Embedded Image Capture

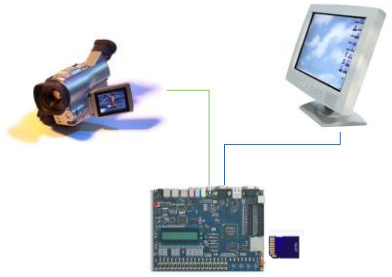
CSEE 4840 Design Document – March 2010

Albert Jimenez   Alexander Glass   Nektarios Tsoutsos
School of Engineering and Applied Science
Columbia University
{alj2110, amg2229, nt2283}@columbia.edu

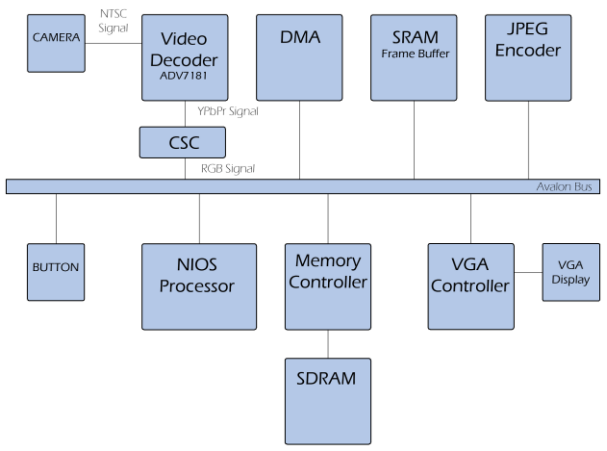
Abstract

This document describes our preliminary design implementation based on put research so far. Our system will use a SRAM frame buffer and DMA controller to save a single frame before compressing it using a hardware JPEG core. At the same time the video input at resolution 320x240 is routed to the VGA output. The NIOS processor controls all peripherals and saves the file to the SD card.

1 Introduction



For this design we plan to use the NIOS process along with several Avalon peripherals. The system will perform 2 basic operations; the first is to display the video input to the VGA output, and the second is to compress a single frame and save it to a file in the SD card. The first operation requires to convert the analog input to digital and change the color space. Then the video signal is sent to the VGA controller to be displayed on the screen. The video signal is also saved frame by frame in a buffer, and upon user request, this buffer is fed to a JPEG module for compression. Finally, the compressed data is saved to the SD card. The system operates at 100MHz, and uses a DMA controller and an SDRAM controller.



2 Hardware description

2.1 Video Decoder

The video input comes from the RCA composite port on the board. This YUV signal is fed to the Video decoder so that it is converted from analog to digital format. The Video decoder has an I^{2}C for setting up the necessary parameters. The output of this module is in the YCbCr color space and the desired resolution will be 320x240 pixels. This module operates at 27MHz and uses the on board clock. The output of this module is send to the color space converter.

2.2 Frame buffer

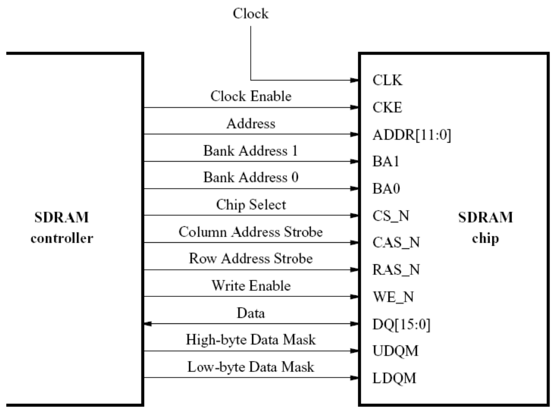
In order to save the color space converter output, we need a memory. We chose the SRAM as our frame buffer because it is faster and has a less complicated interface than the SDRAM. The SRAM module on the Altera Board is 512KB, which limits the size of our frame. We plan to use frames of 320x240 pixels with 16 bits total depth, which very well fits in this buffer. Running at 50MHz, the delay to access a word in this module will be 20ns. The maximum delay for reading an entire frame will be about 1.5 ms. The frame buffer data will later be read by the JPEG module to produce the compressed frame.

2.3 DMA

Our motivation to use Direct Memory Access stems from the need to decouple the SRAM in the frame buffer from the NIOS processor. The controller will allow the video decoder to feed pixel lines into the SRAM seamlessly, so that we can access the result when needed. In principle, the DMA controller will define the write address and control signals in the SRAM, depending on the video output.

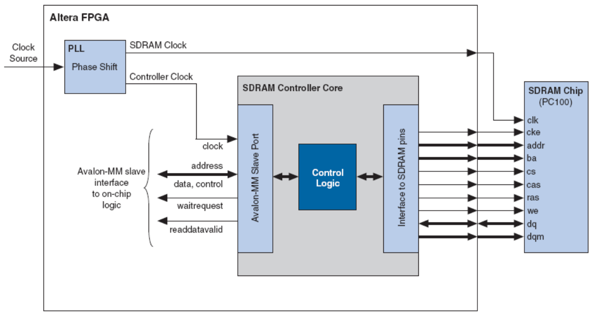
2.4 SDRAM

The onboard SDRAM chip provides us with 8 MB of memory to store captured images. We will also be using the chip to store the jpeg converter program since the SRAM will be used primarily for the video buffer. The controller will be generated using the SOPC builder, which conveniently provides the signals necessary to communicate with the SDRAM chip. We will use the NIOS processor clock as the chip's clock signal, which will inevitably introduce a clock skew. For proper operation it is necessary that the chip's clock signal leads the processor's signal by 3 ns. We will accomplish this by using a phase locked loop (PLL) to supply the correct clock signal.



2.5 Memory controller

The SDRAM needs a special controller to handle refreshing and all read write operations. This memory controller is a megafunction and will be generated by the SOPC builder. The NIOS processor communicates with this controller via the Avalon fabric, so that it can access the stored program.



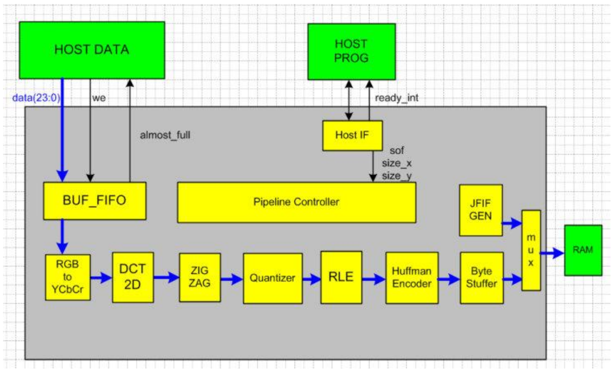
2.6 VGA controller

This module controls the VGA display. The input for this module comes directly from the color space converter in RGB format. The input resolution will be 320x240, but since the desired output is 640x480, the image will be centered with unused pixels set to black.

2.7 JPEG module

The JPEG module performs the compression of raw RGB data. This data comes from the frame buffer in the SRAM. The module performs all the necessary steps of JPEG encoding on 8x8 pixels blocks, that include:

- Color space conversion
- 2D Discrete cosine transform
- ZigZag transformation
- Quantizer
- Run length encoding
- Huffman encoding
- File headers generation



The output of this module is saved to memory. Inside, this core has a complex Finite State Machine, that keeps track of the current hardware operation. The NIOS processor can orchestrate when is module is invoked, so that the output will be saved to a file in the SD card. This module is based on project *mkjpeg* from opencores.org.

2.8 SD card

The SD card module operates using an SPI interface. The processor will issue commands and send data to be written on the card. In particular, the system will save *.jpg* screenshots of the video input, upon user request. Then, the SD card contents can be read from any other card reader.

2.9 PLL

For this project we plan to use two PLL modules. The first module will be used to insert delay to the SDRAM clock, compared to the NIOS clock. The second one will be used to double the frequency of the system. Instead of running the system at 50MHz, it is possible to run it at 100MHz using a clock that comes from a PLL. We plan to use this feature to improve the overall system performance.

2.10 Color space converter

This module is used to change the color space from YCbCr to RGB, so that is can be used by the VGA controller. This module is a megafunction that is included in the SOPC builder.

2.11 The NIOS II

The central module of our system will be the NIOS II processor. This IP core comes from the SOPC builder and will be configured to have a hardware multiplier and divider, as well as a floating point unit with division support. The NIOS will use the SDRAM as its main memory, via a special memory controller. The NIOS will also use the Avalon bus to control all the other peripherals. The operating frequency will be 100MHz, using the PLL clock multiplier.

The processor will be responsible for orchestrating the JPEG compression, for saving files to the SD card (using the SPI interface) and for configuring the Video decoder using the I^2C interface. For debugging, the NIOS incorporates a JTAG interface.

3 Software description

3.1 The NIOS software

The NIOS software will be responsible for coordinating all operations on the board. The program will communicate with the Video decoder using the I^2C protocol. It will also communicate with the card reader using the SPI protocol. The software will be responsible calculating a CRC wherever this is needed. When the user presses a push button, the program will direct the contents of the frame buffer into the JPEG module and grab the output. This output shall be saved to the SD card as a separate file. The program communicates with the VGA controller and the other peripherals using the Avalon bus.

3.2 Writing FAT files

In order to write to the SD card, we will have to use a simple file system. We plan to use the FAT16 file system that is very simple to implement on the NIOS processor. The first 512 bytes are the Parameter Block and contain the volume information, the directory information and the FAT table. Each file consists of several clusters described in that table. The file name we will use are in the standard DOS format and the directory structure will be flat.

4 Milestones

- Milestone 1 (March 29): Getting the input of the camera to the VGA display
- Milestone 2 (April 12): Compressing a single frame using JPEG
- Milestone 3 (April 28): Saving the compressed image to a file

5 References

http://www.cs.columbia.edu/~sedwards/classes/2009/4840/designs/RVD.pdf

http://www.cs.columbia.edu/~sedwards/classes/2009/4840/designs/RJ.pdf

http://www.cs.columbia.edu/~sedwards/classes/2007/4840/designs/Imagic.pdf

www.cs.columbia.edu/~sedwards/classes/2010/4840/tut_DE2_sdram_vhdl.pdf

http://www1.cs.columbia.edu/~sedwards/classes/2010/4840/n2cpu_nii5v3.pdf

http://www.opencores.org/project,mkjpeg

http://white.stanford.edu/~brian/psy221/reader/Wallace.JPEG.pdf

http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=90209

http://freedos-32.sourceforge.net/