Project Daltonismo

Cody Anderson
Ben Nollan

February 16, 2017

Jeremy Thomas Lukas VanGinneken

Dept. of Electrical Computer Engineering

ECE 310L, 3^{rd} Year CE Project

Abstract

Sufferers of color vision deficiency (CVD) report that videogames rarely take CVD into account during design. Sufferers of CVD are left at a distinct disadvantage during gameplay, especially when color is an integral part of gameplay. Project Daltonismo aims to create a device that transforms the set of colors used in a video signal, such that users who suffer from CVD can more easily discern colors from one another. The aforementioned device will be constructed on an FPGA. The device will have access to the video signal by being connected between the video source and display via a High Definition Multimedia Interface (HDMI) input and an HDMI output. Color Transformations will be done by first converting the original image from the RGB (for Red, Green and Blue) color space into a second color space, transformed to move colors out users' ambiguous hue range, converted back into the RGB color space, and then transformed once again, to minimize image distortion. Daltonization is a popular technique of colorblindness transformation that will be investigated and, based on feedback from users, may be used. Transformation of the video signal will be done in real time at a minimum of 60Hz, with much less than a frame of latency, taking advantage of the speed of the FPGA.

1 Introduction

For this project we are looking to create a device that transforms the color space for the people with CVD. The end goal is a device that will alter the color space when plugged in between an HDMI source and HDMI sink requiring no modification to either device to operate. It will have switches to switch between different color filters for different types of CVD. We will consider our device a success if video is outputted when inputted and color filters are applied when switches are flipped. Previous work of getting HDMI passthrough on an FPGA was done by Mike Field using VHDL on the Nexys Video FPGA [1]. Similar work of making colors easier to differentiate was done by mapping words or slashes to the colors [2].

2 Methods, Techniques, and Design

Our device will take HDMI as an input and output HDMI, while the signal is internal to the FPGA, we will perform some color manipulation to make the colors more differentiable (Figure 1).

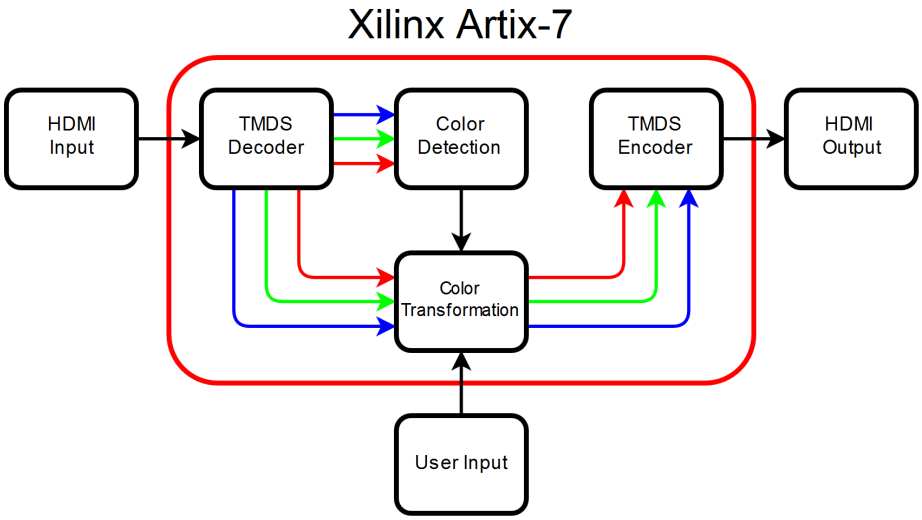

Figure 1: Block Diagram

2.1 Color Conversion

Color transformation begins by converting colors to a new color space from RGB, the color space they are provided in by the HDMI signal. The colors are then transformed within the new color space. After the new transformation, colors are then converted back into RGB and are given one last operation from within RGB in order to fine tune the output image (Figure 2).

The RGB color space is ubiquitous in cameras, computers, video sources and video displays. For what RGB provides in ubiquity, it introduces new challenges in difficulty of color transformation. Each color space provides its own benefits and drawbacks, depending on the use case. In the case of Project Daltonismo, transformations outside of RGB will be done within the LMS color space or within the HSV color space, depending on what algorithm is used.

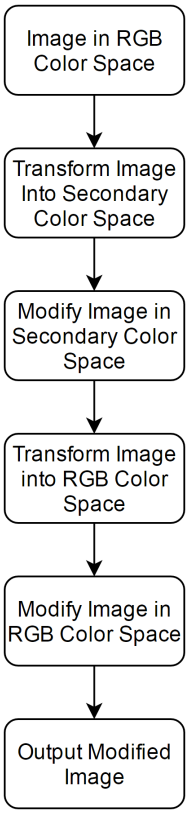

Figure 2: Flowchart of Image transformation sequence.

2.1.1 Daltonization

Daltonization is an algorithm proposed to utilize the way the LMS color space inherently supports a matrix of color responses, or a Chromatic Adaptation Transform (CAT) matrix. Daltonization is preformed in four basic steps:

1. Convert RGB into LMS
2. Simulate colorblindness with a CAT matrix
3. Shift the colors that are mapped closely on the CAT matrix away from each other
4. Convert LMS back into RGB

Since one can easily change how the different levels of colors are perceived in the LMS color space, daltonization naturally lends itself to using the LMS color space.

2.1.2 Red-Stripes Method

The Red-Stripes Method of color transformation, starts by converting into the LMS color space, like daltonization, in order to detect which colors are closely

mapped. The difference comes when the colors are converted into the HSL color space. The HSL color space allows for easily brightening colors without altering their saturation. The Red-Stripes uses this to brighten stripes in some of the colors that would be hard to tell apart so that those which are brightened and those which aren't an be differentiated.

2.2 HDMI

High Definition Multimedia Interface(HDMI) is an interface capable of transporting video and audio simultaneously. It uses four differential signaling wire pairs, three for pixel data and one for the pixel clock. There are 10 bits sent across each of the pixel data pairs for one transition of the clock pair [3].

HDMI was chosen as the transmission medium for this project because it is very commonly used for video output on consoles and computers. To give good compatibility with other devices, our project is capable of a maximum resolution of 1920x1080 at 60Hz, the clock rate at this resolution is 148.5Mhz and the bit rate on the three data pairs will be 1.485GHz.

2.2.1 TMDS Encoding

HDMI and DVI use a signal encoding technique called Transition Minimized Differential Signaling (TMDS) to reduce the EMI that the cable experiences. This encoding scheme performs either the XOR or the XNOR operation on the inputted bit with the previously derived bit and adds a ninth bit to represent whether XOR or XNOR was used (Figure 3). Next, a tenth bit is added to represent whether the entire bit is inverted. This is done to maintain an overall DC balance of the Differential signaling lines. Since neither Xor or Xnor is the best choice in all cases (Figure 4), first a bit count is used to determine the encoding used (Figure 5). This gets the maximum transitions to five, with one more check on the least significant bit (Figure 6) the maximum number of transitions are lowered to four.

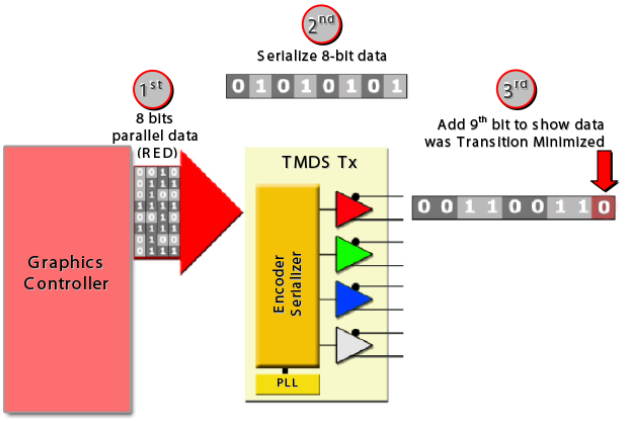

Figure 3: TMDS Encoding [4]

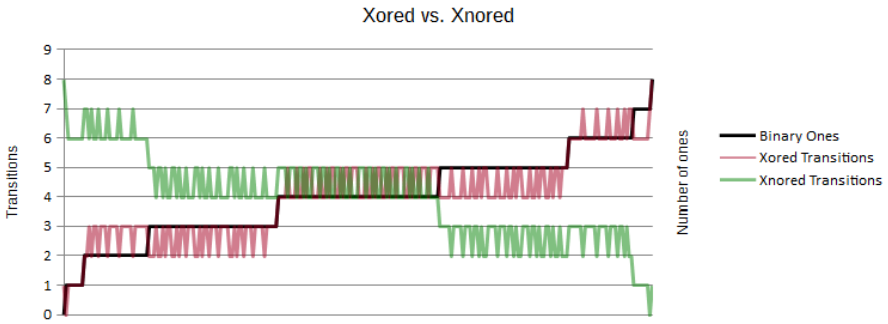

Figure 4: Xor vs. Xnor

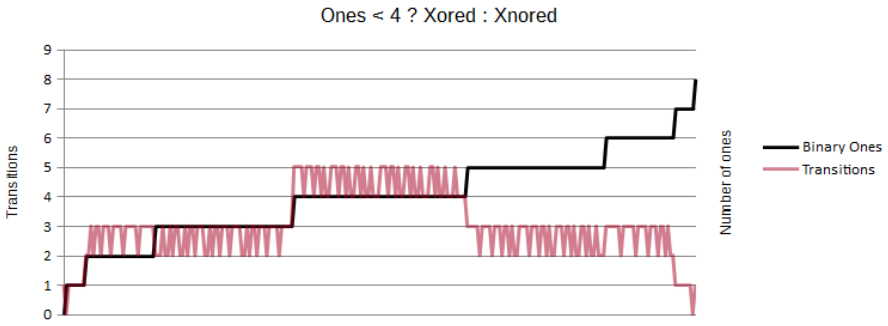

Figure 5: First Optimization

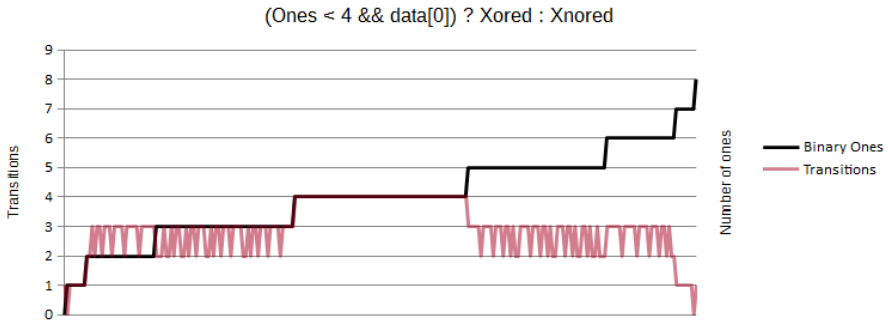

Figure 6: Second Optimization

2.3 HDMI Signal Decoding

To decode the incoming HDMI signal, the data first gets aligned, this is done by delaying the signal to match the internal clock phase using a delay primitive (IDELAYE2). Then, the data needs to be converted from a serial data stream into a parallel stream by instantiating a serial to parallel converter primitive

(ISERDESE2). This converter is only capable of converting an 8-bit wide signal, so two connected together are required for the 10-bit wide HDMI symbols. At this point in the process, the parallel data is probably not aligned, this is corrected by having the serial to parallel converter drop bits until the data outputted is valid.

2.4 Build and Implementation

As all projects do, Project Daltonismo was met with a good number of challenges that influenced how color blindness compensation was implemented.

2.4.1 Bitslipping

The initial design of the serial to parallel conversion had a switch tied to bitslipping, this was tedious as the switch would have to be flipped around five times on average to get the data alignment correct. The solution applied to correct this was to count the number of valid data symbols received, if one of the symbols out of 65535 was invalid, it would bitslip and restart the counting. This worked for certain devices but not others, this was due to the fact that some devices output the data channels with the bits misaligned with the other channels. To fix this the bitslipping process was applied to each channel individually instead of applied to all three at once.

2.4.2 Matrix Multiplication

One of the first challenges reached was Matrix Multiplication. Matrix multiplication is useful for many color space transformations and is necessary for the desired RGB-XYZ and XYZ-RGB transformations. The original implementation of Matrix Multiplication was a very naive solution of doing all the multiplications necessary of all the input numbers and the constants that would be placed in associated transformation matrix in order to complete a transformation. The very first problem with this method was that all numbers and entered constants in System Verilog are stored as an unsigned integer representation of that number. This means that any decimal value more precise than the 1's place is truncated off, any numbers that grow in size too large lose data due to multiplication overflow and that negative values are not represented in any ways. To add onto this, each multiplication was done in one clock cycle. Operating at the HDMI signal's 60fps 1080p timing of 148.5 MHz, this required the transistors in the FPGA to operate far faster than they are capable when taking into account the propagation delay of such a large operation. The result was a non-deterministic output, which still worked on an HDMI display, but resulted in absolutely incorrect output and very significant visual artifacts.

2.4.3 Fixed-Point Number Representation

As a solution to the inadequate integer number representation, a specific fixed-point number representation for the project was devised, with 10 whole number bits (enough to represent the largest possible number during transformation) and 13 decimal number bits (enough in order to accurately represent the smallest possible number during transformation) and one sign bit place so that negative

numbers could also be represented. Along with the fixed point implementation, multiplication for this specific fixed point was also implemented, handling cases for negative numbers and handling the operation of shifting the numbers correctly after multiplication in order to retain each bit's correct significance. Division was represented as a multiplication by a decimal. While the results were correct for the fixed point multiplication, it did nothing to solve the issues that existed with attempting to do all the necessary multiplications in one 148.5 MHz clock cycle, so visual artifacts remained and the color transformations between RGB and XYZ color spaces, never came to fruition.

2.4.4 HSV Transformation

With deadlines fast approaching on the project and with much work still to do to get one colorblindness compensation method working, the color space transformations between RGB and XYZ were not finished because transformations between RGB and HSV were much simpler to implement than the matrix multiplication was to debug at the time. The implementation of transformations between RGB and HSV color spaces went off largely without a hitch. The main sticking point encountered was one that was all too familiar: pushing the FPGA transistors faster than they're comfortable with. In this case two instances 8 bit integer divisions were being attempted every 148.5MHz clock cycle as apposed to the nine instances of 24 bit fixed-point multiplications attempted in the previously attempted matrix multiplication. Since the load was significantly less on the transistors for this operation, it was guessed that it would be able to be accomplished in only one clock cycle. The propagation delay of even just the 8 bit division was too great to work correctly on a 60Hz 1080p HDMI signal, but it was working properly on a 60Hz 640x480 pixel resolution HDMI signal, due to the clock cycle being far smaller, somewhere closer to a 18.5 MHz clock, allowing a good portion more time for operations to propagate. This lead to the solution of pipelining the division operations.

2.4.5 Division Operation Pipelining

Pipelining the division operations allowed each division to take more clock cycles, so that each small part would only introduce only a small propagation delay, which was lesser than the time between each clock cycle. In exchange for the ability to do more complex operations on an every clock cycle basis, the output of the operation is essentially delayed by the number of operations which have to be completed. Every clock cycle, the output of the current operation is then clocked into the input of the next operation. This means that for the first number of clock cycles equal to the number of operations to be completed for a division, the output of the division operation is undefined and it also means that non-pixel signals, such as the horizontal sync (HSync) and vertical sync (VSync) also have to be delayed in order to match up with the outputted pixel values. If the HSync and VSync the entire image will be shifted on the screen in the direction in which scan lines travel by the number of pixels equal to the number of operations in a pipelined division. This same kind of pipelining can also be used to overcome problems with the matrix multiplications required for transformations between RGB and XYZ.

2.4.6 Color Blindness Compensation Utilizing the HSV Color Space

Any color can be broken down into a hue, saturation, and value. (the H, S, and V of HSV, respectively) The difference between full color vision and vision as experience by a colorblind can be modeled as a set of pairs of hues which appear very similar (confusion pair), or even identical as a color blind individual. Both of these occurances can be used to create a sort of easy to implement and quite effective color blindness compensation. This is done by checking if the hue of the current pixel belongs to a confusion pair and, if so, shifting encoding the hue into the value and saturation of the color. Since every confusion pair has one higher hue value and a lower hue value, a lower hue will always be encoded in one way and a higher hue will always be encoded in another way. This ensures that all colors of nearby hues are compensated for in a similar way, avoiding jarring sudden changes in colors, allowing a non-colorblind individual to see the image fairly normally.

3 Parts

Digilent Nexys Video Part#:410-316 - $490 or $290 (With Educational Discount)
2-HDMI male to male cables - $12

4 Testing and Design Verification

4.1

4.2

5 Discussion

The goals for this project were to have it apply a color transformation on an HDMI signal and operate at 1920x1080 resolution at 60Hz. To test this our device was hooked up between a laptop and a monitor and was used for several days. During this time no major issues arose.

6 Conclusions and Future Work

6.1 Signal Decoding/Encoding

The project at this point decodes and encodes the signal properly. One possible improvement to the decoding is to automate the delaying of the signal. This is currently done manually using switches on the development board. Another improvement is to handle audio that is sent with the video data. Currently it doesn't support audio with the video.

6.2 Color Transformations

The current transformations that are being performed are rough approximations of what they should be, with future work these should be made much more accurate.

7 Acknowledgements

A special thanks to:

- Jeremy Thomas
- Lukas VanGinneken
- Nick Rivera
-

8 Author Contributions

- Ben's Acts:
  - HDMI Input
  - Monitor Identification
  - TMDS Decoding
  - TMDS Encoding
  - HDMI Output
  - RGB to HSV
  - HSV to RGB
  - Division Pipelining
- Cody's Acts:
  - HSV Transformation Prototyping
  - XYZ Transformation Prototyping
  - Colorblindness Simulation Prototyping
  - Colorblindness Correction Prototyping
  - RGB to HSV
  - HSV to RGB
  - All authors contributed equally to this paper.

9 Appendix

9.1 Color Space Definitions

RGB The RGB color space stands for red, green, and blue. RGB is a color space wherein colors are represented by values of red (∼665nm), green (∼550nm), and blue (∼470nm) light. This color space is what is used for most photographs, videos, cameras and electronic displays.

HSV The HSV color space stands for hue, saturation, and value. HSV is a color space wherein colors are represented by their hue (the base color, e.g. red, orange, yellow, green, etc.), their saturation (how close the color is to the pure color and far from a gray color) , and value (how bright the color is). Hues are represented on a 360° spectrum, with red (∼665nm) at both 0° and 360°, orange (∼630nm) at 30°, yellow (∼600nm) at 60°, green (∼550nm) at 120°, blue (∼470nm) at 240°, indigo (∼425nm) at 270°, and violet (∼400nm) at 300°.

XYZ The XYZ color space is very similar to the RGB color space. It is also a color space wherein colors are represented by values of red, green, and blue. The difference in the XYZ color space is that the primary colors it uses are based off which primary colors human eyes are actually sensitive to. XYZ's red is a range of hues ∼564–580nm instead of RGB's ∼665nm, XYZ's green is ∼534–545nm instead of RGB's ∼550nm, and XYZ's blue is ∼420–440nm instead of ∼470nm.

LMS The LMS color space is very similar to the XYZ color space since it has the goal of modeling human vision. This combined with the fact that the XYZ color space uses the wavelength bands which human eyes can primarily sense, as the three primary colors, allows for easy transformation between the XYZ and LMS color spaces. Theses three colors with differently weighed color responses are based on the three primary bands of color from the XYZ color space, which are directly based on the three bands of color that human eyes can perceive. The inherent way that LMS simulates human color vision stems from the process of conversion into LMS. The only step necessary is to use a Chromatic Adaptation Transform (CAT) matrix which would map the primaries from XYZ color space into the amount that they stimulate in the LMS color space. The problem is that there is no definitive transformation matrix, though various Color Appearance Models (CAMs) offer their of CAT matrices.

References

[1] Mike Field. Artix 7 1080p passthrough, 2015.

[2] David R. Flatla, Alan R. Andrade, Ross D. Teviotdale, Dylan L. Knowles, and Craig Stewart. Colourid: Improving colour identification for people with impaired colour vision. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3543–3552, New York, NY, USA, 2015. ACM.

[3] DDWG. *DVI Spec Rev 1.0*. 1999.

[4] Lars Weinand. *TMDS Encoding*. 2004.

bibliography.bib

```
@misc{field_2015,
author={Field, Mike},
title={Artix 7 1080p passthrough},
url={http://hamsterworks.co.nz/mediawiki/index.php/Artix_7_1080p_passthrough},
urldate={2016-10-4},
journal={HamsterWorks},
year={2015}
},

@book{dvi_spec_1999,
author={DDWG, },
title={DVI Spec Rev 1.0},
url={https://web.archive.org/web/20120813201146/http://www.ddwg.org/lib/dvi_10.pdf},
urldate={2016-10-4},
year={1999}
},

@book{weinand_2004,
author={Weinand, Lars},
title={TMDS Encoding},
url={http://img.tomshardware.com/uk/2004/11/29/the_tft_connection/tmds-transmitter.jpg},
urldate={2016-10-4},
year={2004}
},

@inproceedings{Flatla:2015:CIC:2702123.2702578,
 author = {Flatla, David R. and Andrade, Alan R. and Teviotdale, Ross D. and Knowles, Dyla
 title = {ColourID: Improving Colour Identification for People with Impaired Colour Vision
 booktitle = {Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing
 series = {CHI '15},
 year = {2015},
 isbn = {978-1-4503-3145-6},
 location = {Seoul, Republic of Korea},
 pages = {3543--3552},
 numpages = {10},
 url = {http://doi.acm.org/10.1145/2702123.2702578},
 doi = {10.1145/2702123.2702578},
 acmid = {2702578},
 publisher = {ACM},
 address = {New York, NY, USA},
 keywords = {colour identification, colour namers, colour vision deficiency, colourblindne
}
```