Light saber generator-Return of the Jedi

CSEE 4840 Project Design - March 2009

Devesh Dedhia, Anusha Dachepally, Roopa Kakarlapudi,Raghu Binnamanglam
Department of Electrical Engineering
Columbia University
{ddd2121,ad2657,rk2489,rsb2145,}@columbia.edu

Abstract

The main goal of this project is to create special effects on an incoming video and display it in real-time. Specifically, we aim to recognize a sword in the input video and replace it with a light saber (of the Star Wars fame!!).

1 Introduction

The light saber generator takes in a live input stream from the digital camera, uses the two blue/green markers on the sword to identify the position of the sword and creates a light saber in its place. A halo effect with varying intensity of color is created around this light saber.
The design has hardware and software components. The hardware component comprises of an I2C bus to transfer data from the processor to the video decoder to configure it, a YCrCb to RGB convertor, an X-Y coordinate locator and an SRAM to store the incoming frame (after RGB conversion).
The software controls the replacement of the sword with a light saber. This is done by running a C code on the NIOS processor that reads the coordinates of the markers (patch) on the sword to calculate the tilt and width of the sword. It then generates the new pixel information and then writes it to the RAM. The figure below is the functional block diagram of the light saber generator and its description:

2 Hardware Design

This section explains the hardware components that go into the light saber generator.

2.1 Video Decoder ADV7181:

The DE2 board is equipped with an Analog Devices ADV7181 TV decoder chip. The ADV7181 is an integrated video decoder that automatically detects and converts a standard analog baseband television signal into 4:2:2 component video data compatible with 16-bit/8-bit CCIR601/CCIR656. The registers in the TV decoder can be programmed by a serial I2C bus, which is connected to the Cyclone II FPGA. The timing diagram of the ADV7181 chip shows that output YCrCb information it obtained at the LLC frequency (27MHz)

Table 53. HS Timing Parameters (see Figure 19)

Standard	Characteristic: HS Begin Adjust (HSB[10:0]) (default)	HS End Adjust (HSE[10:0]) (default)	HS to Active Video (LLC1 Clock Cycles) (C in Figure 19) (default)	Active Video Samples/Line (D in Figure 19)	Total LLC1 Clock Cycles (E in Figure 19)
NTSC	00000000010b	00000000000b	272	720Y + 720C = 1440	1716
NTSC Square Pixel	00000000010b	00000000000b	276	640Y + 640C = 1280	1560
PAL	00000000010b	00000000000b	284	720Y + 720C = 1440	1728

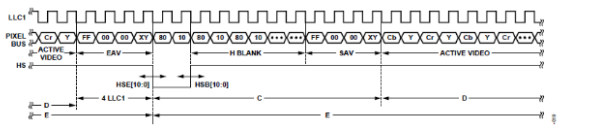

Figure:HS timing

2.2 YCrCb to RGB converter:

The data format of the incoming video stream from the ADV7181 analog Devices video decoder is in YCrCb format. The data processing and display on the VGA is possible only in the RGB format. To accomplish this task we will design an YCrCb to RGB converter. The equations described below are used for the conversion:
B = 1.164(Y - 16) + 2.018(Cr - 128)
G = 1.164(Y - 16) - 0.813(Cb - 128) - 0.391(Cr - 128)
R = 1.164(Y - 16) + 1.596(Cr - 128)
As the Y, Cr, Cb information is multiplied by constants and since floating point multiplication is time consuming, therefore we plan to implement a look-up table method in place of floating point multiplication to do the following conversion.

2.3 The X-Y coordinate generator:

This component reads the complete pixel information after RGB conversion and outputs the coordinates of the markers at the ends of the swords.

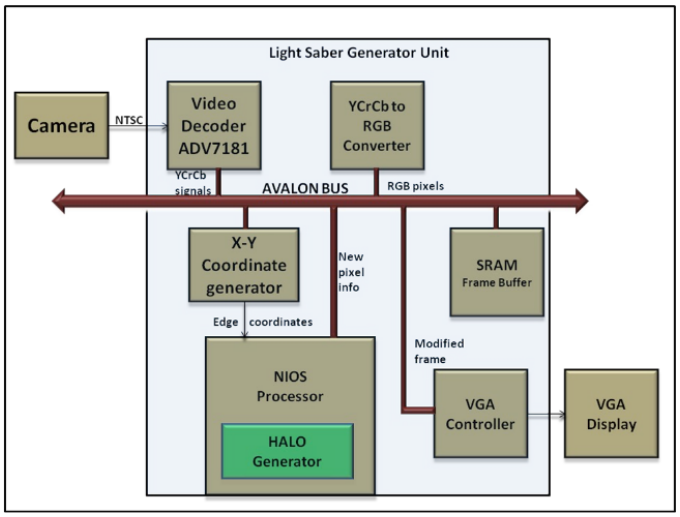

Figure 1: **Block Diagram**

The marker on the top is going to be blue and the one at the bottom green. We need to make sure that these two colors are not present anywhere else in the image.
Each marker occupies a large cluster of pixels in the image. The challenge is to return a single coordinate at the center of each marker. A common method is to flag all blue pixels and find the center of mass of their coordinates. Stray blue/green pixels in the picture would displace the center coordinates significantly causing errors. In order to avoid this we need a filter that removes the stray pixels. One of the methods would be to identify a cluster 7 or more blue/green pixels.The flagging of blue/green pixels can be done by defining thresholds for these colors. In the RGB color space, it is easy to define ranges of R, G, and B values for green and blue colors.

2.3.1 Algorithm to find the center of mass:

To find the center of mass of the blue/green marker

1. Example: If current pixel is blue
   Add the value of it s X and Y coordinates to Xsum and Ysum respectively
2. Increment the number of the Bluepixelcount
3. Center of Mass
   X= Xsum/Bluepixelcount
   Y= Ysum/Bluepixelcount

2.4 SRAM:

The ISSI IS61LV25616 high-speed, 4,194,304-bit static RAM organized as 262,144 words by 16 bits. The functional block diagram of an SRAM on the Altera DE2 FPGA board is as given below:

A0-A17	Address Inputs
I/O0- I/O15	Data Inputs/Outputs
CE	Chip Enable input
OE	Output Enable Input
WE	Write Enable Input
LB	Lower–byte control(I/O0- I/O7
UB	Upper-byte control (I/O8- I/O15
VCC	Power
Gnd	Ground
NC	No connection

The RGB converter stores the pixel data into the SRAM (Frame Buffer). The NIOS processor (HALO generator) which is described in the Software Design section below generates the coordinates which need to be replaced with the new pixel information to generate the light saber. It then writes this information (light saber pixels) into the SRAM overwriting the old pixels.

The timing diagrams for the SRAM module for read and write cycles are as shown below:

2.4.1 Read Timing

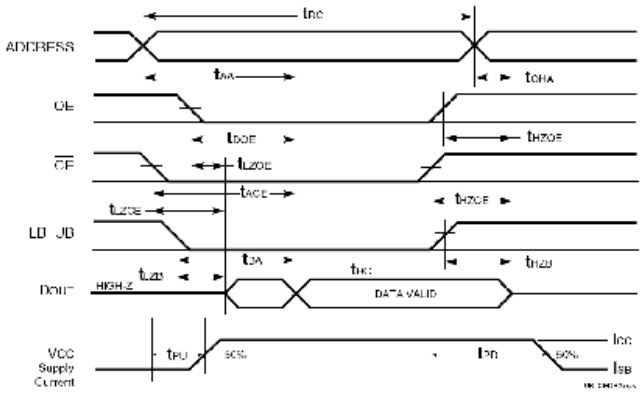

Figure: ReadTiming

2.4.2 Write Timing

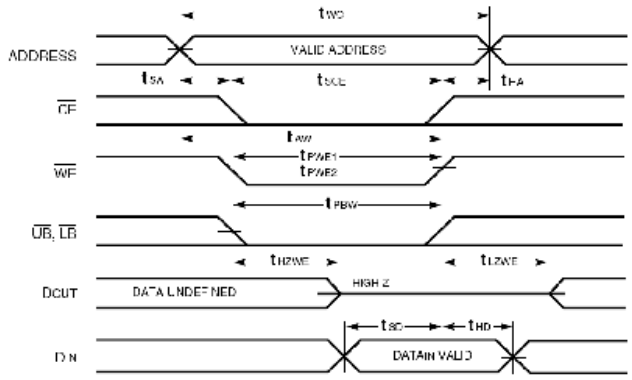

Figure: Write Timing
The internal Write time is defined by the overlap of CE = LOW, UB and/or LB = LOW, and WE = LOW. All signals must be in valid states to initiate a Write, but any can be deasserted to terminate the Write. The t SA, t HA, t SD, and t HD timing is referenced to the rising or falling edge of the signal that terminates the Write.

Alternative approach: Instead of writing the data into a complete frame and then displaying it, we could display the data using a line buffer (display per line basis). Now the block RAM (on-chip) can be used as the line buffer (line buffer size for 320x240 resolution would be 640B considering 1word per pixel).

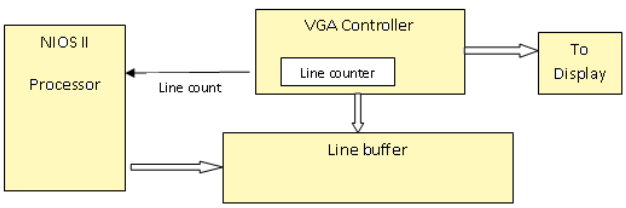

Figure: Write Timing The NIOS processor gets the line count from the VGA controller for every line. Based on the current line count and the information of the light saber pixels (constructed by the C code) the NIOS processor replaces the corresponding pixels (light saber pixels) in the line buffer if any. The pixel data from the updated line buffer is finally displayed on the VGA screen.

3 Software design

The hardware design described above would take in the data from the camera and process it to locate the coordinates of the blue/green markers on the sword. These coordinates are now made available to the software unit through the Avalon bus. The Halo generator uses these coordinates to generate the light saber which will replace the sword in the RAM. The details of this process are as described below:

3.1 Nios Processor

Nios II is a 32-bit embedded-processor architecture designed specifically for the Altera family of FPGAs. Nios II incorporates many enhancements over the original Nios architecture, making it more suitable for a wider range of embedded computing applications
The Nios II/e includes features like the JTAG debug module, optional debug enhancements, and up to 256 custom instructions. The SOPC (System-on-Programmable-Chips), a component of the Quartus-II package is used to configure and generate a Nios system. One can choose the Nios-II's feature-set and add peripheral and I/O-blocks (timers, memory-controllers, serial interface, etc.) to the embedded system. When the hardware specification is complete, Quartus-II performs the synthesis, place route to implement the entire system on the selected FPGA target.

Halo Generator: The code that generates the halo runs on the NIOS II processor, it reads the location of the sword markers and generates a light saber. It sends this information to the SRAM which stores the frame. The code is based on the following algorithm.
The algorithm:

1. Read the coordinates (x1, y1) and (x2, y2) of the markers on the sword.
2. Calculate the slope m = (y2- y1) / (x2- x1)
3. Calculate the angle of tilt of the sword = tan-1m

4. Calculate apparent width of the sword w' = w/sin Where w is the width of the sword in the image when it is vertical.

5. For y = (y1 to y2)

   (a) Find x from the two point line equation (y- y1) = (y2 - y1) * (x- x1) / (x2- x1)

   (b) Find all the light saber white core pixel coordinates – from x – [w'/2] to x + [w'/2]

   (c) Find the halo pixel coordinates from x – [w'/2] – 8 to x – [w'/2] and from x + [w'/2] to x + [w'/2] + 8 (Since the halo is around the saber)

   (d) Send the coordinates and the corresponding to pixel colors of the saber to replace the sword pixels in the RAM.

4 Timing Considerations

The Light Saber Generator has the following main blocks whose timing needs to be taken into account:

- **VIDEO DECODER**: The DE2 board consists of an on-board video decoder; ADV7181 which takes the incoming NTSC format pixel data and converts it into YCrCb information. It operates at 27MHz and produces digital outputs in 8/16 bit formats.

- **SRAM**: The maximum access time required to access one word (16 bits) from SRAM is approximately 20ns. With such access rate, there are two possibilities we will discuss using a frame buffer:

  1. Resolution = 640x480
     Pixel depth = 16 bits (5 bits each R, G and B and 1-bit padding)
     Frame Size = 640x480x16 bits = 600KB
     Access time = 6.14ms (to read the entire frame)
     The size of the SRAM on-board is only 512KB so this cannot be used. Although the size of SDRAM is 8MB it cannot be used due to its higher access time comparatively.
  2. Resolution = 320x240
     Pixel depth = 16 bits (5 bits each R, G and B and 1-bit padding)
     Frame Size = 320x240x16 bits =150KB
     Access time = 1.53ms (to read the entire frame)
     Therefore this option gives us a considerable performance, enabling us to read the incoming pixel data and update the current frames at a rate required to produce real-time video.
  3. Another possibility would be to use line buffer, in which the case the above values would be:
     Resolution = 320x240
     Pixel depth = 16 bits (5 bits each R, G and B and 1-bit padding)
     Line Size = 320x1x16 bits =640B
     Access time = 6.4 us (to read one line)
     We will consider the pros and cons of the aforesaid approaches and based on the limitations and trade-offs faced by our experiments, we intend to choose the efficient approach.

- **YCbCr to RGB CONVERTER:** To accomplish the conversion from YCbCr format to RGB format, we intend to use look-up table method rather than floating point multiplications. To store the data required for implementing the look-up table we will use the embedded Block RAM of the FPGA; Altera Cyclone II EP2C35F672C6. The device family EP2C35 contains 105 M4K memory blocks with total number of RAM bits being 483840bits, i.e.60KB. The latency involved in accessing the Block RAM is negligible.

5 Milestones

1. 1st milestone : Getting the output of the camera (NTSC) to display on the VGA screen
2. milestone (Halo Effect Generation) – Identifying and using the coordinates of the ends of the sword, determine the pixel information for producing the halo effect.
3. milestone (Optimization) – Reduce the pixel processing time to produce frames at a rate fast enough to generate video.

6 References

1. Altera Corporation. Avalon Memory-Mapped Interface Specification. www.altera.com
2. Altera Corporation. Cyclone II Device Handbook. www.altera.com
3. Altera Corporation. NIOS II Processor Reference Handbook. www.altera.com

4. http://www.terasic.com
5. Timing Considerations with VHDL-Based Designs- A handbook
6. Altera University Program Forums