Proceeding of the IEEE
International Conference on Information and Automation
Shenzhen, China June 2011

Hardware System Design of SD Card Reader and Image Processor on FPGA

Yansi Yang, Yingyun Yang, Lipi Niu, Huabing Wang and Bo Liu
Information Engineering School
Communication University of China
Beijing, China
yys19892007@sina.com

***Abstract* - We designed a useful digital signal generating system which transforms various file data stored in SD card into SDI output signal based on the FPGA hardware platform. This paper presents the hardware design and implementation of the system, which includes two steps. First step is the design of the NIOS II system, which includes SRAM controller and SD card controller IP core design. Second step is the generation of the whole functional SOPC system which using Quartus II development tool. NIOS II system is integrated with the scrambling encoders in this step. And then the hardware system is implemented.**

Key Words – FPGA, SDI signal, SD card, SOPC, NIOS II CPU.

I. Introduction

SD card is a new generation storage device based on semiconductor flash memory, with high memory capacity, fast data transfer rate, great moving flexibility and good security. Because of these advantages, SD card is widely applied in portable devices, such as digital camera. At present, SONY and Panasonic of Japan have released video camcorder which use SD card to store video. SD card has gradually been used in television storage.

This paper describes FPGA hardware platform which use NIOS II CPU to get files stored in SD Card, to process the image files, and to output SDI signal. The FPGA chip is Cyclone Ⅲ EP3C25Q240 which complete with high performance. The hardware platform is shown in Fig.1.

BMP files will be written into the SD Card at first. Then image files in SD Card will be processed by embedded system based on NIOS II CPU and written into SRAM. The image data in SRAM will be transmitted to the scrambling encoder, converted from parallel signal to serial signal, and finally, output as SDI signal.

II. Framework of Hardware System

To achieve the function, it is need to configure NIOS II CPU, SRAM controller, SD card controller, DMA, timer and Scrambling Encoder in FPGA. Fig. 2 is the framework of this hardware system.

The most important part is NIOS II system, the core of which is NIOS II CPU. This part can be generated as NIOS II system module by SOPC Builder. Some peripheral equipments of NIOS II system, such as PIO, timer, JTAG and DMA, are selected from peripheral equipment library of SOPC Builder. While SRAM controller and SD Card controller are user-defined IP cores which do not exist in the peripheral equipment library of SOPC Builder. The Avalon bus can be automatically generated by SOPC Builder system. most of the modules can be automatically connected to the Avalon bus, but several modules which have complex master-slave relationship should be connected by user [1].

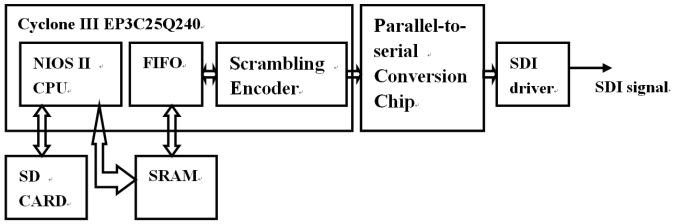


Fig.1 hardware platform

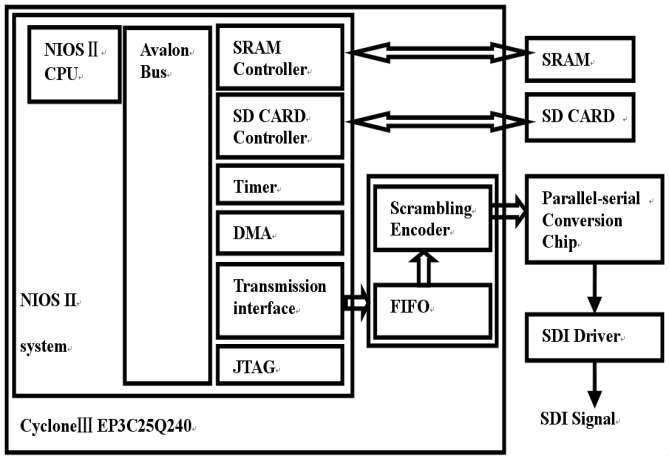


Fig. 2 Framework of Hardware System

After Files from SD Card are transformed to appropriate data stream by NIOS II system, the data stream will be transmitted into Scrambling Encoder through the video transmission interface. Scrambling Encoder can be chip or FPGA module. The second option is applied in the design. The NIOS II system module will be combined with the Scrambling Encoder to consist the FPGA top system which is in the chip Cyclone III EP3C25Q240.

The FPGA chip will not only achieve data transmission between NIOS II system and SRAM chip or between NIOS II system and SD CARD SOCKET, but also transmit parallel signal from Scrambling Encoder to parallel-serial convertor to

generate the SDI signal. Parallel-serial convertor can be chip or FPGA module. In this design, we choose parallel-serial conversion chip.

So the hardware system consists of NIOS II system, FPGA top system and Peripheral hardware, and NIOS II system which is the most important and complicated part. NIOS II system will be created by the SOPC Builder development tool, and the FPGA top system will be created by the Quartus II development tool. Here we will analysis the hardware system.

III. The Creation of NIOS II System Module

The NIOS II system will be created in SOPC Builder, and it should set CPU, JTAG, On-chip memory, SRAM Controller, DMA, Timer, SD CARD Controller and the video transmission interface.

The creation of NIOS II system module in SOPC Builder is depicted in Fig.3.

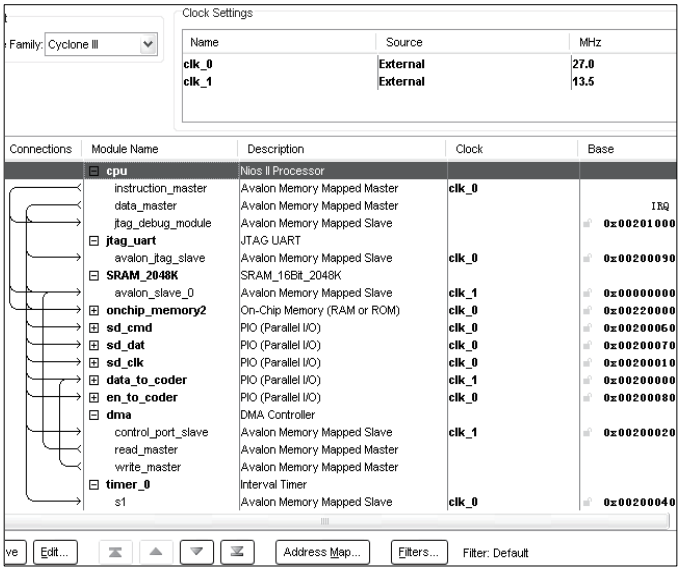


Fig. 3 System Configuration Page of SOPC Builder

The form of NIOS II system peripheral device is IP core. There are two kinds of peripheral devices in NIOS II system, which are standard peripheral device and customize peripheral device. Standard peripheral device come from the standard peripheral device library provided by Altera Company, such as Timer, SPI, PIO, SRAM Controller and other memory controllers. SOPC Builder contains these peripheral devices. Customize peripheral device is the peripheral device created by designer user or third party and will be integrated into NIOS II system, which described by HDL language[1].

Some peripheral devices, such as CPU, JTAG, On-chip memory, DMA, Timer, will be achieved by standard peripheral device IP cores. We can choose these devices in SOPC Builder.

SRAM Controller, which is an customize peripheral device, will created by user.

The other parts of NIOS II system module, such as SD CARD Controller and video transmission interface, will be achieved by PIO. Especially the SD CARD Controller, made up of three PIOs, the function of which will be designed by software.

Avalon bus-regulation provides interconnection model of data transmission between peripheral port and bus module [1]. Some simple connection, such as between CPU and SRAM Controller, can be automatic generated by system. While the complex connection, such as connection of DMA, SRAM Controller and the video transmission interface, should be connected by user.

In this system configuration page, users should add NIOS II CPU and some peripheral devices, set clock frequency and FPGA chip type, and set master and slave ports for some complex peripheral interfaces. In this design, there are two clocks: 27MHz and 13.5MHz. External crystal frequency and CPU frequency are the same, 27MHZ. Because the data width of SRAM is 16bits, the device frequency related to the SRAM will be set as 13.5MHz. DMA and video transmission interface which related to the video transmission will use this frequency, too.

A. SD Card Data Transmission Interface (SD CARD Controller)

Read-write mode of SD card includes SD mode and SPI mode[3]. It is different between these two kinds of read-write mode in amount and setting of transmission interface. Consider for the transmission rate, we use SD mode.

The bus topology of SD mode is shown in Fig.4.

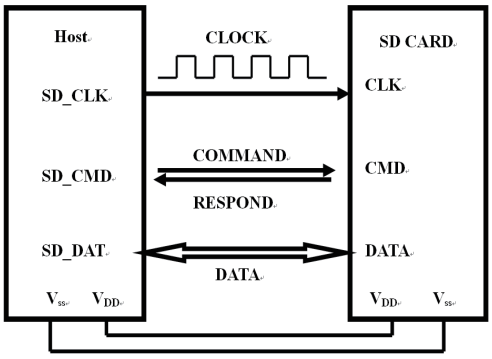


Fig. 4 bus topology of SD mode

In SD mode, based on SD Card Socket pin assignment, SD Card Data Transmission Interface is divided into three parts, which are SD_CLK, SD_CMD and SD_DAT. They will be achieved by PIO core existing in SOPC Builder component library. PIO interface can just achieve the data transmission. SD card read-write transmission control will be the software mission.

First is the Interface SD_CLK. On this interface, the host, NIOS II system, will transfer clock signal to SD card. Its width is 1bit.

Second is SD_CMD, which send command from host to SD card and receive respond from SD card. SD_CMD should be tri-state interface, its width is 1bit.

The interface SD_DAT takes control of data transmission between NIOS II system and SD card, which should be tri-state interface. In SD mode, the maximum parallel data width can be 4bits, so its width is set as 4bits.

B. Video Transmission Interface

Video transmission interface include data interface "data_to_coder" and encoder enable interface "en_to_coder". The mission of video transmission interface is to enable scrambling encoder and is used to transfer digital video component signal form SRAM to scrambling encoder. To match with SRAM data width, data width of data_to_coder should be set as 16bits, and frequency should be set as 13.5MHz.

Encoder enable interface "en_to_coder" is used to transmit enable signal to scrambling encoder. Scrambling encoder will be enabled shortly before data transmission, that can ensure recourses are not wasted. The frequency will match with CPU, select 27MHz.

C. SRAM Controller (16bits, 2MBytes)

For reasonable planning and using the resource, there are two memory devices, On-chip memory and SRAM, in the NIOS II system. The former is achieved by FPGA resource. It has low memory capacity and stores program and stack. The latter is an independent chip with high memory capacity, is applied to store some processed image and video data.

The SRAM chip on PCB is IS61WV102416BLL-10_1. The data width is 16bits, the address line width is 20bits, and the capacity is 2Mbytes. Corresponding SRAM controller do not exist in component library of SOPC Builder. User should design IP core to control reading and writing of SRAM. SRAM controller core is shown in figure 5.

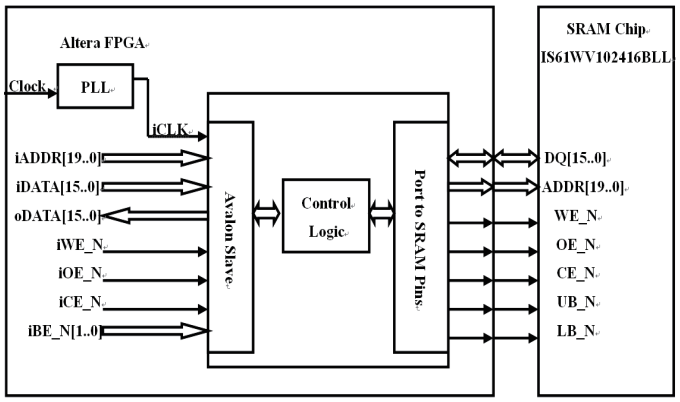


Fig. 5 SRAM Controller Core

At first, we see the Avalon slave port. SRAM can not initiatively transmit data to Avalon bus. It just response data transmission requisition from other master device, so the SRAM controller just contains slave port. The signals of slave port are clock "CLK", 20bits address line "iADDR", 16bits data line "iDATA", 16bits data line "oDATA", chip select line "iCE_N", read enable line "iOE_N", write enable line "iWE_N" and 2bit byte enable signal "iBE_N".

Tri-state does not exist in system, so tri-state signal must not exist in the slave port of SRAM. Because of that, the data line can not be set as a tri-state signal, which should separate the input function from the output function of the data line.

When Avalon slave port transport data to the memory device which data width is more then 8bits, the signal byte enable signal iBE_N will assign the particular segment of bytes. The data width is 16bits, twice of 8bits, so the width of byte enable signal is 2bits.

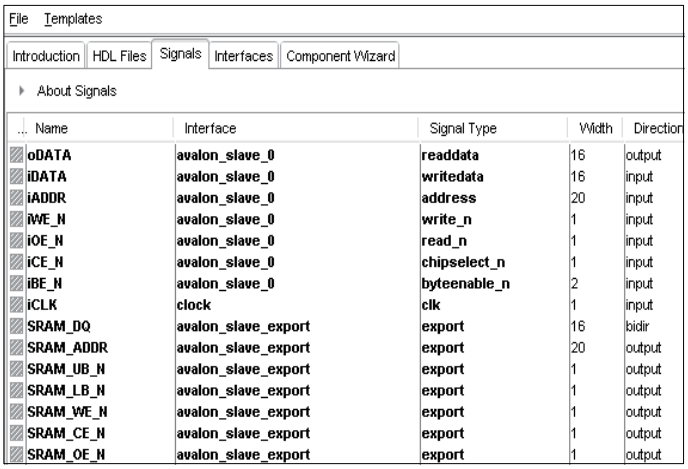
The port connected to SRAM pins should correspond with the pins of SRAM chip IS61WV102416BLL. Pin description of IS61WV102416BLL is shown in table 1.

TABLE I
SRAM PIN DESCRIPTIONS [2]

Pins	Description
A0-A19	Address Inputs
I/O0-I/O15	Data Inputs/Outputs
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
$\overline{LB}$	Lower-byte Control(I/O0-I/O7)
$\overline{UB}$	Upper-byte Control(I/O8-I/O15)
NC	No Connection
V_{DD}	Power
Paper title	Ground

Because SRAM reading and writing control is simple, slave port signal can just correspond with the pins of SRAM chip one by one.

When we create the SRAM controller IP core in SOPC Builder, signals should be set into different type, as Fig. 6.



Name	Interface	Signal Type	Width	Direction
oDATA	avalon_slave_0	readdata	16	output
iDATA	avalon_slave_0	writedata	16	input
iADDR	avalon_slave_0	address	20	input
iWE_N	avalon_slave_0	write_n	1	input
iOE_N	avalon_slave_0	read_n	1	input
iCE_N	avalon_slave_0	chipselect_n	1	input
iBE_N	avalon_slave_0	byteenable_n	2	input
iCLK	clock	clk	1	input
SRAM_DQ	avalon_slave_export	export	16	bidir
SRAM_ADDR	avalon_slave_export	export	20	output
SRAM_UB_N	avalon_slave_export	export	1	output
SRAM_LB_N	avalon_slave_export	export	1	output
SRAM_WE_N	avalon_slave_export	export	1	output
SRAM_CE_N	avalon_slave_export	export	1	output
SRAM_OE_N	avalon_slave_export	export	1	output

Fig. 6 Signals and Interface Configurations SRAM Controller

There are three kinds of interface type in this IP core. The slave port avalon_slave_0 connecting with Avalon bus is Avalon Memory Mapped Slave type, which control by the signal clock. The signal clock is Clock Input type. The port avalon_slave_export connected with SRAM pins is Conduit type. The Conduit type port can become the output port of NIOS II system module.

D. DMA Controller and Timer

DMA, Direct Memory Access, based on Avalon bus, can be applied to transmit mass of data between memory and memory, memory and peripheral device, or peripheral device

and peripheral device. Avalon master devices, such as NIOS II CPU, assign data transmission to DMA. When DMA transmits data, Avalon master devices can accomplish other task [1].

DMA controller is used for direct memory access from source address space to destination address space. The source or destination address space can be an Avalon slave device or a memory. DMA controller include two master ports and a slave port that the former is master read port and master write port, while the latter is used to receive the control signal from NIOS II CPU [1]. DMA controller can be seen in Fig. 7.

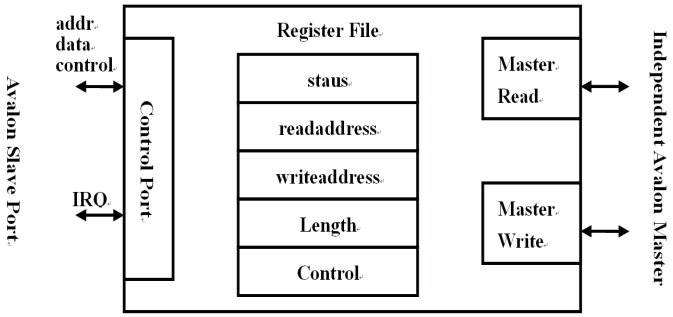


Fig. 7 DMA Controller Core[1]

DMA in this design achieves data transmission between SRAM and video transmission interface that the former is source address space and the latter is destination address space. This design will achieve the output of digital video component signal under the scanning format 625/50i, which output a frame of image data including 1080000 Bytes in the display period of a frame, namely 40ms. Data transfer rate is demanded 270Mbps(10bits per word). Without DMA, the hardware system can not get such high transmission rate just depending on CPU.

In the Advanced Options of DMA, we can choose the kind of data width. To match the data width of SRAM, it can be selected the option "Halfword(16bits)" to save the FPGA logic recourse used by DMA.

The master ports of DMA should be connected by user. In this design, DMA transmits the data from SRAM to video transmission interface, so the master read port will be connect to SRAM, and the master write port will be connect to video transmission interface.

Just starting the DMA, data will be continually transmitted from SRAM to video transmission interface with the bit rate 270Mbps. For normally display, several images should be output continuously. But the DMA of NIOS II can not circularly transmit the data in same address space. We add a Timer in system to interrupt after counting 40ms, namely the display period of one frame, to start DMA again.

IV. FPGA Top System

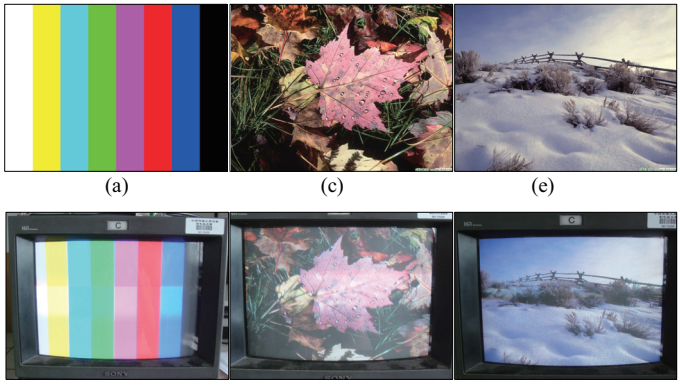
FPGA top system includes NIOS II system module, 16bits-to-8bits converter module and Scrambling Encoder module. Integration will be accomplished in Quartus II development tool.

Data from SRAM is 16bits parallel data per clock cycle under the frequency 13.5MHz, will be changed into 8bits parallel data per clock cycle under the frequency 27MHz by 16bits-to-8bits converter module.

The function of scrambling encoder is scrambling coding and NRZ – NRZI transform coding. Because it needs to recover clock signal when decoding at sink, and serial interface cannot use a single clock line to transmit clock signal as parallel interface, we can just use the step of signal to recover clock, we need scrambling coding and NRZ – NRZI transform coding. The purpose of scrambling coding is to decrease long continuous "0" and "1", there are just short continuous "0" and "1" in data stream, to increase level steps and enrich clock information. The aim of NRZ – NRZI transform coding is that NRZI code is different from NRZ code. It uses the variation of level to represent "0" and "1" but not the polarity of level. We can use this characteristic to easily get clock information and decode at sink just making use of transform response of level polarity but not polar response of data stream [4].

V. Experimental Results

Combining the Hardware platform and software design, it can process the BMP file stored in SD CARD, output real-time SDI signal and display the image on TV monitor. Fig.8 is the experimental results.



(a) colorbar.BMP in SD CARD; (b) color bar on TV monitor;
(c) leaf.BMP in SD CARD; (d) "leaf" on TV monitor;
(e) snow.BMP in SD CARD; (f) "snow" on TV monitor.

Fig. 8 Experimental Results.

Because of the limit of memory capacity, at present, it just can process and display one image but not video. We will improve the hardware, increase SRAM chips, even SDRAM chip to achieve the goal.

References

[1] Gang Wang, Lian Zhang, "SOPC Embedded System Design and Typical Examples Based on FPGA," Beijing：Electronic Industry Press.
[2] ISSI, "IS61WV102416ALL," October 2006.
[3] SD Group, "SD memory card Specifications, part 1: physical layer Specifications," version 1.0, March 2000.
[4] Lipi Niu, Yingyun Yang, Yansi Yang, Xiuhua Jiang. "Conversion from CVBS to SDI based On FPGA," Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, Chengdu, China: IEEE Press, 2010, Volume 8, Page 345.