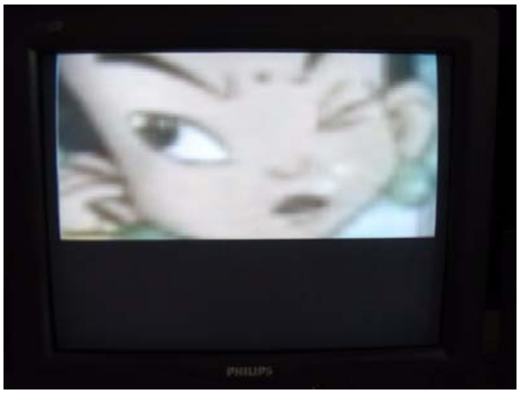
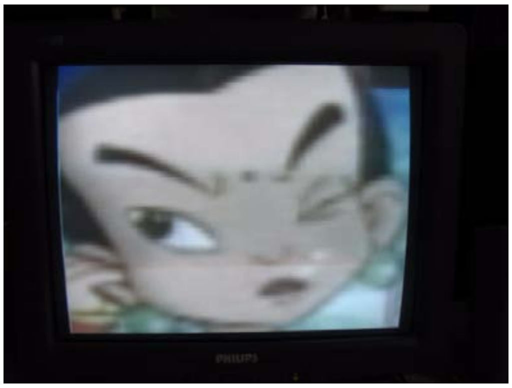
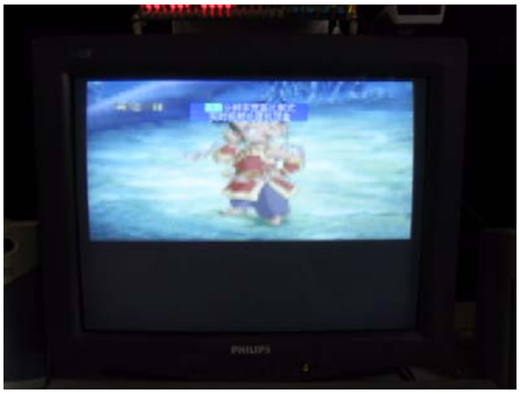
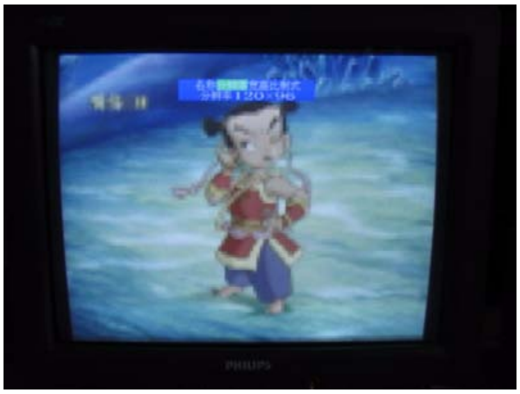
Using the Nios II processor is a new concept in embedded system design. After several months of study, we witnessed the powerful function of the Nios II processor as an embedded processor. When designing systems with SOPC Builder, users define each component according to the system's functional requirement. Development tools provide common IP cores for users to deploy directly. Additionally, users can add custom peripherals to simplify the hardware design, enabling us to focus on function development and algorithm design, thereby improving our design efficiency.

The EP2C35 FPGA chip on the DE2 board that we used in this competition has limited storage resources. Therefore, we only created an FPGA-based STB design. If we used an FPGA with more resources, our system could implement additional functions such as real-time decoding, network communications, video-on-demand, and conditional access of the video code stream of MPEG-2 or MPEG-4.

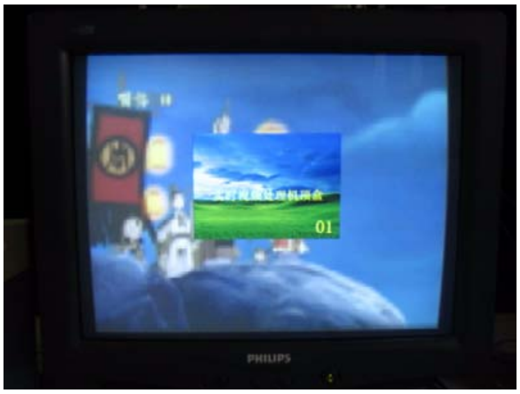
We are grateful to Altera Corporation for giving us the opportunity to participate in the competition.

Appendix 1

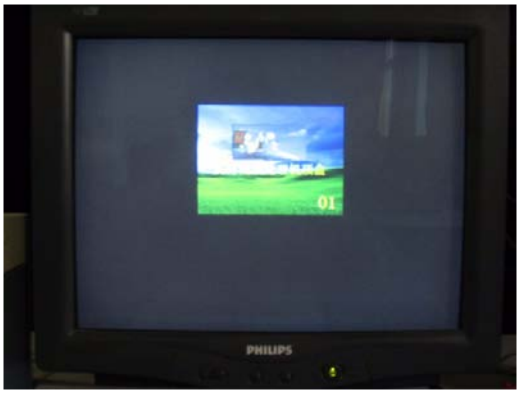
Ths following photos show the display effect of the STB capable of real-time processing.

	16:9	4:3
Display		
Snapshot		
Menu		

Video-in-Picture:



Video-in-Picture:



Appendix 2

The following table shows the STB remote control buttons. Because the input system control signals are pulse signals, the switch should be turned down as soon as it is turned on to simulate a pulse signal while using the switch to carry out corresponding functions.

STB Remote Control Buttons

Name of Remote Control Buttons	Corresponding Buttons of DE2 Board	Meaning
NUM8	KEY3	Video position up
NUM2	KEY2	Video position down
NUM4	KEY1	Video position left
NUM6	KEY0	Video position right
NUM8 (Num = 1)	SW3	Internal video up in the snapshot
NUM8 (Num = 1)	SW2	Internal video down in the snapshot
NUM8 (Num = 1)	SW1	Internal video left in the snapshot
NUM8 (Num = 1)	SW0	Internal video right in the snapshot
ENTER	SW15	Pause
ESC	SW16	Snapshot and display video conversion
Page+ (NUM = 1)	SW10	Processing video and original video conversion
DVI	SW17	16:9 and 4:3 conversion
Video (NUM = 1)	SW9	Picture-in-video and video-in-picture conversion
Freeze (NUM = 1)	SW11	Picture up in video-in-picture
Comp	SW12	Picture down in video-in-picture
Effect (NUM = 1)	SW13	Menu on/off
Effect	SW14	Menu options