ELEX 7660: Digital System Design
Composite Video Encoder with SPI Interface

Matthew Knight, Cole Harkness

2017–04–17

Contents

Introduction

In the simplest of terms, our project is making video driver for composite video. But what differentiates us from other video drivers is that we plan on implementing a majority hardware solution. When this project was first brought up, we knew whatever project we choice had to meet our own personal requirements. It had to be challenging, useful, and have a hardware and software component. Although this already exists in the market they can be over priced. We plan on trying to make the same product for cheaper. Not only would solution be cheaper, it could potentially run faster because a hardware solution can run faster than a software solution could.

Background

The Composite Video Signal

Originally video was broadcast in black and white, only requiring one channel. In terms of driving the television itself, one needs only change DC voltage in the active video region of the signal to change luminosity levels.

Once colour televisions became a reality, video needed to be broadcast in colour as well. In order to also be backwards compatible with black and white televisions still in common use, colour was added to the video signal by adding an amplitude and phase modulated sinusoid. This allowed for only requiring one additional radio band for broadcast, keeping the now "composite" signal on a single wire, and black and white televisions would discard the colour components by passing the signal through a low-pass filter.

Input Output Machine

The simplest way to view this project is as an input output machine. The input being any device that could drive a 9-bit input and a clock. This input was basically, an image broken down into pixels and colours. Our machine would store the input signal and covert it to a composite video output. The composite video output is complicated and requires precise timing, and precise repetition at a high frequency. The purpose of this was to be able to take any device as an input we could convert it to a component video output.

Objectives

Main Objective

The main objective of this project is to generate a black and white composite video signal that will drive a Cathode Ray Tube (CRT) television. To accomplish this, we will develop a video card with a serial interface that can generate a composite video signal. We will be able to achieve a resolution of 240 x 320 pixels on the screen.

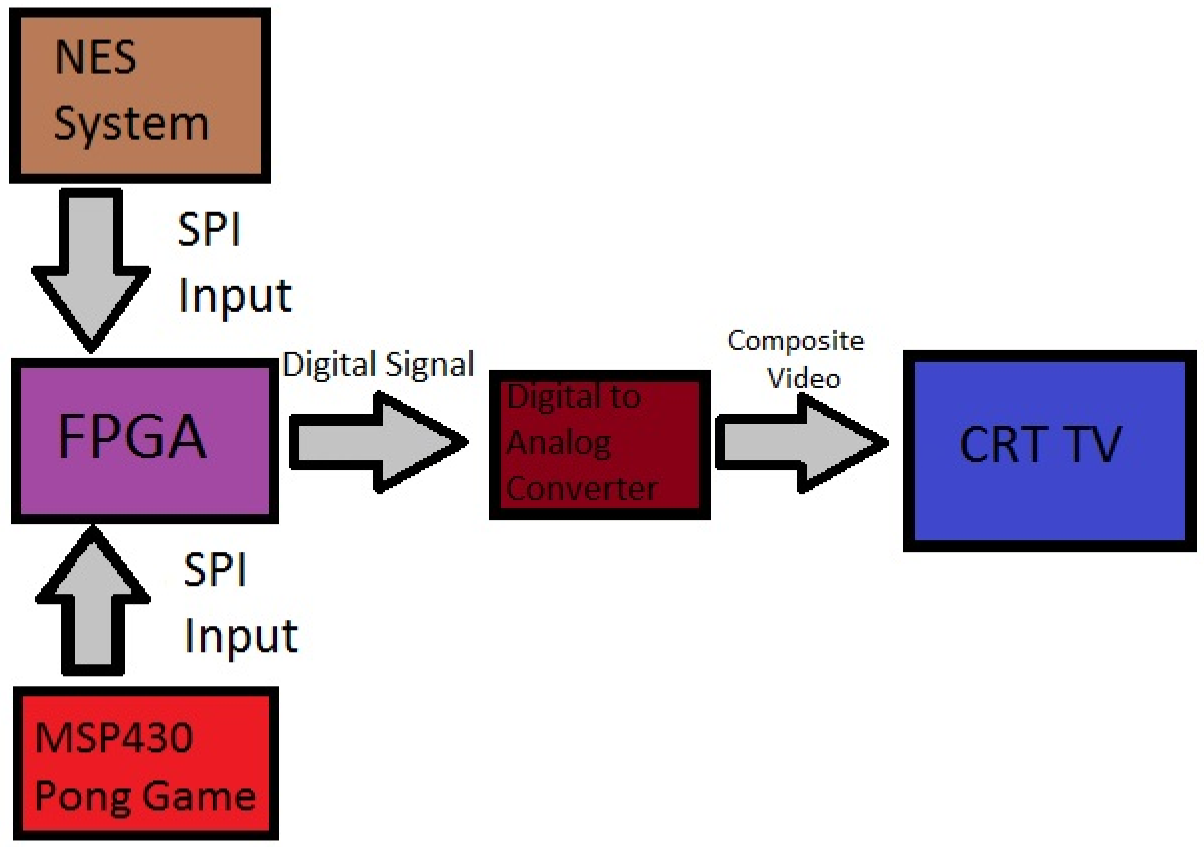

Figure 1: System Overview

Secondary Objectives

Framerate Control

The framerate for a CRT is roughly 30Hz while the shutter rate for most video cameras is 24Hz. When filming a CRT screen, slightly more than one frame will be displayed in the time that the image is captured on the camera, this results in a sweeping, white bar that will appear on the television when viewing the film. This is undesireable.

Many prosumer and professional cameras have an output signal called HSYNC. This signal will be used to trigger the composite video encoder so that only one frame is written to the television during one shutter period. In theory this will remove any erronous artifacts that would normally be seen in the video.

Sound Synthesizer

Adding to the SPI interface, we could add commands that trigger pre-generated sound "bites" which would play in conjunction with pong. These sounds could either be programmed to the encoder through SPI, and saved into RAM or hardcoded into ROM. A DAC would then be needed to drive a speaker. Sound information and playing would be modeled after some standard, possibly something like WAV.

Open MSP430 Synthesis

On Opencores.com an open source IP of the MSP430 core can be downloaded for synthesis in an FPGA. To cut back on the hardware used in the project, we can put this core on our FPGA and program pong onto it. This also adds some complexity as the ADC on the evaluation board would need to be interfaced to the system.

Colour

The colour component of a composite signal is transmitted through the phase and angle modulation of a 3.58MHz carrier. In order to generate this signal we would need to employ a parallel DAC instead of our cheap, lopsided, but inexpensive resistor divider DAC. Another project within our class is to create a Ninstendo Entertainment System on an FPGA, and so we will base our colour output on the NES.

Hardware

To be able to complete this project we will need more hardware than what we currently have. We have a MSP430 microcontroller, the FPGA board, and a mini black and white TV. We need an 8-bit parallel, single channel DAC. We will use the AD9748 from Analog Devices Inc. which can be ordered from Digikey, and we will also need a breakout board for the 32-QFN package which can also be picked up from Digikey. Any hardware we may need, such as parts for filtering will be provided by us.

Implementation

Figure 2: Module Diagram

8-bit DAC Hardware

Composite Driver

Timing Control

Line Buffer

Colour Modulation

SPI Module

The SPI module was designed to take a 9-bit input and a clock input. 8 of the 9 bits was to indicate colour. The last bit was for internal communication. Internal communication included setting individual pixels, resetting the frame and any other special features the input device could use. Our SPI module was designed this way so any device, slow or fast, could be used as an input.

Once the SPI module obtained an input, the module would stack two input pixels and store the input on an array 320 elements long. Each element on the array would store 16-bits or two pixels. This make for easy conversion to the SDRAM.

The SDRAM stores 16 bits worth of data at a time.

RAM Interface

The RAM interface was not completed. Although to get this working we would needed to use NIOS 2 software to give us a file to be able to access the SDRAM. Once this Verilog file would be created we would have to modify it to fit our needs and properly implement it in our system.

Conclusion

Appendix

Module Listings

Colour Module Listings