

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES

INTERACTIVE PARALLEL VIDEO EDITING
USING FPGA STRUCTURES

by
Okan ÇOBANOĞLU

September, 2011
İZMİR

INTERACTIVE PARALLEL VIDEO EDITING USING FPGA STRUCTURES

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering**

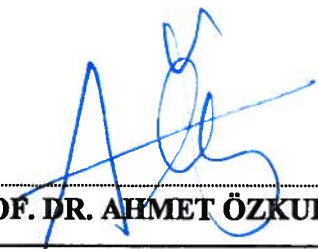
**by
Okan ÇOBANOĞLU**

September, 2011

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**INTERACTIVE PARALLEL VIDEO EDITING USING FPGA STRUCTURES**” completed by **OKAN ÇOBANOĞLU** under supervision of **ASST. PROF. DR. AHMET ÖZKURT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


ASST. PROF. DR. AHMET ÖZKURT

Supervisor


Prof. Dr. Uğur Cam
(Jury Member)


Doc. Dr. İdris Şahin
(Jury Member)


Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Ahmet ÖZKURT for his guidance and endless patience during the study. He has always encouraged and supported me during the thesis work.

I also would like to thank to my friend Serkan ÖZYÜREK for his absolute guidance, supports and patience during this research. His endless friendship and encouragement always motivated me.

Finally, I would like to thank my parents Tülay ÇOBANOĞLU and Cihan ÇOBANOĞLU who have the highest effort on the status I have today, and I would like to thank to my brother Okay ÇOBANOĞLU for his continuous support during this research.

Okan ÇOBANOĞLU

INTERACTIVE PARALLEL VIDEO EDITING USING FPGA STRUCTURES

ABSTRACT

In this thesis, a real time video processing system is proposed and it is implemented on a Field Programmable Gate Array (FPGA) platform. 3D video applications including frame sequential video, field sequential video and anaglyph video are implemented by using the parallel processing feature of the FPGA. In addition, other common industrial applications such as picture in picture and chroma keying are added to the system.

This thesis is realized on Altera DE2-70 FPGA development board and is divided into two parts. In the first part, the video frames from a single camera is buffered in the FPGA board and sent to the monitor without any additional video processing operations. In the second part, the second camera is added to the system. In this proposed video processing system, two video sources are used for the parallel video applications. Firstly, composite video inputs in NTSC format are converted to digital video format. TV decoder ICs are used to decode the analog video inputs. Then, SDRAMs are used to buffer the video frames. All the video preprocessing operations and video applications are implemented in FPGA hardware. Finally, the parallel video applications are combined into one design and implemented. The processed video is sent to the VGA output of the board.

In order to evaluate the processed video, some special devices are also used. Head mounted display is used for the frame sequential video application and colored glasses are used for anaglyph video application.

Keywords: Field Programmable Gate Array (FPGA), real time video processing, anaglyph, chroma keying, frame sequential video, field sequential video, picture in picture

FPGA YAPILARI KULLANARAK PARALEL ETKİLEŞİMLİ VIDEO EDİTLEME

ÖZ

Bu tezde, Sahada Programlanabilir Kapı Dizleri (SPKD) ortamında gerçekleştirilmiş gerçek zamanlı bir video işleme sistemi önerilmiştir. SPKD'nin paralel işleme özelliği kullanılarak, çerçeve sıralı video, alan sıralı video, anaglif video gibi 3 boyutlu video uygulamaları gerçekleştirilmiştir. Ayrıca buna ek olarak, resim içinde resim ve renk anahtarlama gibi endüstriyel video uygulamaları da sisteme dahil edilmiştir.

Bu tez, Altera DE2-70 SPKD geliştirme kartı üzerinde gerçekleştirilmiştir ve iki bölümden oluşmaktadır. İlk bölümde, bir kameradan alınan video çerçeveleri SPKD kartında tamponlanmış ve fazladan hiçbir video işleme işlemi yapılmadan ekrana gönderilmiştir. İkinci kısımda, ikinci bir kamera sisteme eklenmiştir. Önerilen bu video işleme sisteminde, paralel video uygulamaları için iki video kaynağı kullanılmıştır. Öncelikle NTSC formatındaki kompozit video girişleri dijital video formatına çevrilmiştir. Analog videoyu çözmek için TV çözücü entegreler kullanılmıştır. Daha sonra, SDRAM'ler de video çerçevelerini tamponlamak için kullanılmıştır. Tüm video ön işleme işlemleri ve video uygulamaları SPKD donanımı üzerinde uygulanmıştır. Son olarak, paralel video uygulamaları tek bir tasarımda birleştirilmiş ve uygulanmıştır. İşlenmiş video çıkışı SPKD kartının VGA çıkışına gönderilmiştir.

İşlenmiş videoyu değerlendirebilmek için bazı özel cihazlar kullanılmıştır. Çerçeve sıralı video uygulaması için sanal gerçeklik başlığı, anaglif video uygulaması için renkli gözlükler kullanılmıştır.

Anahtar Sözcükler: Sahada Programlanabilir Kapı Dizileri(SPKD), gerçek zamanlı video işleme, anaglif, renk anahtarlama, çerçeve sıralı video, alan sıralı video, resim içinde resim

CONTENTS

	Page
M.Sc THESIS EXAMINATION RESULT FORM.....	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
CHAPTER ONE - INTRODUCTION.....	1
1.1 How 3D Effect Works	1
1.2 3D Display Technology.....	2
1.3 Real Time Video Processing	4
1.4 Scope of the Thesis.....	5
1.5 Outline of the Thesis	8
CHAPTER TWO - BACKGROUND OF VIDEO.....	9
2.1 Introduction	9
2.2 Analog Video	9
2.2.1 NTSC Composite Video Interface	10
2.2.2 PAL Composite Video Interface	10
2.3 Digital Video	11
2.4 Subsampled YCbCr Formats.....	13
2.4.1 4:4:4 YCbCr Format.....	13
2.4.2 4:2:2 YCbCr Format.....	14
2.4.3 4:1:1 YCbCr Format.....	14
2.5 De-interlacing Techniques	15

2.5.1 Weave De-Interlacing- Field Combination	15
2.5.2 Bob De-Interlacing (Line Doubling) - Field Extension	16
2.5.3 Scan Line Interpolation De-Interlacing - Field Extension	17
2.6 Color Spaces.....	17
2.7 YCbCr to RGB Color Space Transformation.....	18
2.8 VGA Interface	18
 CHAPTER THREE - PARALLEL VIDEO APPLICATIONS	 21
3.1 Introduction	21
3.2 Frame Sequential Video	21
3.3 Field Sequential Stereoscopic Video.....	23
3.4 Anaglyph Stereo	24
3.5 Chroma Keying	27
3.6 PIP (Picture in Picture).....	28
 CHAPTER FOUR - PROGRAMMABLE LOGIC DEVICES	 30
4.1 Evolution of the Programmable Logic Devices	30
4.2 FPGA Architecture.....	32
4.2.1 Logic Elements.....	33
4.2.2 Logic Array Blocks (LAB)	34
4.3 FPGA Design Entry.....	35
4.3.1 Schematic Design Entry	35
4.3.2 Hardware Description Languages	36
4.4 FPGA Simulation	37
4.5 FPGA Design Software (Synthesis Tool)	37
4.6 FPGA Development Board	38

4.7 Signal Tap II Logic Analyzer	40
4.8 MegaWizard Plug-In Manager	41
CHAPTER FIVE - IMPLEMENTATION OF MAIN VIDEO BLOCKS	42
5.1 Introduction	42
5.2 TV Decoder	44
5.2.1 I2C Interface	45
5.2.2 I2C Configuration	46
5.3 ITU-R 656 Decoder Block	48
5.3.1 Down-sampling the Video Lines	48
5.3.2 Extracting 4:2:2 YCbCr Data	49
5.4 SDRAM Frame Buffer Block	50
5.4.1 Writing Frame Buffer	50
5.4.2 Reading Frame Buffer	51
5.4.3 De-interlacing	52
5.5 YUV422 to YUV444 Block (Chroma Resampler)	53
5.6 YCbCr to RGB Block	54
5.7 VGA DAC IC	54
5.8 VGA Controller Block	55
CHAPTER SIX - IMPLEMENTATION OF VIDEO APPLICATIONS	57
6.1 Introduction	57
6.2 Frame Sequential Video Implementation	60
6.3 Field Sequential Video Implementation	62
6.4 Chroma Keying Implementation	63
6.5 Anaglyph Implementation	68

6.6 PIP (Picture in Picture) Implementation	71
6.7 Combination of the All Designs in One Implementation.....	74
6.8 Project Design Report	81
CHAPTER SEVEN - CONCLUSIONS AND FUTURE WORK.....	82
7.1 Conclusions	82
7.2 Future Work	83
REFERENCES.....	84

CHAPTER ONE

INTRODUCTION

Recently 3D techniques and technologies have been developing rapidly. Hardware has both improved and become considerably cheaper which makes the real time and interactive video available to the hobbyist, as well as to the researchers. There have been major studies in such areas as molecular modeling, photogrammetry, flight simulation, CAD, visualization of multidimensional data, medical imaging, tele-operations such as remote vehicle piloting and remote surgery, and stereolithography (McAllister).

1.1 How 3D Effect Works

Each of our eyes sees the world from slightly different perspectives. Our brain integrates these two views into a three-dimensional picture. The key element in producing the stereoscopic depth effect is parallax which is the horizontal distance between corresponding left and right image points. The process that creates the 3D images is known as stereoscopy. Stereoscopic image is composed of two images which correspond to the right and left views of the viewer. By presenting two offset images separately to the left and right eye of the viewer, the viewer can perceive the 3D depth (Engdahl, 1997).

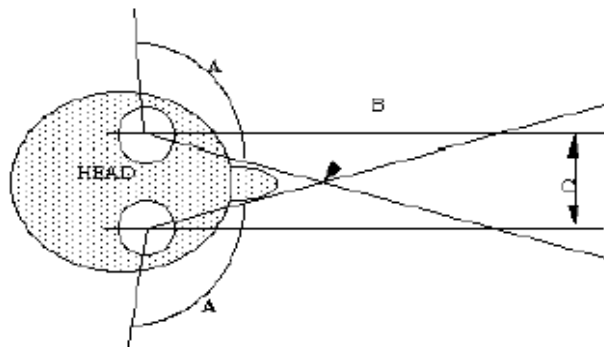


Figure 1.1 Human eye parallax

1.2 3D Display Technology

3D display technology had been developed about one hundred and fifty years ago. From 1850 to 1930, Brewster invented stereoscope and succeed in commercial. A stereoscopic named View-Master became more popular product from 1940 to 1950. Then, vectorgraph and earlier relief television had come into humans' life. Now, 3D display technology has been greatly improved (Zhu & Zhen, 2009).

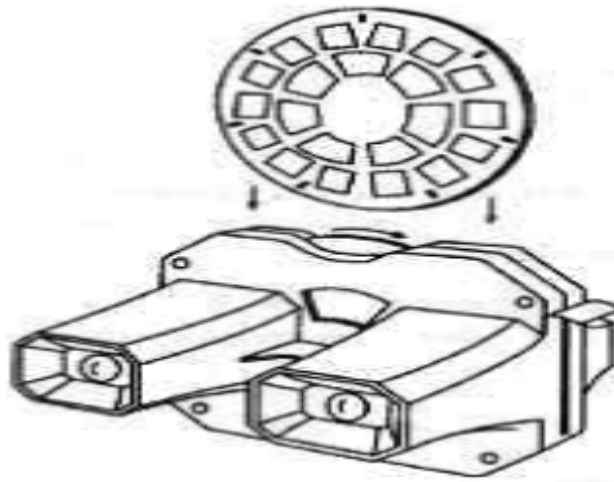


Figure 1.2 View Master (David F. McAllister)

Today, 3D display technology can be classified into several types shown as Figure 1.3. The basic 3D display requirement is to display offset images that are filtered separately to the left and right eye. 3D stereoscopic display methods are typically classified into two types:

- With lenses:
 - Anaglyph 3D (with passive colored lenses)
 - Polarization 3D (with passive polarized lenses)
 - Field sequential, frame sequential systems (with active shutter lenses)
 - Head-mount system (with separate display positioned in front of each eye)
- Without lenses:
 - Autostereoscopic displays (lenticular and LCD)

In addition, 3D image implementation using a holographic video is being developed (Huh, 1999).

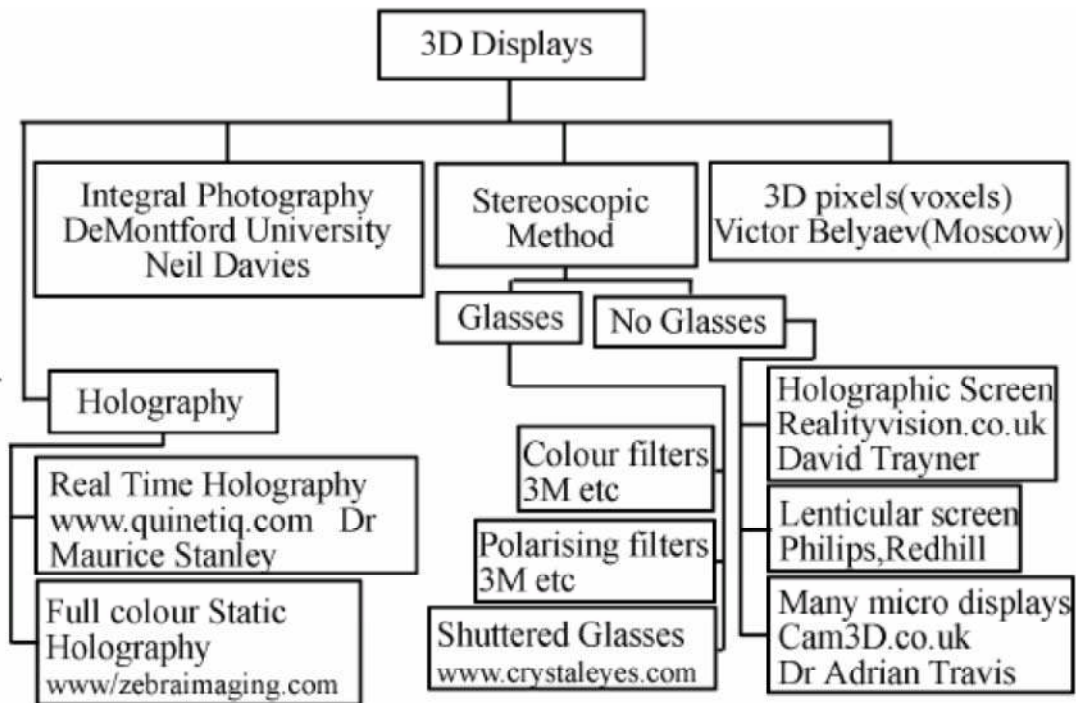


Figure 1.3 Classification of 3D display technologies (Zhu & Zhen, 2009)

In Figure 1.4 there is an example of the anaglyph image. Anaglyph image contains multiple views and each view has a specific color in the image. Special glasses having different color for each lens are needed to view the anaglyph images. Each lens allows one view to pass through the lens to the eye and blocks the other. Then, human visual system merges the stereo views, to create the impression of stereo vision (Gallagher, 2010).



Figure 1.4 Example of anaglyph image (Dubois, 2000)

1.3 Real Time Video Processing

Real-time video and image processing is used in a wide variety of applications from video surveillance and traffic management to medical imaging applications (Neoh & Hazanchuk). Designers of the first generation video systems typically implement video processing algorithms in software using DSP (Digital Signal Processor) devices, microprocessors or multicore processors. However, digital signal processors or other standard processors cannot accommodate new requirements of the next generation video applications for high video resolutions, high frame rates and algorithm complexity. These operations typically require very high computation power. Digital signal processors certainly have advantages for video applications, including software programmability using the C language, relatively high clock rates and optimized libraries that allow for quick development and testing. However, the number of instructions they can perform in parallel is limited in DSPs. They also have a fixed number of multiply/accumulators, fixed instruction word sizes and limited I/O (Pellerin, 2009).

On the other hand, FPGA devices are capable of performing hundreds or even thousands of multiply operations simultaneously in parallel on data of varying widths. Because of the advantages in compute-intensive video processing, FPGAs or

combinations of FPGAs and DSPs have become the favored choice for designers of the most advanced video systems (Pellerin, 2009).

1.4 Scope of the Thesis

The main goal of this thesis is to design a real time video processing system based on the FPGA structures. 3D video applications including frame sequential video, field sequential video and anaglyph stereo video are implemented. In addition, other common industrial applications such as picture in picture (PIP) and chroma keying are studied in the scope of this thesis. A real time parallel video processing system based on hardware without using standard processors or DSPs is proposed in this study.

First of all, FPGA boards with two video inputs and one video output are searched in the market. Altera DE2-70 board which is the most appropriate FPGA board for parallel video editing applications in terms of performance and price is chosen. The thesis is divided into two parts. In the first part, the captured video signals from a single camera is buffered on the Altera DE2-70 board and sent to the monitor without any additional video processing operations. In the second part, the second video source is added to the system and the parallel video applications are implemented and combined into one design.

Two cameras which can supply standard NTSC video output are used in this thesis work. Firstly, the implementations of the video applications are performed in MATLAB to see the design issues in FPGA. Then, VHDL and Verilog hardware description languages and Quartus II synthesis tool is used for the FPGA design and simulations. Signal Tap II logic analyzer software is used for debugging the internal signals. Also a menu interface is designed to select the required application by using the push-buttons on DE-2 70 board.

Figure 1.5 illustrates the general block diagram of the system studied during this thesis. As seen in figure, composite video signals from the cameras are digitized by

using ADV7180 TV decoder chips. Captured video frames of the both video sources are buffered in SDRAMs. All the video processing operations are performed in Cyclone II FPGA. According to the selected video application, the result of the processed video output is changing. After the video processing operations, the processed video is sent to the VGA output. Some special devices are needed to evaluate the output results of the FPGA. Head Mounted Display (HMD) is used for the frame sequential application and colored filter glasses are used for anaglyph application.

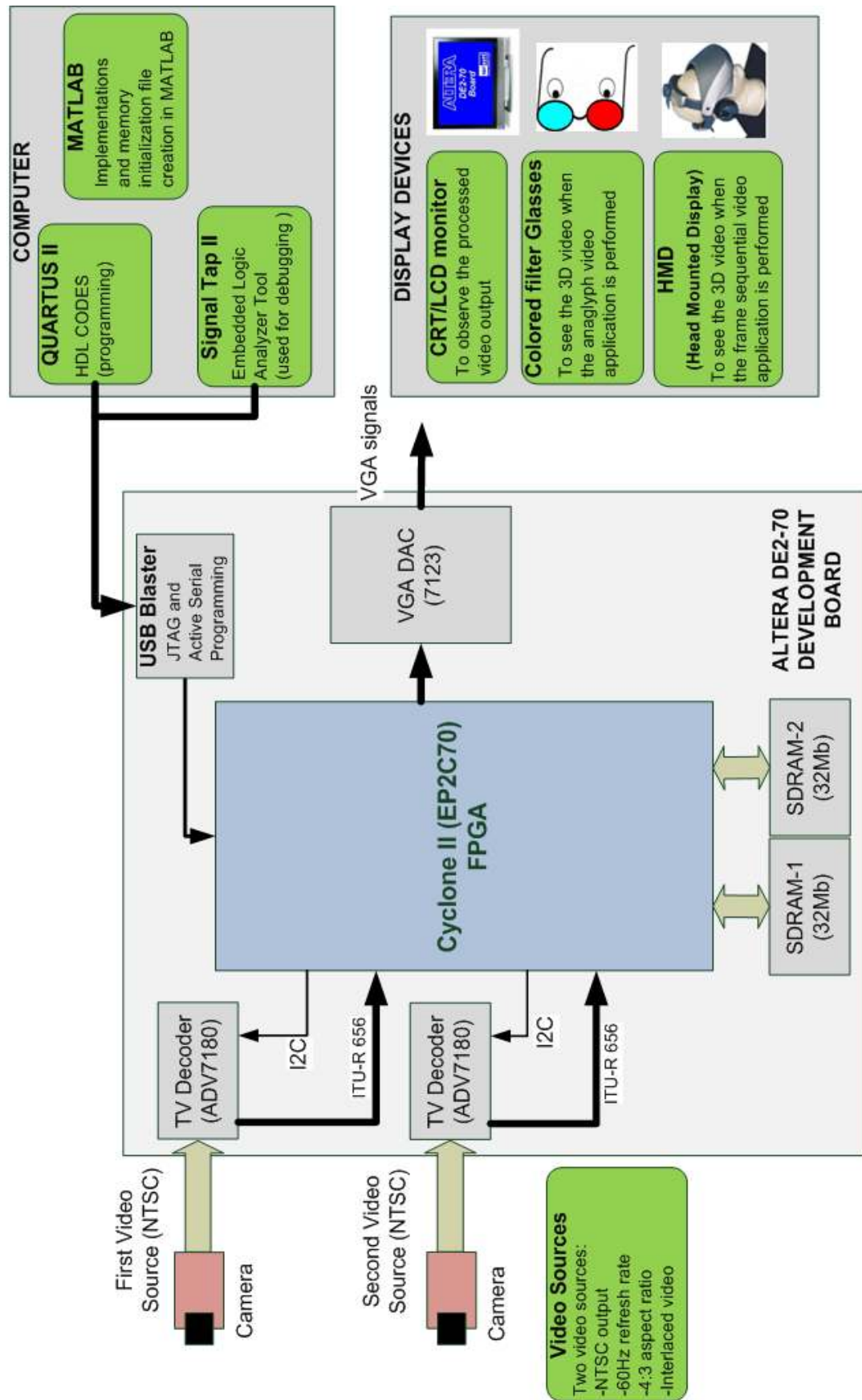


Figure 1.5 General block diagram of the project

1.5 Outline of the Thesis

This thesis consists of seven chapters. Chapter Two includes the details of the analog video and digital video. De-interlacing techniques, color space conversions, ITU-R BT.656 digital video standard, VGA interface are the main subjects mentioned in this chapter. Chapter Three gives information about the parallel video applications which are frame sequential video, field sequential video, anaglyph stereo video, chroma keying and picture in picture. In Chapter Four, the history of programmable logic devices and FPGA is introduced. Then, information about the FPGA development board and the used design software is given. In Chapter Five, the implementation of the main video blocks for a single video source is explained in details one by one. In Chapter Six, the implementations of the parallel video applications are explained. The obtained results are presented. Finally, in Chapter Seven, the conclusion is given, and possible future studies are discussed.

CHAPTER TWO

BACKGROUND OF VIDEO

2.1 Introduction

In this thesis, one of the main steps is to display an analog video input on a VGA monitor by using a FPGA board. In order to process video signals, analog video format should be converted to digital video format.

In this chapter, background information about the analog video and digital video is given. As seen from the Figure 2.1, an analog video source basically passes through the following blocks.

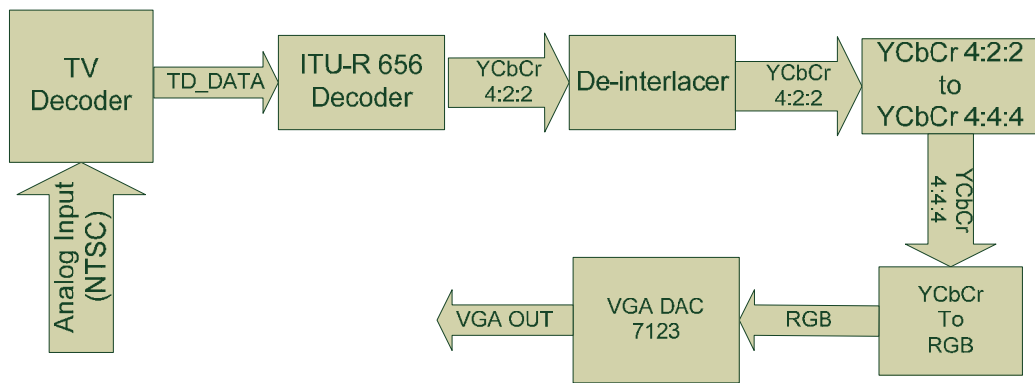


Figure 2.1 Block diagram of the Analog Video to digital VGA output conversion

2.2 Analog Video

An analog video signal contains luminance (brightness) and the chrominance (color) information. Composite video (CVBS), s-video and component video are the analog video signals. In component video and s-video, luminance and chrominance information is carried in separate signals. On the other hand, in composite video signal, the luminance(Y) and chrominance(C) components are combined into a single line level signal. In this work, composite video is in our interest. NTSC, PAL and

SECAM are the standard broadcast video formats (Jack, 2007). The following subsections give information about the NTSC and PAL video formats.

2.2.1 NTSC Composite Video Interface

NTSC (National Television System Committee) analog composite baseband video signal has a refresh rate of 30 full-frames per second. The refresh rate can be thought as 60 half-frame per second because NTSC has an interlaced video format. In interlaced video, each full frame is composed of two fields; odd fields and even fields. Odd fields and the even fields are displayed in sequence. Each field has 262.5 lines. Therefore odd field and even field lines make up a 525 lines per frame. 486 of these lines are visible lines. Figure 2.2 shows the relation between the fields and frames (Jack, 2007).

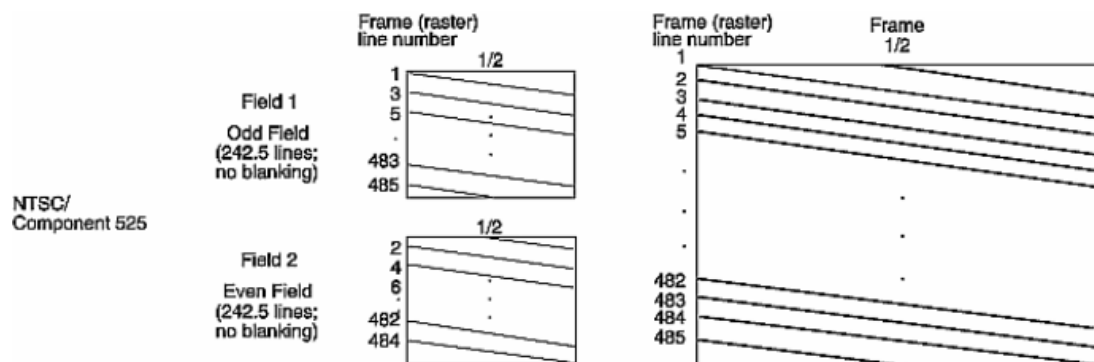


Figure 2.1 Fields and Frame (NTSC)

2.2.2 PAL Composite Video Interface

PAL (Phase Alternating Line) is an analog composite video signal. PAL video has a refresh rate of 50 half-frames per second. Each full frame contains 625 horizontal lines. 576 of these lines are visible lines.

2.3 Digital Video

RGB and YCbCr are the most common digital video signals. RGB is the digitized version of the analog RGB video signals. YCbCr is simply the digitized version of the analog YPbPr video signals (Jack, 2007).

A composite analog video signal can be converted to a digital video stream via an analog to digital converter (ADC). A video decoder IC performs this conversion and some analog signal preprocessing operations. Analog baseband video television signals (NTSC, PAL, and SECAM) are converted to digital formats to be processed and displayed. ITU-R BT.656 describes a simple protocol to transmit digital video for the uncompressed PAL, SECAM, or NTSC television signals. ITU-R BT.601 standard describes the fundamentals of the video digitization process and ITU-R BT.656 standard defines the parallel and serial interfaces for transmitting the 4:2:2 YCbCr digital video.

There are several YCbCr sampling formats such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0. The 4:2:2 component video means that, luma component (Y) is digitized at every pixel position (13.5 MHz sample rate) and chroma components (Cb, Cr) are subsampled by a factor 2. This operation is called as Chroma subsampling. Because of the storage and bandwidth limitations, compression and optimization are necessary for video signals. Chroma subsampling method reduces the required bandwidth. The size of subsampled 16-bit 4:2:2 YCbCr data is 2/3 the 24-bit 4:4:4 YCbCr data. Human cannot perceive the visual difference between the original and subsampled image because the human eye is less sensitive to variations in color than intensity (Digital Creation Labs Incorporated, 2004). Chroma subsampling process is illustrated in Figure 2.3.

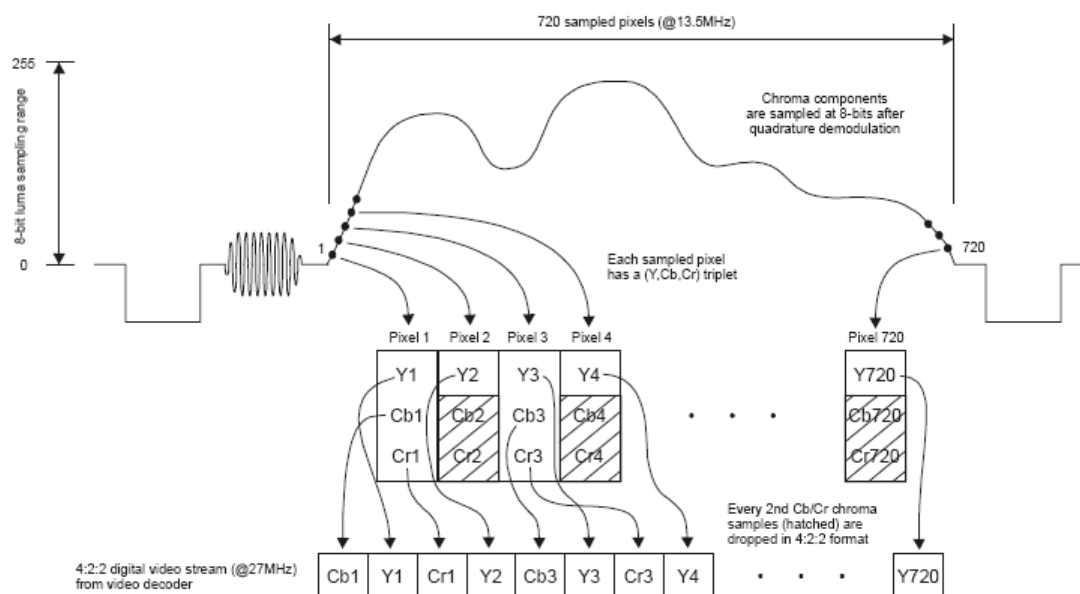


Figure 2.2 ITU-R BT.656 4:2:2 digital video stream

In ITU-R BT.656 standard, each line of video is sampled at 13.5 MHz, it corresponds to 720 active samples per line. 4:2:2 YCbCr data has 720 active samples for Y and, 360 active samples for each Cb and Cr per line. Because of the multiplexing Y data and CbCr data, ITU-R BT656 data stream rate should be at 27MHz (Intersil Americas Inc, 2002).

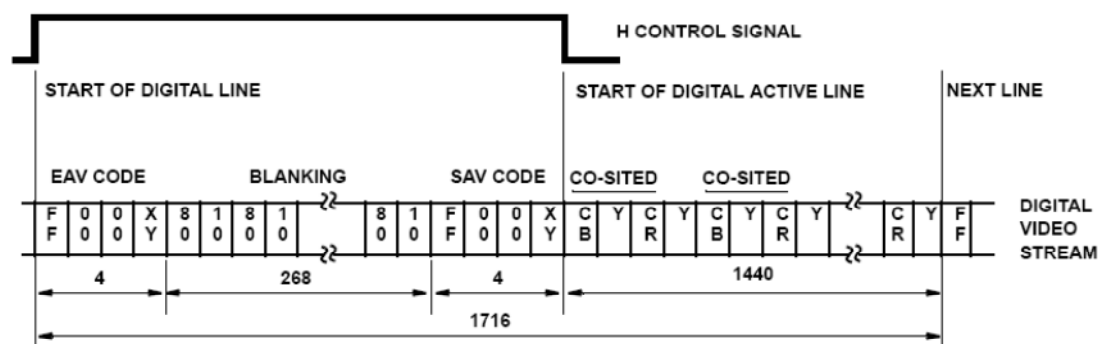


Figure 2.3 BT.656 8-bit Parallel Interface Data Format for NTSC (525/60) Video

2.4 Subsampled YCbCr Formats

Chroma subsampling is widely used technique to reduce the bandwidth. In this technique, resolution of the color information is reduced, because the human eye is more sensitive to the intensity than color differences. There are several YCbCr subsampling formats such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0. In the following subsections the details of the subsampled YCbCr formats are given (Jack, 2007).

In order to display the subsampled YCbCr data (4:2:2, 4:1:1, 4:2:0), it is first converted to uncompressed YCbCr 4:4:4 data. Interpolation method is used to generate the missing Cb and Cr values.

2.4.1 4:4:4 YCbCr Format

Each sample has Y, Cb and Cr value. Each value is typically 8 bit. Therefore, 24 bit is required for each sample. Positioning of the Y, Cb, Cr samples for the 4:4:4 format is illustrated in Figure 2.5.

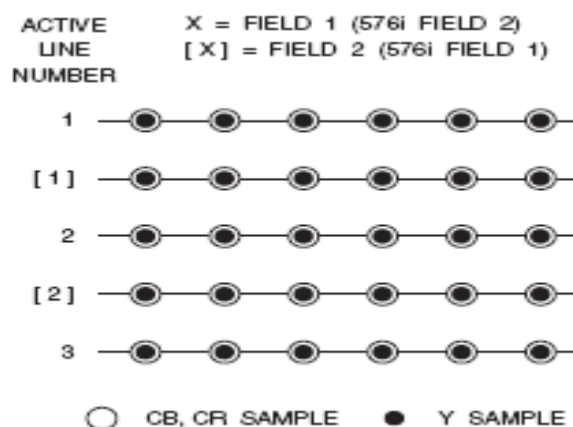


Figure 2.4 4:4:4 Co-Sited Sampling. The sampling positions on the active scan lines of an interlaced picture.

Also 4:2:0 YCbCr format is commonly used for video compression. In addition to 4:2:2 sampling format, reduction of Cb and Cr is performed on both horizontal and vertical directions.

2.5 De-interlacing Techniques

In order to display an interlaced video signal on a non-interlaced (progressive) display device, de-interlacing conversion is required. Interlaced video sequence consists of even fields and odd fields. Even field contains the signal of even-numbered lines of a frame while the odd field contains the odd lines. For a video sequence two fields alternate in time. Field combination and field extension are two basic de-interlacing techniques (Hsiao & Jeng).

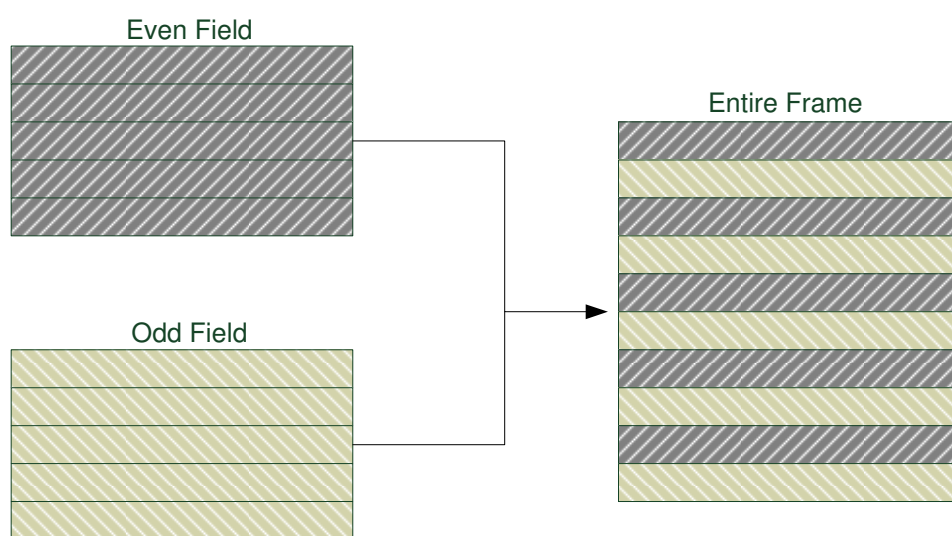


Figure 2.7 The relationship between the odd field even field and the entire frame

2.5.1 Weave De-Interlacing- Field Combination

Weaving is the simplest combination de-interlacing method which combines the odd field and even field into a frame. This method causes line crawling effect in the videos with motion (Hsiao & Jeng).

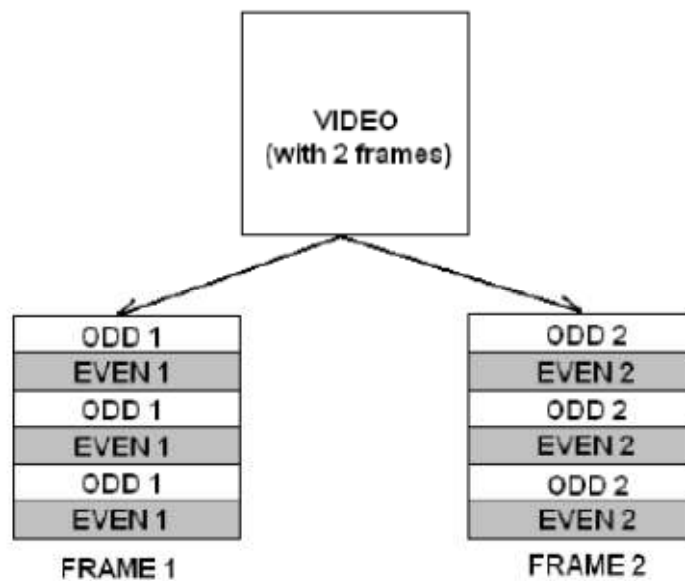


Figure 2.8 Weave De-Interlacing

2.5.2 Bob De-Interlacing (Line Doubling) - Field Extension

Bob is an extension technique. The odd lines or even lines are duplicated to form an entire frame. This technique prevents the line crawling effects but it causes a reduction in picture quality since the vertical resolution is halved.

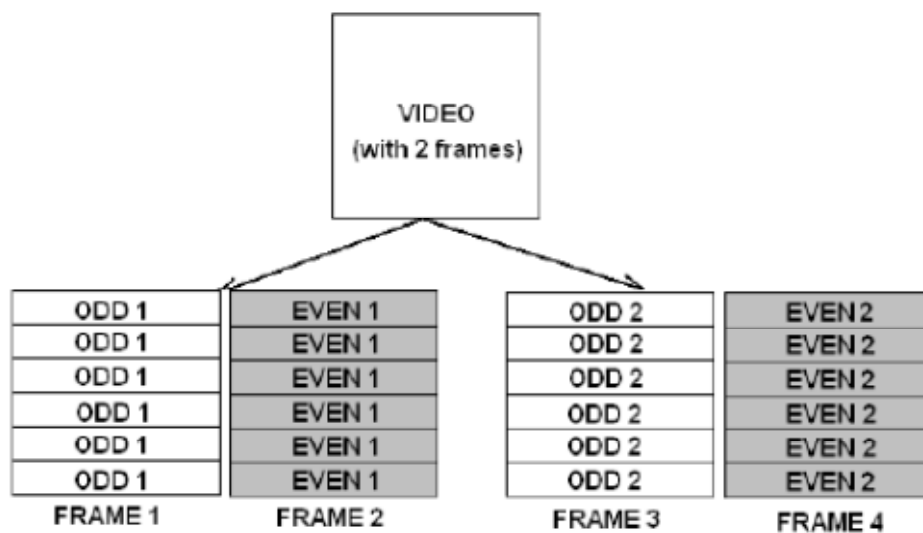


Figure 2.9 Bob (Line Doubling) De-Interlacing

2.5.3 Scan Line Interpolation De-Interlacing - Field Extension

Linear interpolation is the most commonly used method for de-interlacing. The empty odd field or even field lines are interpolated by averaging the upper and lower line of the existing field.

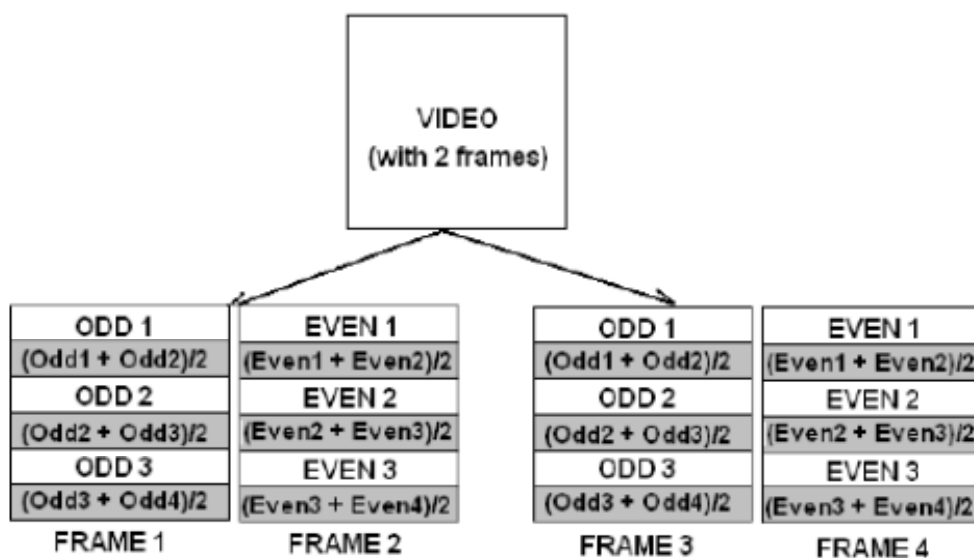


Figure 2.10 Scan line Interpolation De-Interlacing

2.6 Color Spaces

Color space is the mathematical representation of a set of color. RGB (Red, Green, Blue) color space is used for computer graphics; YIQ, YUV, or YCbCr are used in video systems; and CMYK is used in color printing. RGB color space is the best choice for the computer graphics, because color displays use red, green and blue to create the desired color. However, processing an image in the RGB color space is not an efficient method. Working in the YCbCr color space simplifies the implementation of brightness, saturation and hue controls. On the other hand, in the RGB color space, three components of the RGB should be considered together to modify the intensity or to change the color of any pixel. Calculations for the intensity or color are made, modifications are performed and the processed RGB values are calculated again. Therefore in many video standards, YUV, YIQ, and YCbCr color

spaces are used where the luma and two color difference signals exist separately (Jack, 2007).

The color information of the composite video is decoded into the YCbCr color space as a result of ITU-R BT656 standard. YCbCr color space can be useful for many image processing applications. However, in order to display the images on a computer, a color space conversion from RGB to YCbCr is required.

2.7 YCbCr to RGB Color Space Transformation

YCbCr is a scaled and offset version of the YUV color space. Y is defined to have a nominal range of 16-235; Cb and Cr are defined to have a nominal range of 16-240. If the RGB data has a range of 0-255 for each component, the following equations are convenient to use. Notice that, after the transformation operation, RGB data should be saturated at the 0 and 255 levels to avoid underflow and overflow problems (Jack, 2007).

$$R = 1.164(Y - 16) + 1.596(Cr - 128) \quad (2-1)$$

$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \quad (2-2)$$

$$B = 1.164(Y - 16) + 2.018(Cb - 128) \quad (2-3)$$

2.8 VGA Interface

VGA interface consists of five controlling signals. These are Red, Green, Blue color signals and vertical and horizontal synchronization signals. The Red, Green and Blue signals transmit the color information. The Vertical and horizontal sync signals help the display synchronization with the video stream. Horizontal and vertical synchronization signals are used to define the ends of each line and frame.

Figure 2.12 basically shows the timing of the horizontal synchronization signal for each row. Each horizontal sync pulse indicates the end of one row and the start of next row (time a interval in the figure). During the time period before and after the h_{sync} pulse RGB data inputs should be zero. These time periods are called as *back*

porch (time *b*) and *front porch* (time *d*). The rest of the time is for the displayed interval (time *c*). Vertical timing specification has also the same characteristics with the horizontal timing. Each vertical synchronization pulse indicates the end of one frame and the start of a new frame.

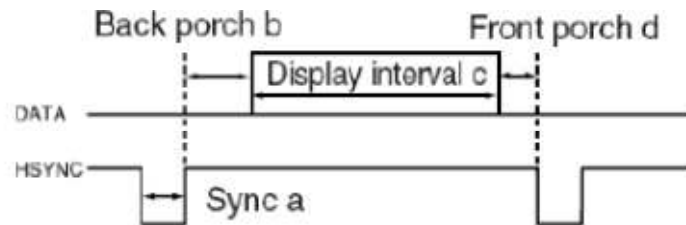


Figure 2.11 VGA Horizontal timing specification

There are lots of VGA standards. Table 2.1 and Table 2.2 show the horizontal and vertical timings defined for different video resolutions.

Table 2-1 VGA horizontal timing specifications (DE2-70 user Manual)

VGA Mode		Horizontal Timings				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(Mhz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25 (640/c)
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36 (640/c)
SVGA(60Hz)	800x600	3.2	2.2	20	1	40 (800/c)
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49 (800/c)
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56 (800/c)
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65 (1024/c)
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75 (1024/c)
XGA(85Hz)	1024x768	1	2.2	10.8	0.5	95 (1024/c)

Table 2-2 VGA vertical timing specifications (DE2-70 user Manual)

VGA Mode		Vertical Timings			
Configuration	Resolution(HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10
VGA(85Hz)	640x480	3	25	480	1
SVGA(60Hz)	800x600	4	23	600	1
SVGA(75Hz)	800x600	3	21	600	1
SVGA(85Hz)	800x600	3	27	600	1
XGA(60Hz)	1024x768	6	29	768	3
XGA(70Hz)	1024x768	6	29	768	3
XGA(85Hz)	1024x768	3	36	768	1

Standard VGA resolution is 640x480. VGA active video has 480 lines and each line contains 640 pixels. That is the active video portion which is the visible part of the video displayed on the screen. The total size of a video frame can be thought wider and taller than the active video portion. Figure 2.13 shows the geometry for a complete VGA frame (800x525).

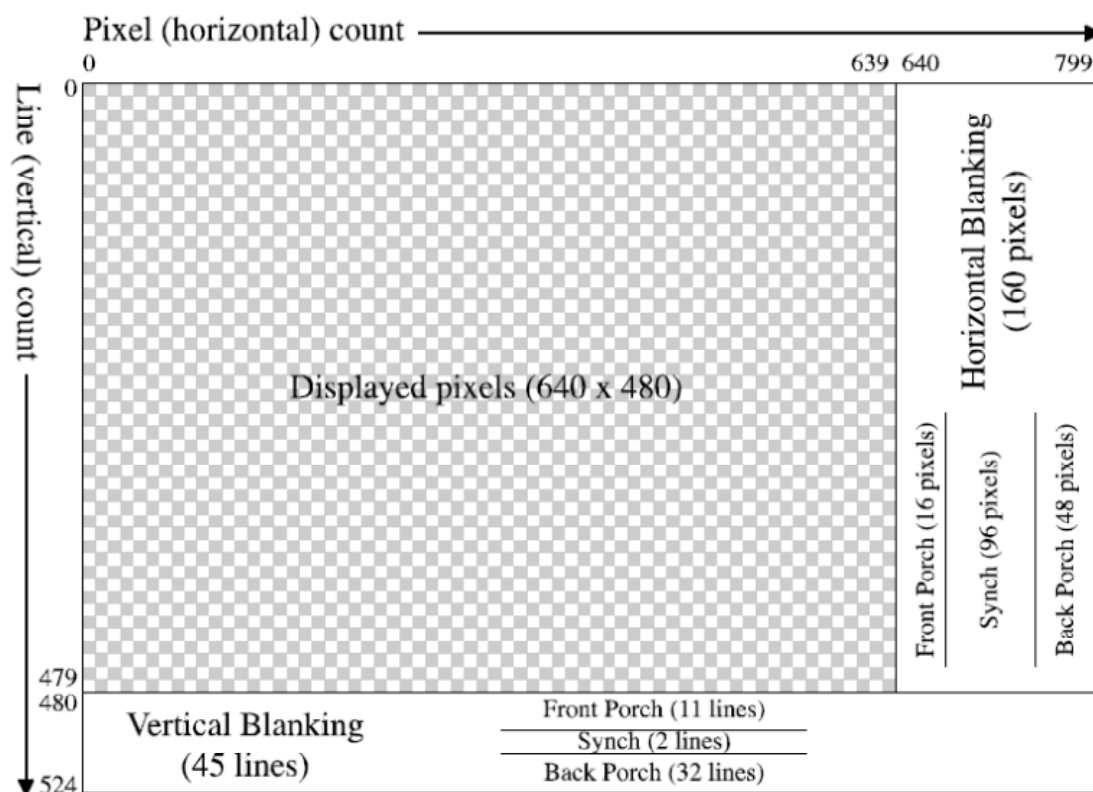


Figure 2.12 Geometry for a complete VGA frame

CHAPTER THREE

PARALLEL VIDEO APPLICATIONS

3.1 Introduction

In this thesis, parallel video editing applications are implemented in a real time system by using FPGA. This chapter gives information about the video applications. Frame sequential video, field sequential video, anaglyph stereo video, chroma keying and picture in picture are the main real time video applications implemented in this thesis.

3.2 Frame Sequential Video

Frame sequential video consists of a sequence of alternating frames for the left eye and right eye. Figure 3.1 illustrates the frame sequential video method. As seen from the figure left eye and right eye images are sent sequentially in time. In other words, while one frame is from the left eye image, the next frame is from the right eye image. The display devices that accept this video stream, separates the frame sequential images. In this way, the segmented left eye images are sent to the left eye of the user, and the likewise for the right eye images. When the individual images are being viewed by each eye, our brain automatically puts these images together and interprets information regarding the third dimension. The disadvantage of this technique is that, visual information sent to the user is halved. For example, a frame sequential video which has a 60Hz refresh rates, has information both the left and the right eye images. After the segmentation of the left eye and right eye images, half of the images go to left eye and the other half go to the left eye. Then, each eye gets the visual image updates at 30Hz. Halving the 60Hz to 30Hz can results noticeable flicker artifacts which may bother some users. To solve this problem, frame sequential devices should work at higher input frame rates. 120Hz is commonly proposed for the frame sequential display (WorldViz LLC).

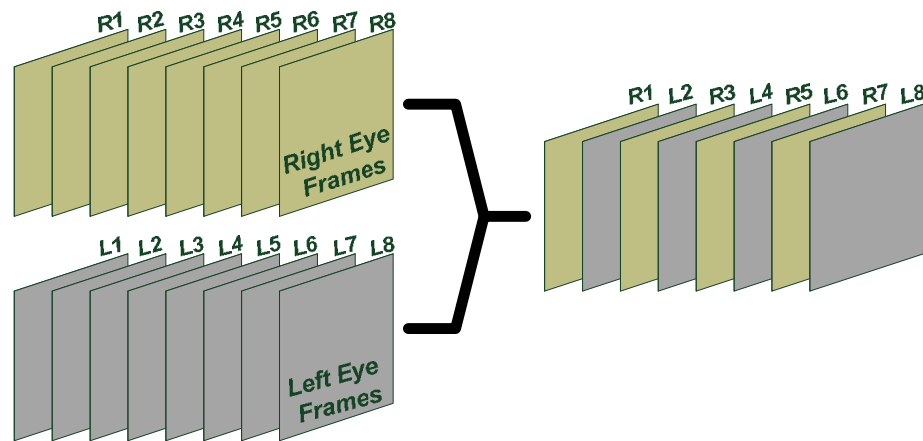


Figure 3.1 Frame sequential video

Frame sequential video can be used for the stereoscopic devices that accept single input for both left and right eye images. For the frame sequential video applications, HMD (Head Mounted Display) devices or active-shutter glasses can be used. These devices are synchronized and allow the correct eye to view the correct view at the precise time.

In one part of this thesis, *Head Mounted Display* 5DT HMD 800-26 is used to see the frame sequential video results on a 3D display device which accepts frame sequential video input. Figure 3.2 shows the 5DT HMD 800-26. HMD's are often used for research and development, virtual reality applications, computer graphics applications, 3D CAD systems and personal entertainments.



Figure 3.2 5DT HMD 800-26

5DT HMD 800-26 accepts both VGA (640x480) and SVGA (800x600) inputs where the VGA output of our design will be connected to this input. Refresh rate can be 60Hz or higher rates. For the 3D mode higher rates are preferable. 5DT HMD 800-26 supports two types of 3D mode. One of them is DDC line protocol used by the nVIDIA Stereo drivers. The other type is frame sequential stereo. In this mode the user can switch the images, if the image on headset is in reverse stereo (HMD User Manual, 2004).

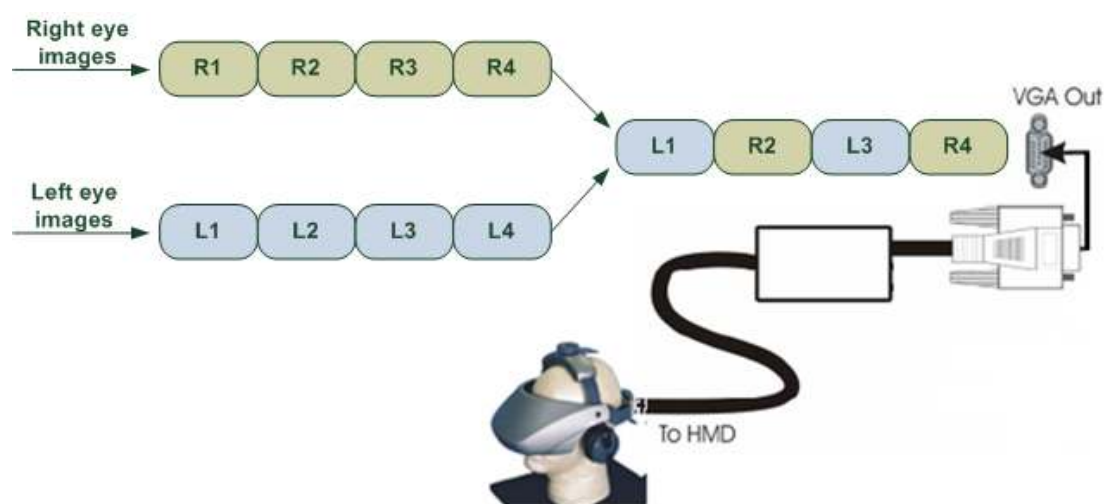


Figure 3.3 Frame sequential 3D system with HMD

3.3 Field Sequential Stereoscopic Video

Field sequential method is an old standard for the stereo video and works only on TV screens that use cathode ray tubes (CRT) for the display. As mentioned on previous chapters NTSC/PAL video is formed by an interlaced series of fields (odd fields and even fields). The screen is scanned twice, once for the odd numbered lines and again for the even numbered lines. 3D images have to be separated into left and right images. Odd fields and even fields can be separated for a 3D image. One field is used for the left eye and the other for the right eye. Since the video comprises two fields, stereo video can be transmitted without doubling the bandwidth. The disadvantage of this method is that, the vertical resolution is halved for each eye because of the field separation (StereoGraphics Corporation, 1997). Figure 3.4

illustrates the field sequential video. As seen from the figure, the odd field of the first video and the even field of the second video are combined for the field sequential video.

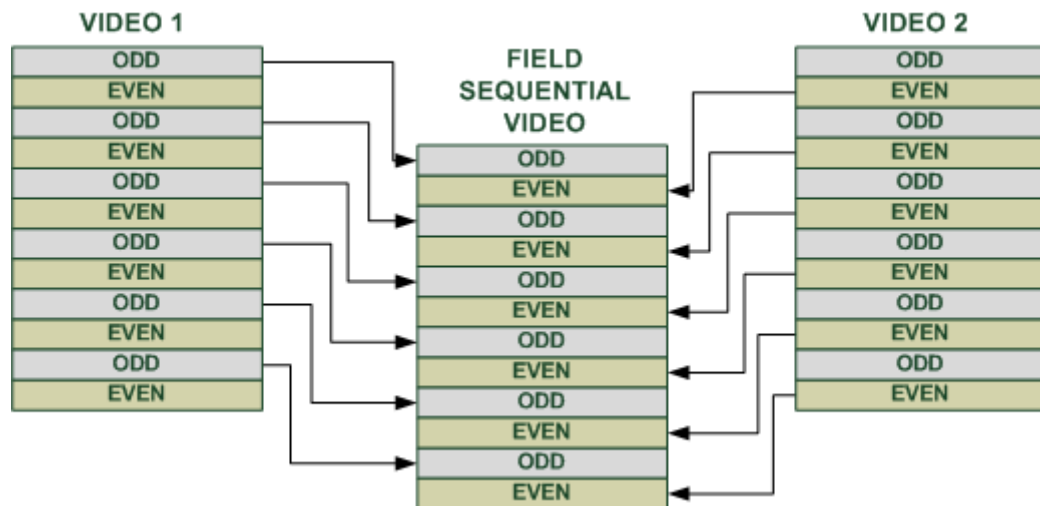


Figure 3.4 Field sequential video

3.4 Anaglyph Stereo

Anaglyph technique was first developed in 1853 by Wilhelm Rollman, a German, in Leipzig (Best 3DTVs). Anaglyph is a method to see stereoscopic images using color spectacles. Although other stereoscopic visualization methods such as polarized or shuttered glasses can give better performance, the anaglyph method is the only way that stereoscopic images can be viewed on ordinary television sets or computer screens without any special hardware, only inexpensive colored glasses (Dubois).

Anaglyph method is very old, but the techniques used to generate the anaglyph images are very empirical. Two images from the perspective of the left eye and right eye projected together as a single image. Generally, red component of the left eye image and green-blue (cyan) components of the right eye image are combined for 3D viewing. In this way, each pixel contains information for the left and right views.

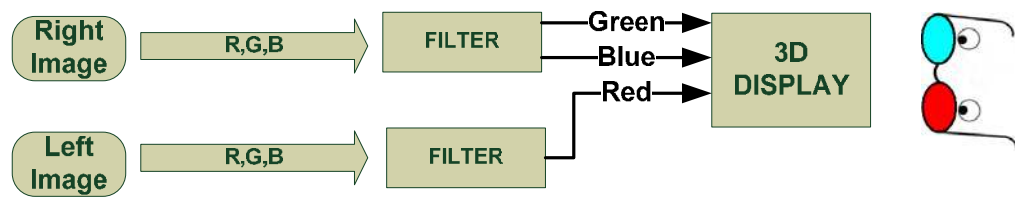


Figure 3.5 Anaglyph image transmission

The superimposed images look blurred because it contains two views for the left eye and right eye in one image. These two views have to be separated correctly for the visualization of the 3D effect. Anaglyph image can be viewed with special anaglyph glasses which have different colors for each lens (Red/Cyan, Red/Blue or Yellow/Blue). A colored lens allows one of the views to pass through the lens to the eye and excludes the other view. For red/cyan glasses, red colored lens allows the red component of the image to pass, and blocks the green and blue components. Cyan colored lens (combination of the green and blue) allows the green and blue components to pass and blocks the red component of the image. In this way, two different images can be viewed by each eye and human visual system can merge these stereo views to create the impression of 3D effect.

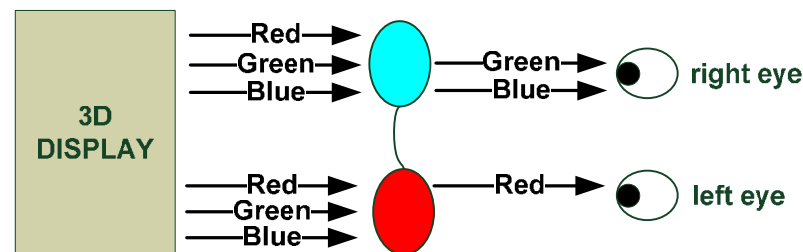


Figure 3.6 Anaglyph image viewing

As mentioned above for anaglyph images the transmitted color and used filters in glasses are important. Chromatically opposite colors are chosen for each eye. Red, green and blue are the primary colors and these colors cannot be created by any combination of the other colors. The secondary colors are cyan, magenta and yellow. Figure 3.7 shows the relation between the primary colors. For example, red and cyan color glasses have chromatically opposite colors. As seen from the figure, red is the primary color and cyan is the combination of the green and blue. There is no

intersection between the color sets red and cyan (Color Vision: One of Nature's Wonders).

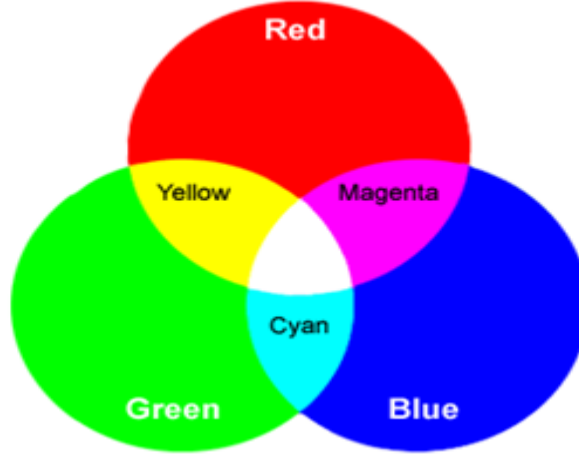


Figure 3.7 Primary additive color combinations

The main problem for the anaglyph images occurs when the green image is viewed through the red filter. As a result of this, instead of single image, mixture of the different images (red and green) is seen by one eye. The observer is affected from this cross-talk on stereoscopic image (Dumbreck & Smith, 1991, 1992).

There are several anaglyph methods such as gray anaglyphs, half color anaglyphs, color anaglyphs. For the full color anaglyph all three primary colors have to reach at least one eye. Equation 3.1 below gives the formula of the full color anaglyph method to calculate the RGB values of the anaglyph image (r_a, g_a, b_a) from the RGB values of the left image (r_l, g_l, b_l) and RGB values of the right image (r_2, g_2, b_2) . Note that, the formula has to be applied for each pixel (Wimmer, 2005).

$$\begin{bmatrix} r_a \\ g_a \\ b_a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} r_1 \\ g_1 \\ b_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_2 \\ g_2 \\ b_2 \end{bmatrix} \quad (3-1)$$

As explained in this section, by mixing the video instead of multiplexing fields or frames anaglyph video can be created. The advantage is that, full video signal is used

and no reduction in resolution or in frame rate occurs. However, due to the color tinted glasses the viewed image has poor color fidelity and shades of red and green.



Figure 3.8 Left and right images combined into a resultant stereoscopic view

As seen from the Figure 3.8, the left eye image with red component and the right eye image with the cyan (green + blue) component combined into a stereoscopic view for the anaglyph image.

3.5 Chroma Keying

Chroma keying also known as green screen video editing is a technique that two frames are combined to create a single frame. These two frames can be called as foreground and background. The foreground object and a solid colored backdrop create the foreground image. By using the chroma keying technique, the solid backdrop is removed and the foreground image with a transparent background is overlaid to another background image. In this way, a new image is created which includes the object with a different background. Figure 3.9 shows the chroma keying technique. As seen in figure, the object in front of a green background is photographed and then the foreground areas containing the key color (green) are replaced with the background image.

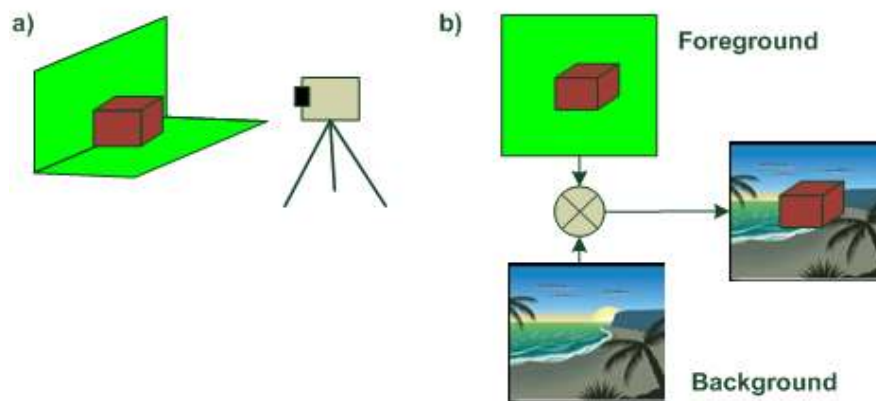


Figure 3.9 Chroma key technique

Chroma key effect is very popular in news casting, weather reporting and virtual environments. Chroma key process uses specific colors which are usually blue or green, because they are the colors most opposite to human skin color. The green backgrounds are more commonly used because the image sensors in the digital cameras are most sensitive to green color and needs less light to illuminate the background. Although the green and blue colors are preferred for chroma keying, any color which is different from the foreground object can be chosen as a key color. Lighting of the background (green/blue wall) is also another important issue for the chroma keying applications. The background must be bright and evenly lit with no hot spot (Wood, n.d.).

In digital video processing YCbCr color space is generally used for the chroma keying. Cb and Cr are used to specify the key color and luminance information can be used to increase the realism of the chroma keying function (Jack, 2007).

3.6 PIP (Picture in Picture)

Picture in Picture (PIP) is also another parallel video application. Picture in picture (PIP) mode is used when the user want to see the two different videos at the same time. One video frame is displayed on the full screen while the other video frame is displayed in a small sub window. Video sound is generally from the main video. Usage areas are generally the televisions. Nowadays PIP becomes available as

a feature of advanced television receivers. In PIP applications one of the videos is chosen as main video and it is displayed on the full screen. The second video is displayed on a small window. Generally it is possible to switch the source of the main video as in Figure 3.10.

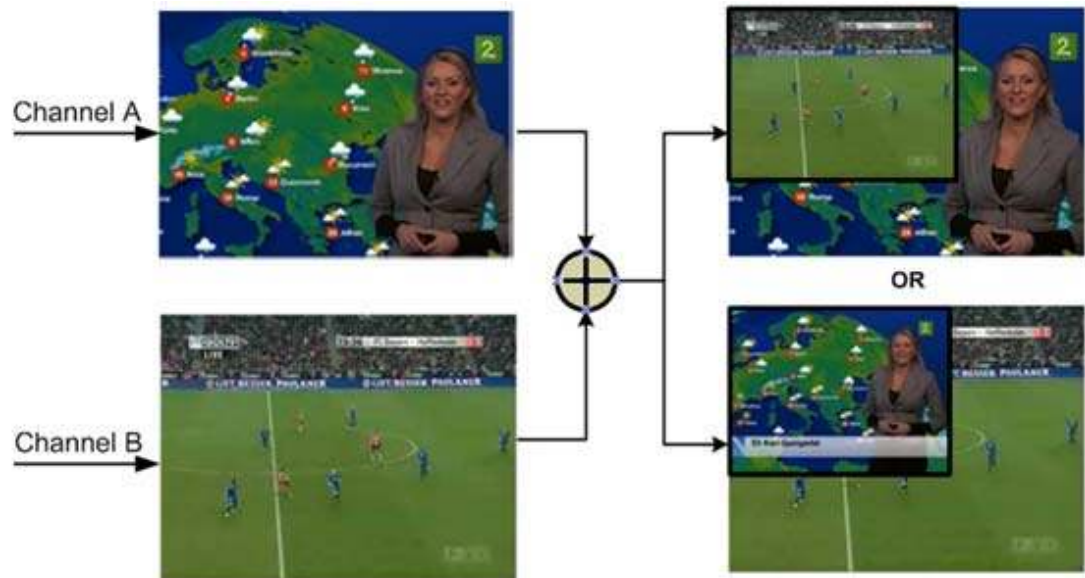


Figure 3.10 Picture in picture application

CHAPTER FOUR

PROGRAMMABLE LOGIC DEVICES

In this section a brief history of the programmable logic devices is given from simple architectures to modern complex architectures. Also FPGA configuration techniques and the Altera DE2-70 development board used in this thesis has been introduced.

4.1 Evolution of the Programmable Logic Devices

The first type of programmable chip that could implement logic circuit was Programmable Read-Only Memories (PROMs). PROMs are memories that can be programmed by the user for a specific pattern. Address lines can be used as logic circuit inputs and data lines as output. A state machine, a simple algorithm or simple combinational logic circuits with limited inputs and outputs can be presented by using this pattern. However, because of the limitations, PROMs have an inefficient architecture for realizing the logic circuits and rarely used in practice for that purpose (Brown & Rose). After the PROMs, in the early 1970s, PLA (Programmable Logic Array) was developed specifically for implementing logic circuits. As seen in Figure 4.1, a PLA device consists of two levels of logic gates: a programmable AND plane, followed by a programmable OR plane. Logic functions in sum-of-products form can easily be implemented using the PLA structure.

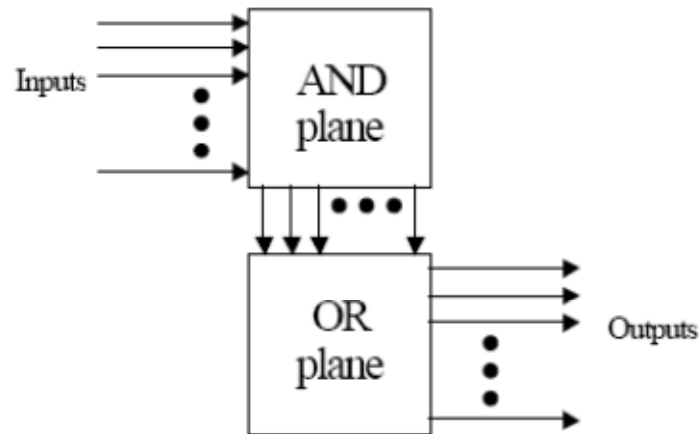


Figure 4.1 PLA architecture (Zeidman)

PLAs were expensive to manufacture and have a poor speed performance. In order to overcome the weaknesses in PLAs, Programmable Array Logics (PALs) were developed. PAL is improved and extremely faster variation of a PLA. It has a programmable AND plane like the PLAs. However the programmable OR plane is fixed to decrease the complexity of the software and the propagation delay time through the device. In addition to the PLAs, other logic devices such as multiplexers, exclusive ORs and latches are added to the inputs and outputs of the PAL device. Most importantly, clocked elements and flip-flops are included. Thus a large number of logic functions including the clocked sequential logic circuits can be implemented by using these logic devices (Zeidman). Figure 4.2 illustrates the PAL architecture.

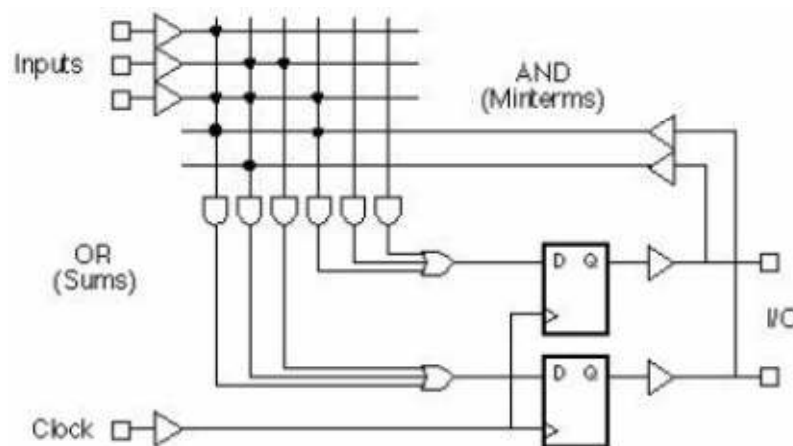


Figure 4.2 PAL architecture (Zeidman)

All these small PLDs (PLAs, PALs etc.) are grouped into a single category called as SPLDs (Simple PLDs). Complex Programmable Logic Devices (CPLDs) can be seen as a continuation of the PLA architecture. CPLDs are the combination of multiple SPLDs which are integrated onto a single chip with programmable connections between the SPLD blocks. Figure 4.3 shows a simple architecture of a CPLD.

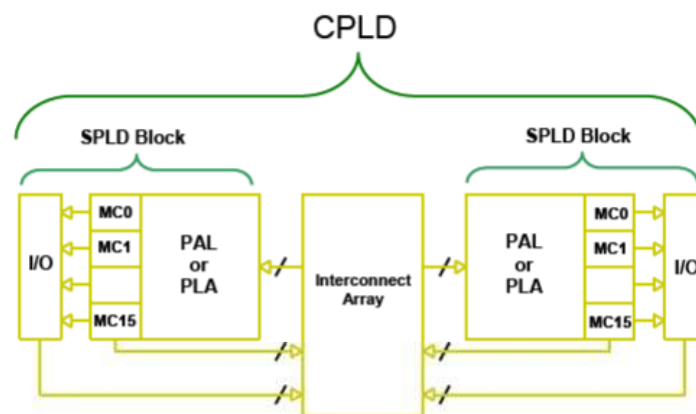


Figure 4.3 CPLD architecture (Xilinx, 2006)

4.2 FPGA Architecture

In 1985, Xilinx introduced Field Programmable Gate Arrays (FPGAs). FPGAs are structured like a gate array ASIC (Application Specific Integrated Circuit). Therefore, today FPGAs can be used in prototyping ASICs or in place where ASIC will be used. FPGAs are called as “field programmable” because it can be configured by the customer and designer after manufacturing. Xilinx and Altera are the most known FPGA manufacturers in the world.

There are two basic types of FPGAs: SRAM-based programmable FPGAs and one time programmable (anti-fuse based) FPGAs (Xilinx, 2006). In SRAM-based FPGAs, the configuration memory is based on volatile SRAM. Therefore, they need to be reprogrammed at each power-up state from a dedicated flash chip.

The FPGA architecture consists of configurable logic blocks, configurable I/O blocks, programmable interconnects and clock circuitry for driving clock signals to each logic block. Also additional logic sources such as ALUs (Arithmetic Logic Units), embedded memory blocks and decoders can be available.

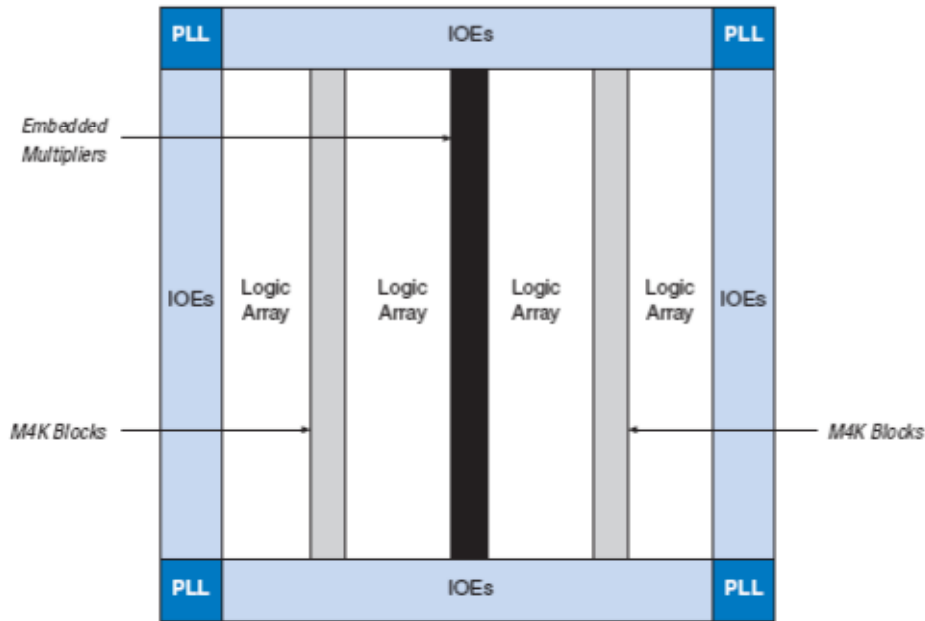


Figure 4.4 Cyclone II EP2C20 device block diagram

The name of the FPGA logic blocks changes depending on the vendors. Logic blocks are called as *Configuration Logic Block* (CLB) for Xilinx and *Logic Array Block* (LAB) for Altera. Since Altera Cyclone II device is used in this thesis, information about the LABs will be given in the following sub-section. Figure 4.4 shows block diagram of Cyclone II FPGA which consists of LABs, M4K memory blocks, I/O blocks, phase-locked loops (PLLs) and embedded multipliers.

4.2.1 Logic Elements

Logic Element (LE) is the smallest unit in Cyclone II architecture which provides advanced features with efficient logic utilization. As seen from the Figure 4.5, each logic element includes a four-input look-up table (LUT) which is a function generator that can implement any function of four variables. Additionally, logic

element includes a programmable register, a carry chain connection and register chain connection (Altera, 2007).

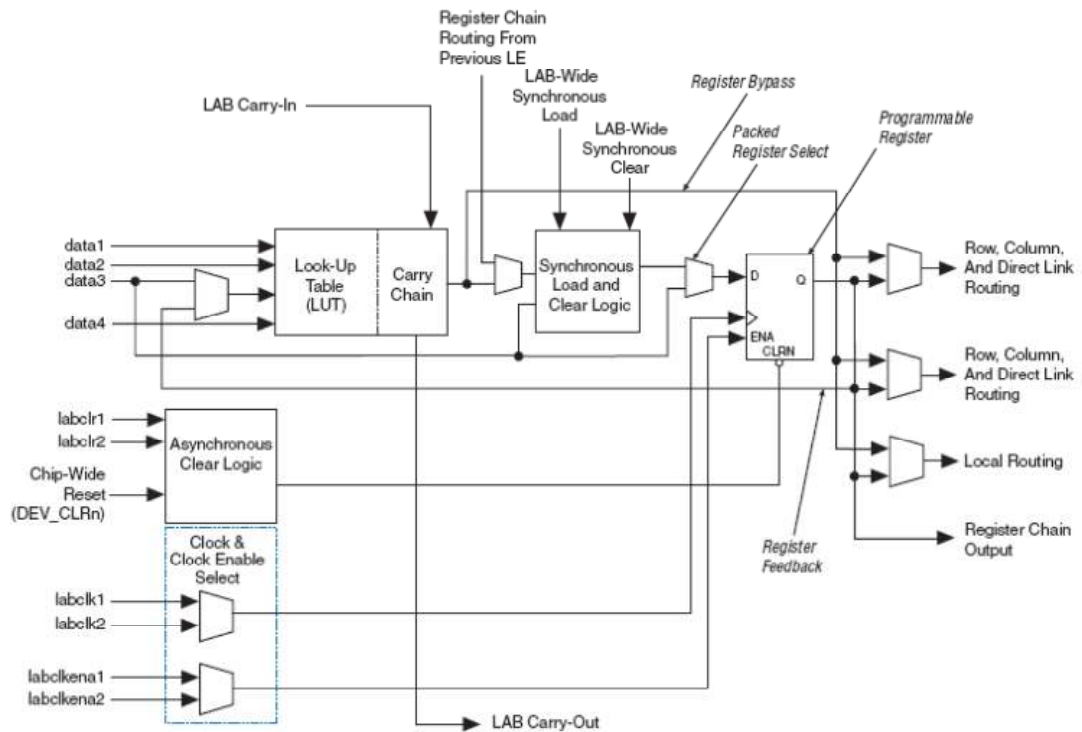


Figure 4.5 Cyclone II LE (Logic Element) block

4.2.2 Logic Array Blocks (LAB)

Logic Array Blocks (LABs) are grouped into rows and columns across the device. Figure 4.6 shows the Cyclone II LAB structure. Each LAB consists of the following: (Altera, 2007)

- 16 LEs
- LAB control signals
- LE carry chains
- Register chains (transfer the output of the one LE's register to the adjacent LE's register within a LAB)
- Local interconnect (transfer signals between LEs in the same LAB)

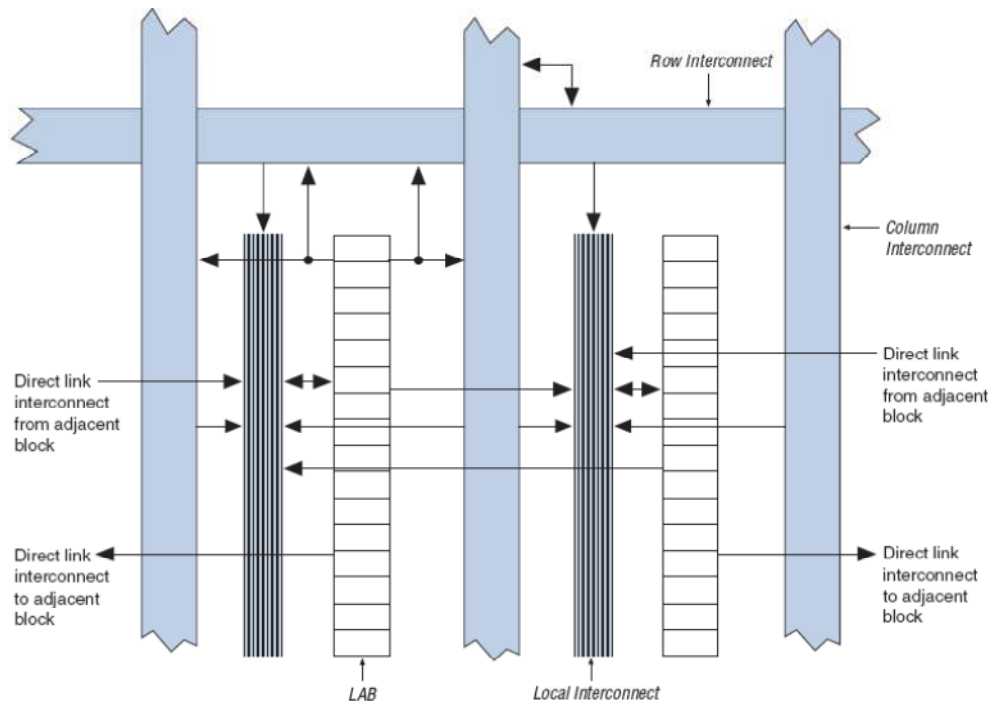


Figure 4.6 Cyclone II LAB structure

4.3 FPGA Design Entry

There are two main methods to implement a hardware design into an FPGA. One of them is schematic design entry and the other is HDL (Hardware Description Language) design entry.

4.3.1 Schematic Design Entry

In the schematic design entry, you draw your design using gates and wires. Schematic design can be a choice for smaller designs. The design can be documented in an easily readable format. However it is not a useful design entry for the large designs and the design file formats are incompatible between the FPGA vendors. Therefore, HDLs are more preferable among the FPGA users because of their portability, flexibility and readability (Zeidman).

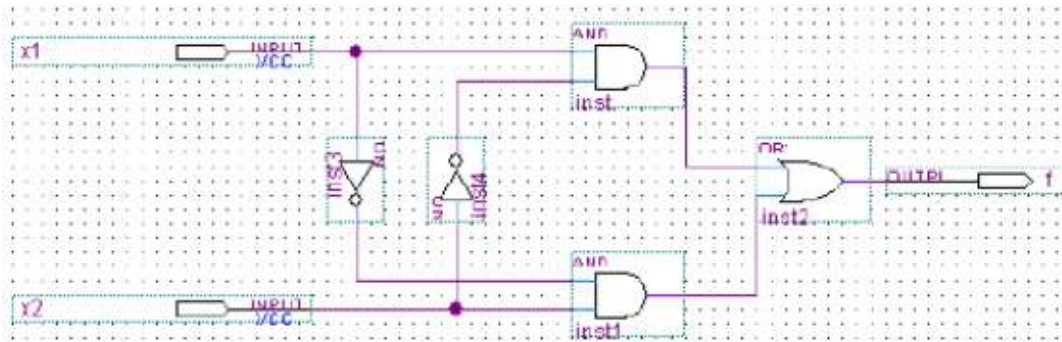


Figure 4.7 Example of schematic design entry

4.3.2 Hardware Description Languages

A more common method is to describe the hardware design using special hardware description languages (HDLs). The two major hardware description languages are VHDL (Very high speed integrated circuit HDL) and Verilog. HDLs are mostly independent from the vendor. Therefore, the HDL code can be easily integrated to another vendor's FPGA after the required changes.

VHDL was originally developed by the US Department of Defense and released in 1985. VHDL became IEEE standard 1076 in 1987 and it was updated in 1993. The Verilog hardware description language has been used extensively since it was launched by Gateway in 1983. Cadence bought Gateway in 1989 and opened Verilog to the public usage in 1990. It became IEEE standard 1364 in 1995. VHDL and Verilog can be used in the same design (Smith D. J.). The syntax constructs of the HDL languages are similar to conventional programming languages. VHDL syntax is similar to Ada programming language and Verilog syntax is similar to C programming language. A specific design model written in VHDL can be used in a design written in Verilog. For example, in this thesis, some of the components are written in VHDL and the rest of them are written in Verilog.

4.4 FPGA Simulation

After the hardware design entry is completed, designers may want to simulate the design on a computer to gain confidence. The required test inputs are given to the design and the output of the signals are observed. There are two ways to create simulation; using an interactive waveform editor or writing a testbench. In interactive waveform editor user enter the shape of the inputs in timescale and the simulator software draws the shape of output. On the other hand, a testbench design is non-synthesizable HDL file that creates stimulus for another synthesizable file. The testbench is usually written in the same behavioral language (VHDL or Verilog) with the design under test.

4.5 FPGA Design Software (Synthesis Tool)

FPGA vendors also provide software programs to configure their devices. These software programs provide a complete design environment and basically can perform four major tasks; design entry, simulation, synthesis and place-and-route, and programming (JTAG). *ISE (Integrated Software Environment)* software is used to develop for Xilinx devices and *Quartus II* software is used to develop for Altera devices.

In this thesis, *Quartus II Web Edition* software is used which is free and a scaled-down version of the full *Quartus II* software. Its functionality is similar to the full version. FPGA board contains a serial EEPROM chip to store the configuration data. When the board's power is turned on, the configuration bit stream in EEPROM is loaded into the FPGA. It is possible to change the non volatile configuration data in EEPROM by using the *Quartus II* software. Also the FPGA can be programmed directly at any time while the board is in power on state.

4.6 FPGA Development Board

FPGA vendors provide development and education boards which includes FPGA and peripheral devices on it. These boards allow the user to implement a wide range of designed circuits. Generally input and output user interfaces (switches, buttons, LCD screen, video input, video output, etc) exist in these designed boards.

In this thesis, Altera DE2-70 development board is chosen because it is sufficient for the video implementations. The main reason while choosing this board was that there are two composite video inputs which are required for our parallel video applications. This section demonstrates the Altera FPGA development board used in this thesis. The following list shows the provided hardware on the Altera DE2-70 board (Terasic, 2009):

- Altera Cyclone® II 2C70 FPGA device
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 2-Mbyte SSRAM
- Two 32-Mbyte SDRAM
- 8-Mbyte Flash memory
- SD Card socket
- 4 pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 28.63-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- 2 TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors

- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- 1 SMA connector
- Two 40-pin Expansion Headers with diode protection

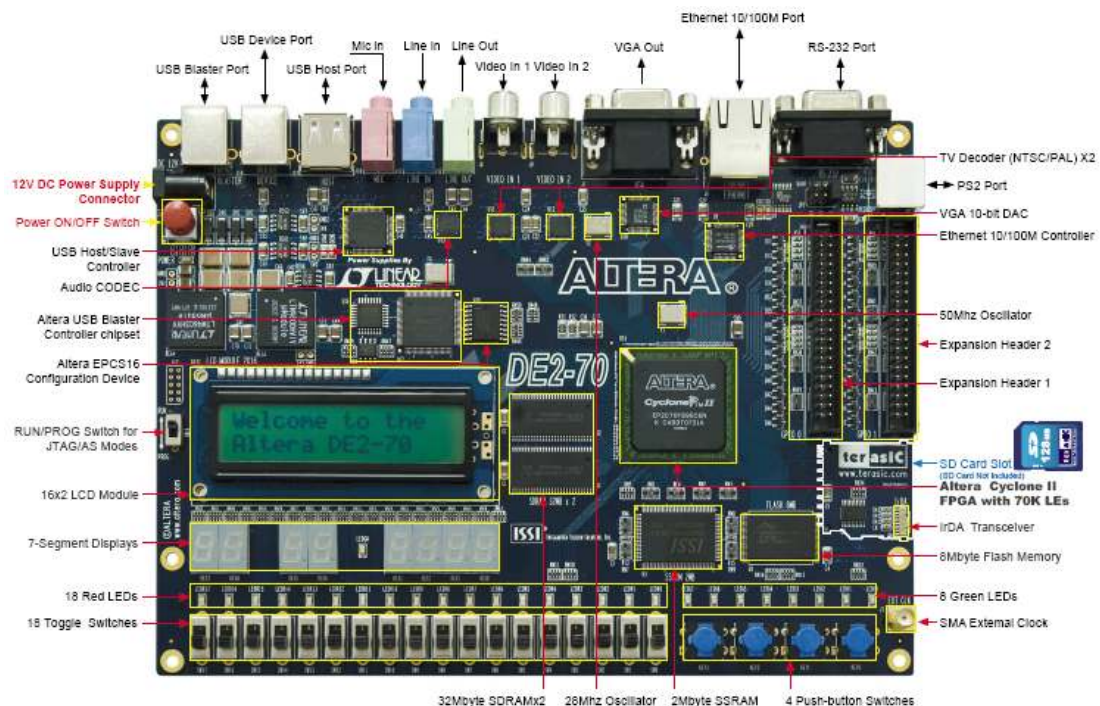


Figure 4.7 A photograph of the DE2-70 Board

The following list shows the feature for the Cyclone II 2C70 FPGA device:

- 68,416 LEs
- 250 M4K RAM blocks
- 1,152,000 total RAM bits
- 150 embedded multipliers
- 4 PLLs
- 622 user I/O pins
- FineLine BGA 896-pin package

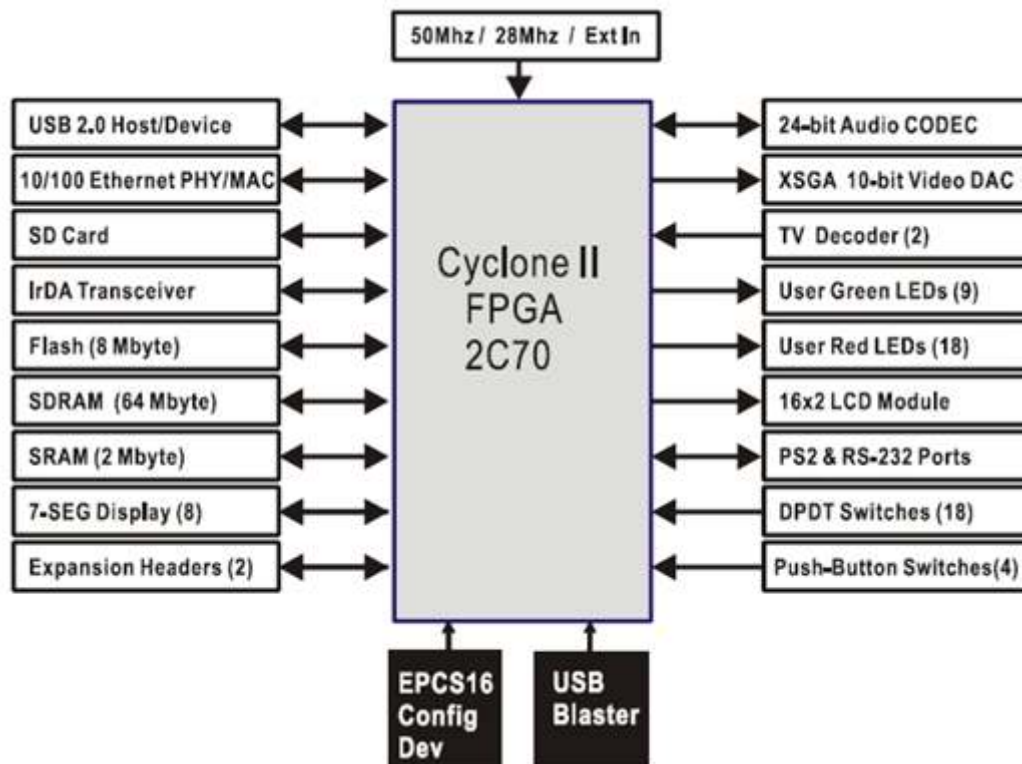


Figure 4.8 Block diagram of the DE2-70 Board

4.7 Signal Tap II Logic Analyzer

It is difficult to add a connection to a pin during the debugging process for the advanced packages (BGA) after a board is designed and manufactured. Cyclone II device supports Signal Tap II embedded logic analyzer tool which monitors design operation over a period of time through the JTAG circuitry. In this thesis, Signal Tap II software which is included with the Quartus II subscription is used to analyze the incomprehensible problems in the design. This system level debugging tool captures and displays signals in designed circuits while the design is running at full speed on the FPGA device. The tool does not require external probes or changes to the design files to capture the state of the internal nodes or I/O pins in the design. Also user can set the triggering options of the logic analyzer to capture the signals during a specific time.

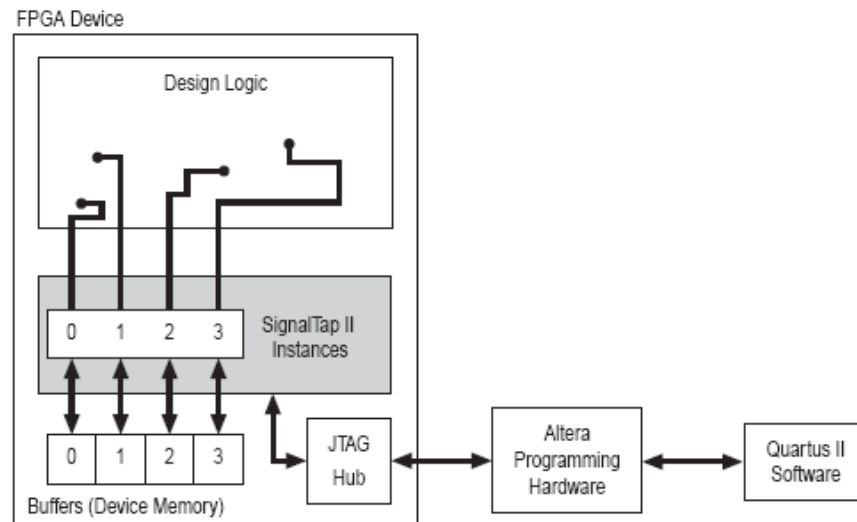


Figure 4.9 SignalTap II Logic Analyzer Block Diagram

4.8 MegaWizard Plug-In Manager

This program helps you create or modify the design files that contain custom variations of megafunctions. Quartus II software contains this program. In this work, division and multiplication operations, SDRAM FIFO read/write operations are created using this tool.

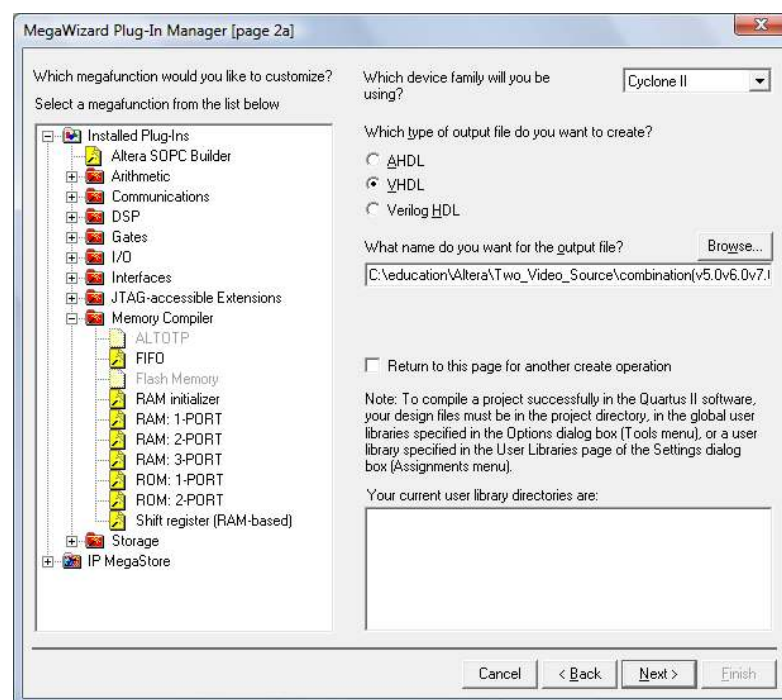


Figure 4.10 Appearance of MegaWizard Plug-In Manager

CHAPTER FIVE

IMPLEMENTATION OF MAIN VIDEO BLOCKS

5.1 Introduction

In this thesis, one of the main purposes is to display an analog video input on a VGA monitor by using the Altera DE2-70 board. While performing these implementations, we have utilized from the DE2_70_TV demonstration which is provided for DE2-70 board. The associated files about this demonstration can be found from the DE2-70 System CD-ROM. This chapter gives a brief detail about the implemented design to display one video source on VGA display. The system overview is depicted as in Figure 5.1. In order to process video signals, analog video should be converted to digital format (TV decoder IC). After the digital processing operations, digital video is converted to analog format again for the VGA monitoring (VGA DAC IC).

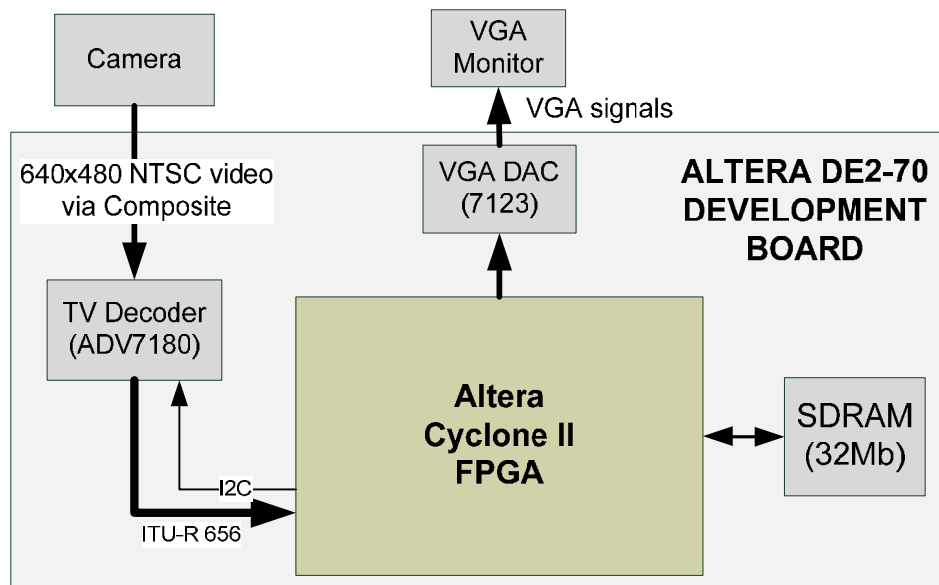


Figure 5.1 System Overview of a single source video system

In the following sections, the main video blocks are defined to convert the analog composite video (NTSC) to the digital VGA output format. There are major blocks in the design called as *ITU-R 656 Decoder*, *SDRAM Frame Buffer*, *YUV422_to_YUV444*, *YCbCr_to_RGB* and *VGA Controller*. Also there is *I2C_AV_Config* block which communicates with TV decoder chip via I²C protocol.

The major blocks are depicted as in Figure 5.2. The digital video stream ITU-R BT656 4:2:2 is sent from the TV Decoder IC (ADV7180) to the *ITU-R 656 Decoder* block. *ITU-R 656 Decoder* block extracts YCbCr 4:2:2 (YUV 4:2:2) video signals and generates data valid control signal. Since, the output video signals of the *TV Decoder* is interlaced, de-interlacing process is performed. Frame Buffer block and a field selection multiplexer (MUX) is used to perform the de-interlacing process. The *VGA Controller* block generates the required data request and odd/even information signals for the *SDRAM Frame Buffer* and *MUX* blocks. After the de-interlacing process, progressive YCbCr 4:2:2 video format is converted to the YCbCr 4:4:4 (YUV 4:4:4) video format in *YUV422_to_YUV444* block. Finally, *YCbCr_to_RGB* block performs the color space conversion between the YCbCr and RGB video formats. *VGA Controller* block also generates standard VGA synchronization signals *VGA_HS* and *VGA_VS* to display the video on VGA monitor (Terasic, 2009).

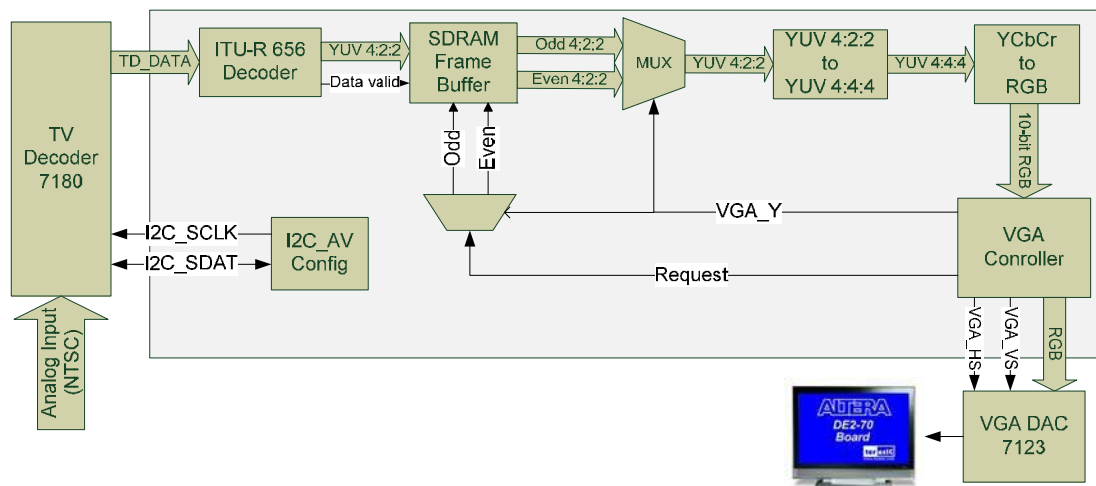


Figure 5.2 Main implemented video blocks

5.2 TV Decoder

Altera DE2-70 board has two ADV7180 SDTV Video decoder chips on it. The ADV7180 automatically detects the standard analog baseband television signals (NTSC, PAL and SECAM standards in the form of composite, S-video and component video) and these video formats are converted into a digital 8 bit ITU-R BT.656 YCbCr 4:2:2 format (ADV7180 datasheet Rev. B).

In this work, NTSC video format is chosen as the video source input. Since video decoder supports NTSC, PAL and SECAM analog video formats, the ADV7180 should be initialized to decode the input video correctly for the NTSC video format. The configuration registers of the ADV7180 can be programmed by a serial I²C bus, which is connected to Cyclone II FPGA as in Figure 5.3.

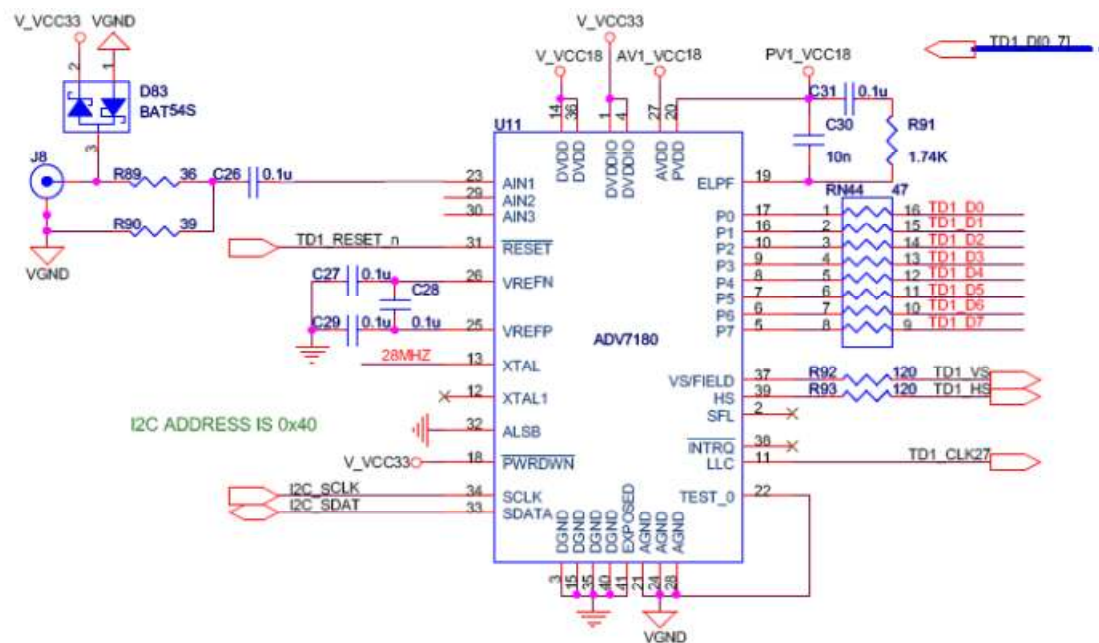


Figure 5.3 TV Decoder schematic

5.2.1 I2C Interface

I²C is a two wire interface, serial data line (SDA) and serial clock line (SCL). SDA transmits data and SCL is used to synchronize all data transfer. In I²C communication, there are one master and multiple slave nodes. Master device drives the clock line and addresses, slave devices receive the clock line and the addresses. Each slave device has a unique address. Both master and slave can transfer data, but the data transfer is controlled by the master device, slave devices can not initiate a data transfer. Before the data transfer, master sends the start sequence. As a result, all connected devices listen to the bus. After the data transfer, a stop sequence is sent by the master device which means that transfer is completed and the bus is available again. Figure 5.4 illustrates the start and stop sequence signal timings.

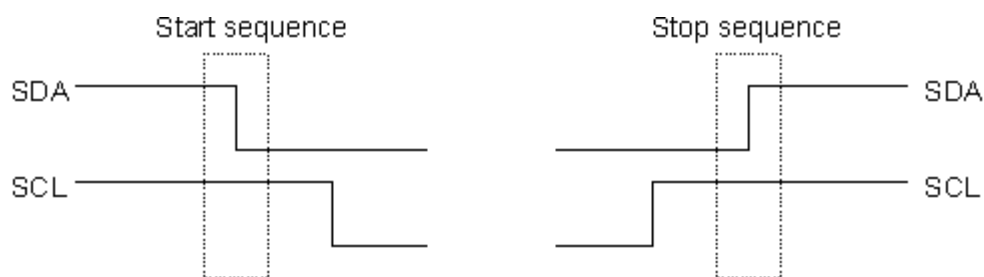


Figure 5.4 Start and stop sequences on the I2C bus

After the start condition, I²C address of the slave device is sent with the R/W bit which indicates the sequence is a write or read sequence. The data length on SDA wire is 8 bits. For every eight bit an acknowledge signal is produced. And finally stop condition complete the bus communication. Figure 5.5 illustrates writing and reading sequence on I²C bus. In the figure, 'S' represents a start condition, 'A' is an acknowledge condition and 'P' is a stop condition.

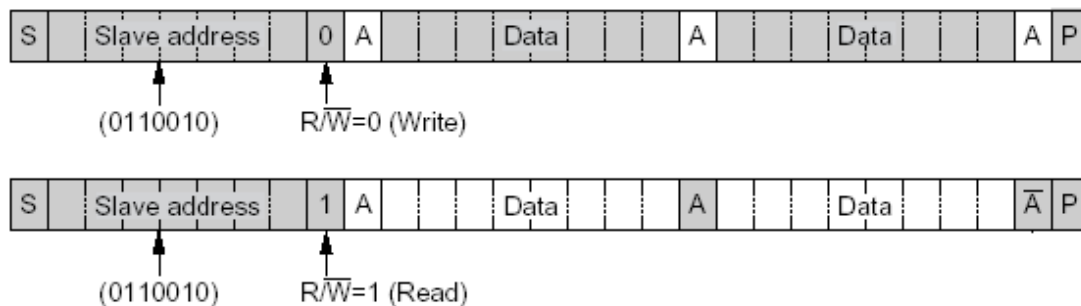


Figure 5.5 Write and read operations on the I2C bus

5.2.2 I2C Configuration

There are two composite video input and two TV decoder ICs on DE2-70 board. TV decoder ICs should be configured to obtain an 8 bit 4:2:2 ITU-R BT.656 data stream from a NTSC composite video format.

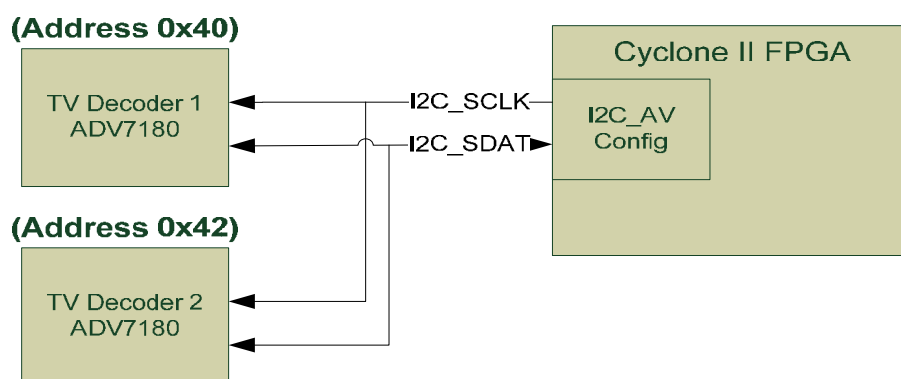


Figure 5.6 Relation between the FPGA and TV decoders on Altera DE2-70 board

There is an I²C block in our design named as *I2C_AV_CONFIG*, which sends configuration data to the TV Decoder (ADV7180) by using I²C protocol. I²C address of the TV decoder 1 and TV decoder 2 are 0x40 and 0x42 respectively as seen in Figure 5.6. I²C clock speed is set as 20 KHz which is obtained by dividing 50MHz internal clock. In our design, the data transfer is unidirectional and only writing sequence is required. After the start sequence signal, TV decoder address is sent. If the two video inputs will be used, both of the decoders should be configured. After the decoder selection, sub address of the data register and the data register is sent to the selected TV decoder. Finally, stop sequence signal terminates the I²C bus

communication. Table 5.1 gives a list of registers inside the ADV7180 TV decoder chip with their addresses, values and descriptions. The detailed information about the register bits can be found in ADV7180 datasheet.

Table 5.1 ADV7180 register addresses and register values

Register Address	Register Value	Register Name
0x00	0x00	Composite video input(Auto detects NTSC)
0xc3	0x01	ADC Switch 2
0xc4	0x80	ADC Switch 2
0x04	0x57	Extended Output Control
0x17	0x41	Shaping Filter Control 1
0x58	0x01	VS/FIELD Pin Control1
0x3d	0xa2	Manual Window Control
0x37	0xa0	Polarity
0x0e	0x80	ADI Control 1
0x08	0x80	Contrast
0x0a	0x18	Brightness
0x2c	0x8e	AGC Mode Control
0x2d	0xf8	Chroma Gain Control 1
0x2e	0xce	Chroma Gain Control 2
0x2f	0xf4	Luma Gain Control 1
0x30	0xb2	Luma Gain Control 2
0x0E	0x00	ADI Control 1

Upon power up, TV decoder registers are programmed via *I2C_AV_CONFIG* block with the values given in Table 5.1. During the I2C programming, TV decoder chip will be unstable for a time period.

5.3 ITU-R 656 Decoder Block

The output of the TV decoder IC ADV7180 is a standard digital video stream (8-bit 4:2:2 ITU-R BT.656). *ITU-R 656 Decoder* block extracts YCbCr 4:2:2 video signals from ITU-R BT.656 data stream. Additionally, data valid signal is generated to indicate that the YCbCr data is valid or not at the output of the *ITU-R 656 Decoder* block. Active video resolution of ITU-R BT656 data format for NTSC is 720 x 486. 486 lines of the 525 lines are visible lines and at each line there are 720 sampled pixels.

5.3.1 Down-sampling the Video Lines

While extracting the YCbCr data, down-sampling operation is performed. NTSC video uses non-square pixels. It means that, it has a pixel aspect ratio of 1:0.906. However, computer screens display square pixels and every pixel has an aspect ratio of 1:1. Thus 720 x 480 NTSC video corresponds to 640 x 480 squared pixel video. Thus, in order to compensate the difference between the square and non-square pixels, the down-sampling method is performed. Sampled pixels in each line are reduced to a lower resolution. In this work, it is aimed to display a VGA output with 640 x 480 resolution. Therefore, 720 x 480 video resolution is down-sampled to 640x480 video resolution (Jackson).



Figure 5.7 Down-sampling

Figure 5.8 illustrates the down-sampling for each line. Equation (5-1) is applied for down sampling method. For each 9 pixel one pixel is removed from the sequence. By performing this to a line, 720 sampled pixels is reduced to 640 pixels.

$$\text{Number of downsampled pixel} = 720 - \left(\frac{720}{9}\right) = 640 \text{ pixel} \quad (5-1)$$

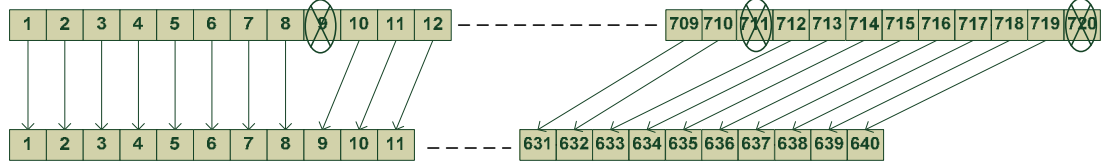
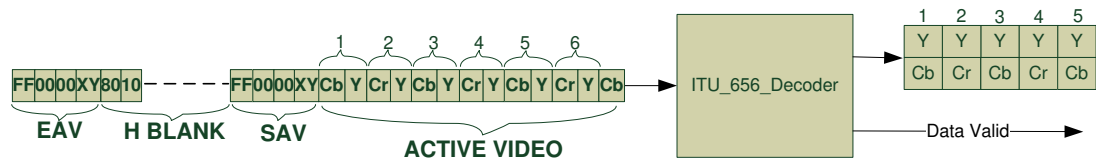
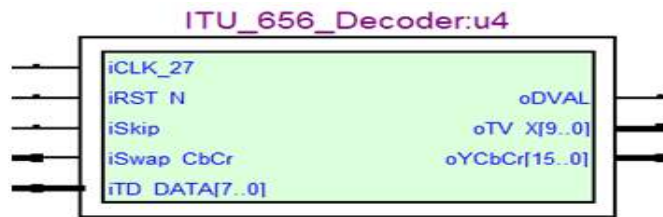


Figure 5.8 Down-sampling the TV lines

5.3.2 Extracting 4:2:2 YCbCr Data

8-bit ITU-R BT.656 4:2:2 digital data is down-sampled and extracted as 16-bit YCbCr 4:2:2 data stream in *ITU_656_Decoder* block. The input and output of the implemented *ITU_656_Decoder* block is illustrated in Figure 5.9. *Data Valid* signal and the active video parts (YCb, YCr) are transmitted at the output of the block.

Figure 5.9 *ITU_656_Decoder* blockFigure 5.9 RTL schematic of the *ITU_656_Decoder* block

The characteristics of the *Data Valid* signal can be seen from the debugging results. As seen in Figure 5.11, while YCbCr data is valid, *Data Valid* signal becomes active and for every nine pixel, one pixel becomes inactive.

Name	6862	6864	6866	6868	6870	6872	6874	6876	6878	6880	6882	6884	6886
ITD_Data	1Ch	80h	1Bh	7Fh	1Ch	80h	1Bh	7Fh	1Ch	80h	1Bh	7Eh	1Ch
...ru4pDVAL													
oYCbCr	1C7Fh	1B80h	1C7Fh	1B80h	1C7Fh	1B80h	1C7Fh	1A81h	2C7Ch	657Ch	7683h	737Ah	

Figure 5.10 Signal Tap II debugging results of the *ITU_656_Decoder* block

5.4 SDRAM Frame Buffer Block

A frame buffer is designed to capture a full video frame. The Frame buffer is necessary to combine the odd and even fields of an interlaced video. Altera DE2-70 board has two 32Mb SDRAM on it. In the design, there is a *SDRAM Control* block which controls all the writing and reading SDRAM operations between the FPGA and the external SDRAMs.

5.4.1 Writing Frame Buffer

The video stream generated from the TV decoder is still an interlaced video. In order to obtain a digital VGA output, de-interlacing operation must be performed on the interlaced video. Because the ITU-R BT656 is an interlaced video, odd fields and even fields come sequentially. Figure 5.12 shows the lines of an interlaced video.

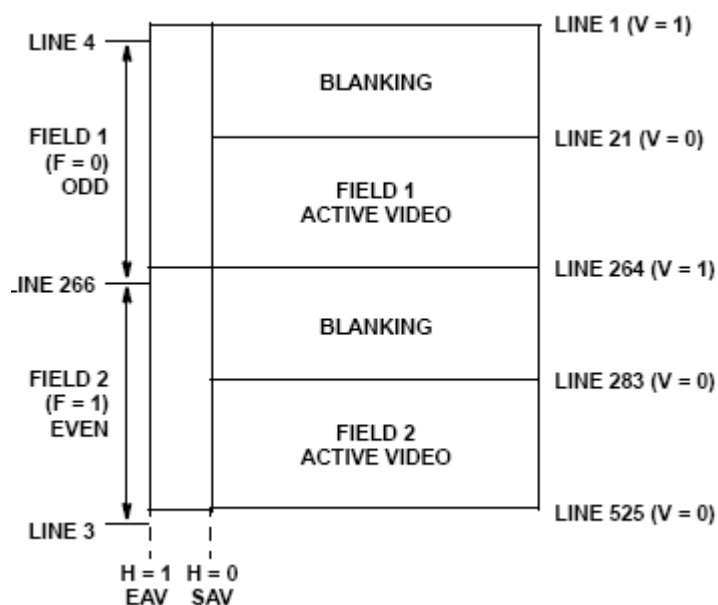


Figure 5.11 Typical BT.656 Vertical Blanking Intervals for 525/60 Video (Intersil Americas Inc, 2002).

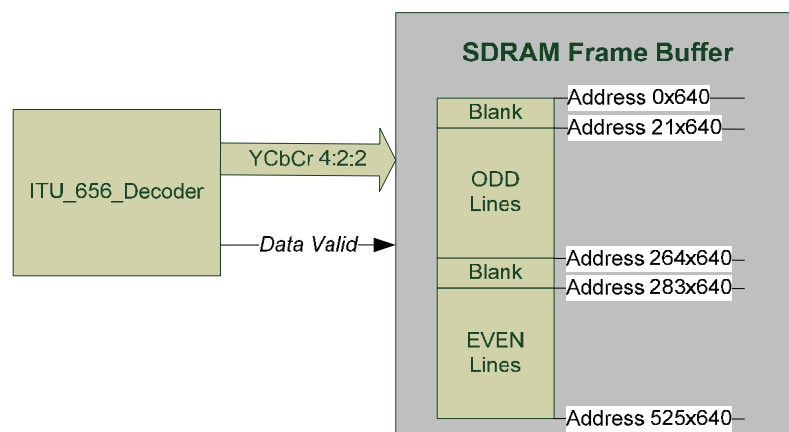
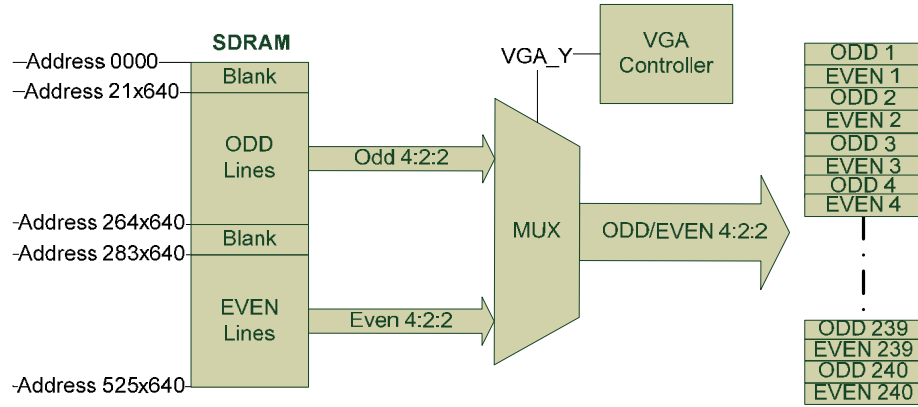


Figure 5.12 Writing the frame buffer

The YCbCr 4:2:2 video data is written to the SDRAM while the *Data Valid* signal is active. Frame Buffer (SDRAM) stores one full frame. Odd field lines and even field lines are stored separately due to the interlaced data. As seen in Figure 5.13, each line contains 640 samples. Required memory address size is 525x640 addresses for a full-frame buffer. Each address includes 16-bit 4:2:2 YCbCr data.

5.4.2 Reading Frame Buffer

VGA Controller block generates VGA_Y signal which indicates the number of required line in current time. According to the VGA_Y signal the read address of the SDRAM is multiplexed. VGA_Y signal gives the information about which line should be read from the memory. Odd lines and even lines are read sequentially as seen in Figure 5.14.



5.13 Reading the frame buffer

5.4.3 De-interlacing

By multiplexing the reading address of the SDRAM, progressive video lines are created. In order to decrease the line crawling effect, one of the de-interlacing techniques is performed. For each line, the effects of the previous and the next lines are added to the current line. In our design, there are two line buffers which hold the data of the previous video lines. While calculating any line previous and the next line are used for de-interlacing. The Figure 5.15 shows the relation between the lines during the de-interlacing operation.

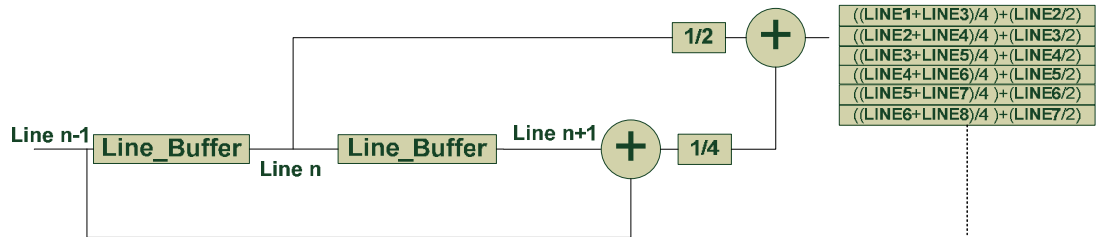


Figure 5.14 De-interlacing

As seen in figure, while calculating the current line (n), $\frac{1}{4}$ of the previous line ($n-1$) and $\frac{1}{4}$ of the next line ($n+1$) are added to the $\frac{1}{2}$ of the current line (n). The equation (5-2) summarizes the implemented de-interlacing algorithm.

$$line_{new} = \frac{1}{4} (line_{n-1} + line_{n+1}) + \frac{1}{2} (line_n) \quad (5-2)$$

During the de-interlacing operation, optimization is performed for each pixel of the each line. While calculating the value of the each pixel, vertical neighbor pixels are being considered as in Figure 5.16.

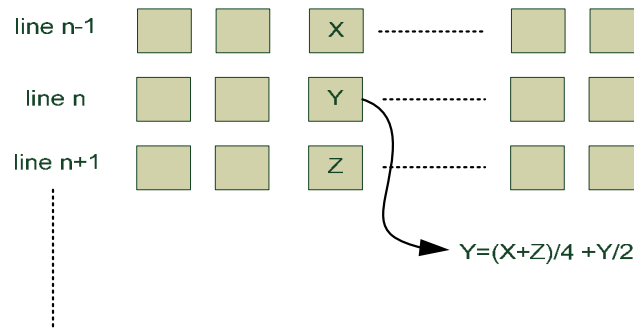


Figure 5.15 De-interlacing operation for each pixel

5.5 YUV422 to YUV444 Block (Chroma Resampler)

After the de-interlacing operation, 16-bit 4:2:2 YCbCr data should be resampled to 24-bit 4:4:4 YCbCr data. Figure 5.17 illustrates the inputs and outputs of the *YUV422_to_YUV444* block.

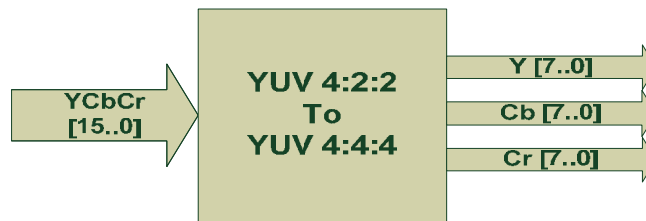


Figure 5.16 *YUV422_to_YUV444* Block

Interpolation is performed for the missing Cb and Cr values. As seen from the Figure 5.18, in YCbCr 4:2:2 format for each horizontal Y sample, there is one Cb or one Cr sample. For the missing Cb, Cr values the previous chroma components are replicated.

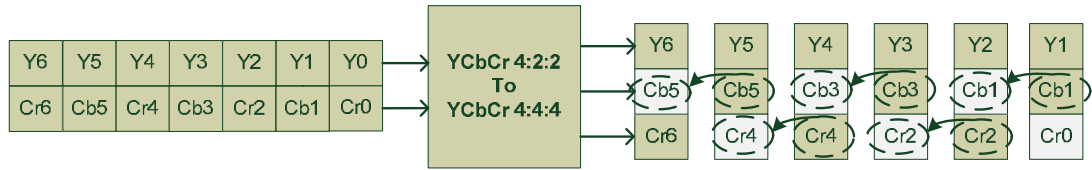


Figure 5.17 Chroma resampling

5.6 YCbCr to RGB Block

YCbCr to RGB block transforms the 24 bit YCbCr color space to 30 bit RGB color space. In chapter two, the equations about the color space conversion is given. Multipliers are used to perform the calculations given in the equations below. Because the VGA DAC IC on DE2-70 board accepts 10 bit R, G, B signals, the 8 bit R, G, B signals are extended to 10 bit R, G, B signals by shifting 2 bit left for each signal.

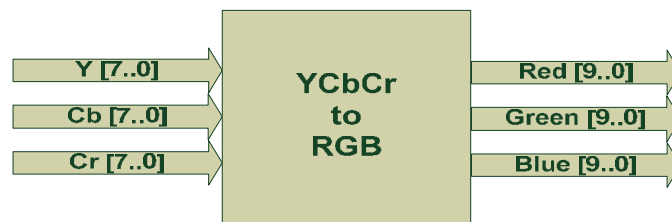


Figure 5.18 Color space conversion block

$$R = 1.164(Y - 16) + 1.596(Cr - 128) \quad (5-3)$$

$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \quad (5-4)$$

$$B = 1.164(Y - 16) + 2.018(Cb - 128) \quad (5-5)$$

5.7 VGA DAC IC

The ADV7123 chip is a digital to analog video converter and can support resolutions of up to 1600x1200 pixels, at 100Hz. It consists of three high speed, 10-bit, digital to analog converters for video signals. The ADV7123 generates the appropriate analog RGB signals with the correct blanking levels. RGB (red, green and blue) data, a sync signal, a blank signal and a pixel clock signal should be provided in order to generate a valid VGA signal from the ADV7123 chip

(ADV7123 Datasheet). Figure 5.19 shows the VGA circuit schematic on DE2-70 board.

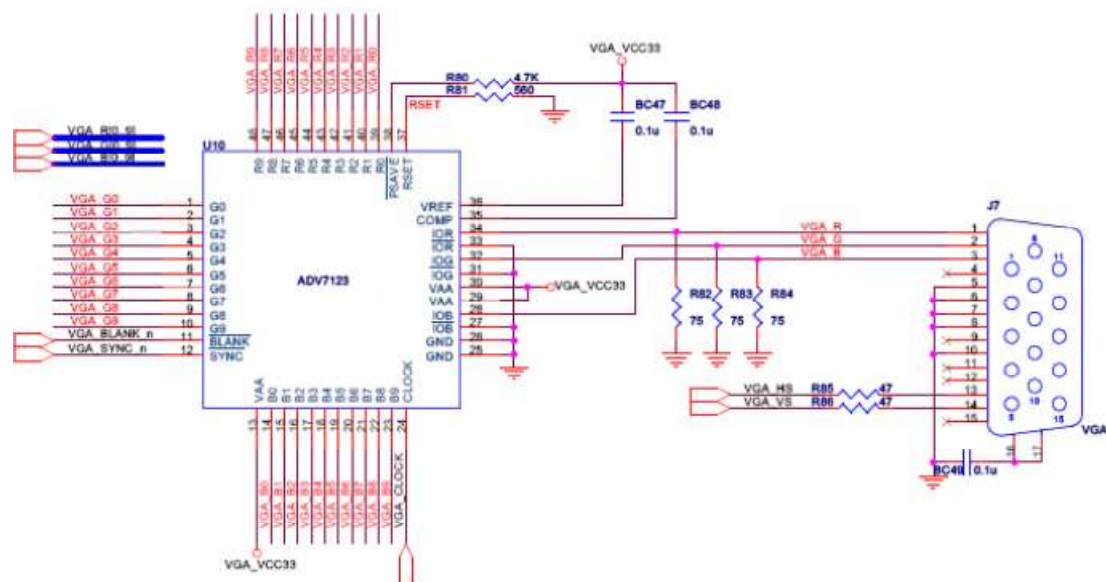


Figure 5.19 VGA circuit schematic

Note that the horizontal and vertical synchronization signals are generated directly by the FPGA, and do not pass through the ADV7123. It means that you need to provide the horizontal and vertical synchronization signals of your video controller as output in order to have a complete VGA signal. All the necessary VGA timing signals are generated in *VGA Controller* block.

5.8 VGA Controller Block

This block is used to generate the VGA timing signals required for Video DAC IC (ADV7123). VGA synchronization signals are generated directly from the *VGA Controller* block. Also *RGB* signals, *VGA_BLANK* and *VGA_SYNC* signals that are needed for the ADV7123 IC are provided from this block. Figure 5.20 shows the *VGA Controller* block and connections to VGA DAC IC and VGA connector on DE2-70 board.

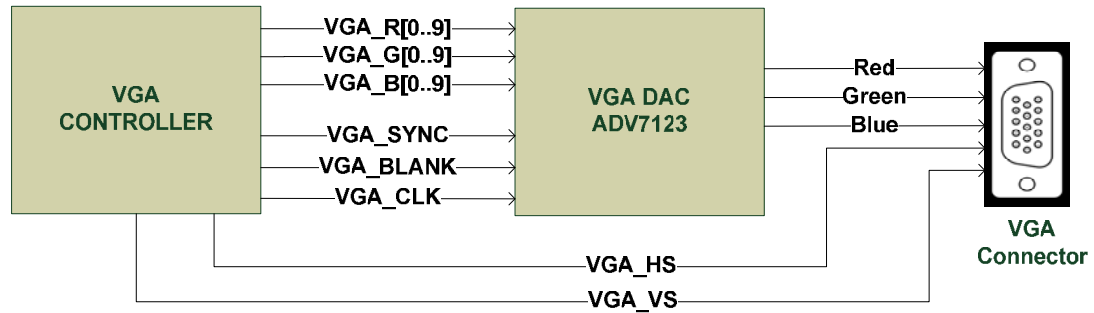


Figure 5.20 VGA timing signals generated by *VGA Controller* block

The RGB signals provided from the *YCbCr_to_RGB* block are transmitted over *VGA Controller* block without any change. 27MHz clock is used as the pixel clock (*VGA_CLK*). *VGA_HS* and *VGA_VS* signals are generated from this pixel clock. In chapter two, VGA interface standards and synchronization timings are explained. The signal timings for 640x480 video at 60Hz refresh rate are given as in Table 5.2. In this thesis, the timings in this table are used for the VGA video.

Table 5.2 Horizontal and Vertical Sync timings

	Pixel Clock(27Mhz)		Number of lines
H_FRONT	16	V_FRONT	11
H_SYNC	96	V_SYNC	2
H_BACK	48	V_BACK	31
H_ACT	640	V_ACT	480

The *VGA_BLANK* is an active low signal and it should be active low during both the horizontal sync period and vertical sync period. While the *VGA_BLANK* signal is a logical zero, RGB pixel inputs sent from *VGA Controller* block are ignored. *VGA_BLANK* is low during the *H_BLANK* and *V_BLANK* timing periods. *H_BLANK* and *V_BLANK* are calculated as:

$$H_BLANK = H_FRONT + H_SYNC + H_BACK \quad (5-6)$$

$$V_BLANK = V_FRONT + V_SYNC + V_BACK \quad (5-7)$$

CHAPTER SIX

IMPLEMENTATION OF VIDEO APPLICATIONS

6.1 Introduction

FPGA board should have the necessary hardware for the parallel video applications. Altera DE2-70 board has two TV decoders and RCA jacks that allow designer to process two video sources simultaneously.

In the previous chapter, the implementation blocks to process a single video source are explained in details. Video blocks are developed and implemented successfully in Cyclone II FPGA. The main goal of this thesis is to process two video sources at the same time. In FPGA design, it is possible to implement functional blocks as possible as the FPGA hardware allows. Therefore, the previous approach for single video source implementation is extended in order to realize parallel video implementations in FPGA. The implemented blocks are duplicated for the second video source. However, some implemented blocks cannot be duplicated due to the several hardware constraints. The reason of these constraints is generally the limited logic sources used in the implemented FPGA.

In this study, the most important hardware constraint was the dedicated clock pins inside the FPGA. Designers have to use this dedicated pins. Cyclone II (EP2C70) FPGA has also dedicated clock pins. The Figure 6.1 shows the clock distribution on DE2-70 board. As seen from the figure the *TV decoder 1* and *TV decoder 2* ICs generate clock inputs to Cyclone II. In this study, these clocks are used as global clocks of the implemented video blocks for video applications.

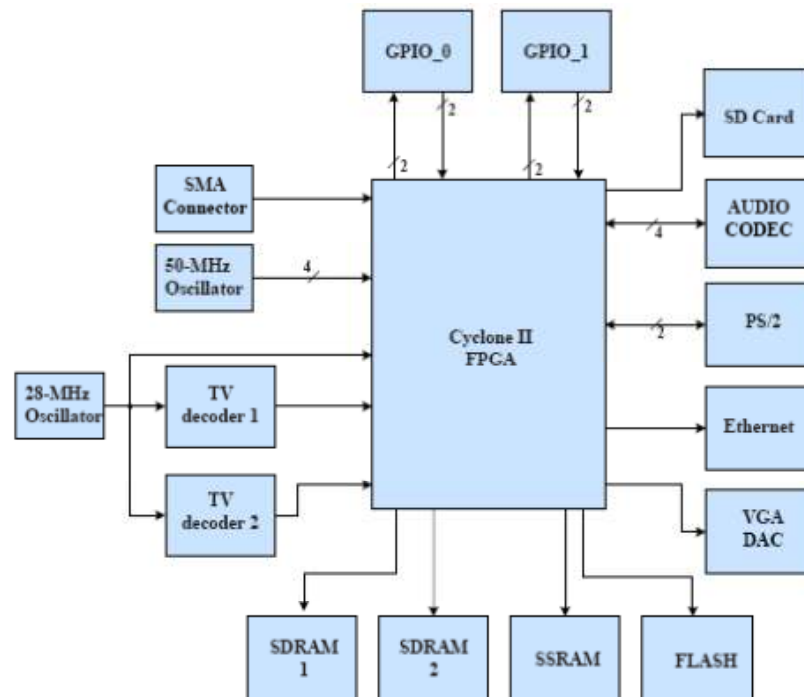


Figure 6.1 Block diagram of the clock distribution (Terasic, 2009)

Cyclone II device provides global clock networks and four PLLs for a complete clock management system. As seen in Figure 6.2, each global clock network has a clock control block to select from a number of input clock sources (PLL clock outputs, CLK[] pins, DPCLK[] pins and internal logic) to drive onto the global clock network. Note that CLK[] pins are dedicated clock pins and DPCKL[] pins are dual-purpose clock pins (Altera, 2007).

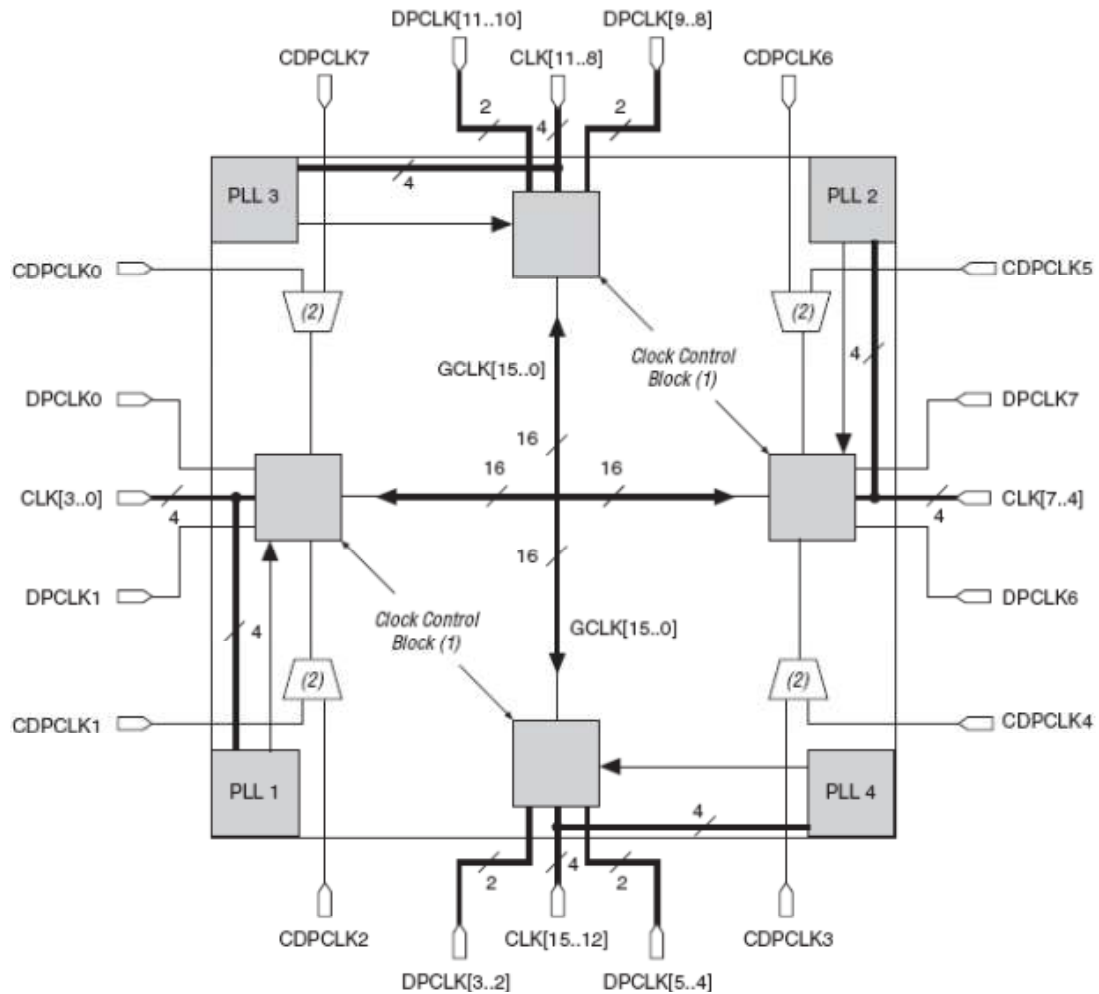


Figure 6.2 PLL, CLK[], DPCLK[] and Clock Control Block Locations

In our design, *iTD1_CLK27* is a clock signal generated by ADV7180 TV decoder IC and used as a system clock. While the video signal data is written to the SDRAM, clock rate should be increased. 108MHz clock signal is generated from the 27MHz *iTD1_CLK27* clock signal by using the PLLs in the FPGA. In Cyclone II FPGA, CLK[10] is the dedicated clock pin for *iTD1_CLK27* clock signal and it uses PLL3 as seen in Figure 6.2.

There is a second TV decoder IC in DE2-70 board for the second video source. The generated clock of the TV decoder, *iTD2_CLK27*, is used in video blocks implemented for the second video source. CLK[11] is the dedicated clock pin for *iTD2_CLK27* clock signal. As seen in figure both dedicated clock pins CLK[10] and CLK[11] are routed to the same PLL. There is only one PLL for the two clock pins

and only one of them can be use this PLL at a specific time. Therefore, the implemented designs for the both video sources cannot be completely independent from each other. PLL3 should be used in common for both of the dedicated clock signals (*iTD1_CLK27*, *iTD2_CLK27*).

Firstly, the design in Figure 6.3 is implemented. The same video blocks implemented for the first video is duplicated for the second video. A common PLL is used for both of the SDRAM. By switching the input of the PLL, two video sources can be processed and displayed separately on CRT/LCD screen. Here, the PLL switching and RGB switching operations are performed manually by using the toggle switches on the DE2-70 board.

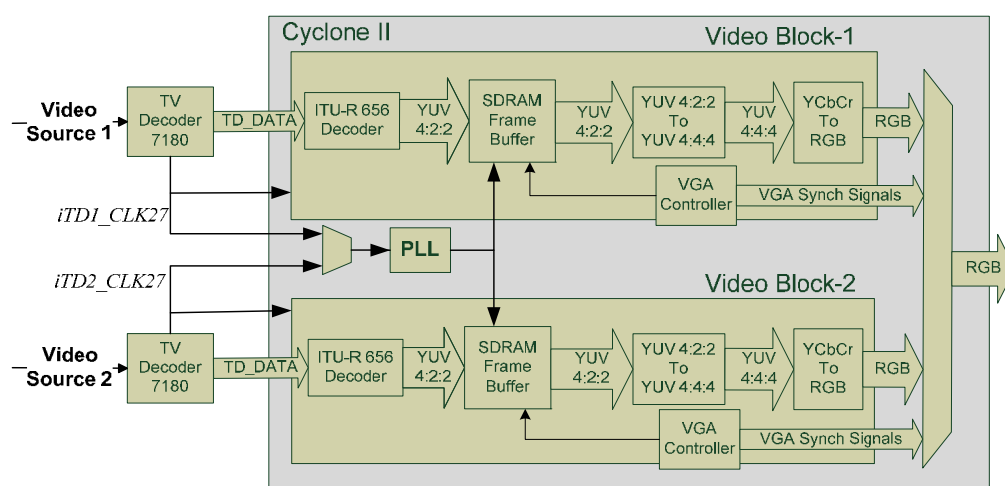


Figure 6.3 Block diagram of two video sources with PLL switching

After it is seen that the implementation of the two video sources in Cyclone II FPGA is being processed successfully, the next step is to implement frame sequential video application.

6.2 Frame Sequential Video Implementation

In the previous section, the process for two video sources is explained. And it is proved that two different video sources can be displayed separately by manual switching between the video blocks. In frame sequential video application, the used

method is similar to the switching video operation mentioned above. The difference is that, in frame sequential video the output video is composed of two different video signals. At each frame one of the input video sources is transmitted to output in a sequential manner.

In the beginning of frame sequential design, PLL is switched at the end of every frame. This caused timing constraints. Instead of switching the input of the PLL after each frame, the generated clock from *iTD1_CLK27* is used for both SDRAM. There are two SDRAMs in the system and each of them is used as a frame buffer. Also the number of the *VGA Controller* blocks decreased to one to provide a stable system. *VGA Controller* block generates *VGA_HS* and *VGA_VS* synchronization signals and *VGA* timing signals required for video decoder ICs. End of each video frame can be detected with *VGA_VS* signal. In this study, the *VGA_VS* signal is used to detect the new frame. At this point, a new block named as *Frame Controller* is designed and added to the system. The *Frame Controller* block receives the *Request* signal from the *VGA Controller* block and at each frame the read request signal is sent to a different SDRAM. Therefore, at each frame, YUV 4:2:2 video data from a different source is transmitted in the system. The Figure 6.4 shows the main block diagrams of the frame sequential video design.

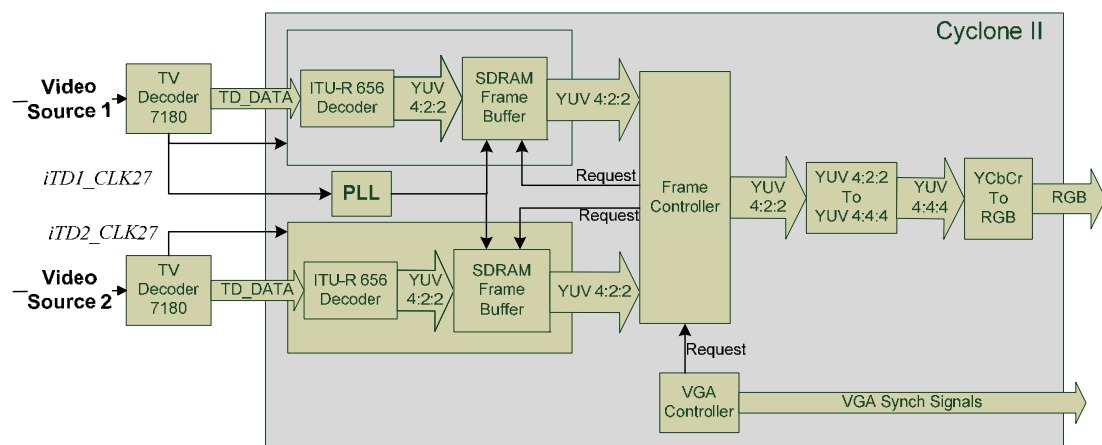


Figure 6.4 Block diagram of Frame Sequential video design

After the implementation, the output is observed in a VGA monitor. The video frames are flickering because of the low refresh rate. The refresh rate of the output

video is 60 Hz which is an acceptable frame rate for human eye. However, there are two different video frames transmitted sequentially. Therefore each video has a refresh rate of 30 Hz and this causes the flickering. In order to see the success of the implementation, the VGA output video is connected to the head mounted display (5DT HMD 800-26). This HMD accepts frame sequential video input. As a result, it has separated the video frames successfully. The video frames are seen differently by the right eye and left eye.

6.3 Field Sequential Video Implementation

In field sequential method, the output video is composed of the odd fields from one video source and the even fields from another video source. While reading the video data from frame buffer, de-interlacing operation is performed. In this study, the odd lines and the even lines are read from different addresses in a sequence. If the odd lines are read from the first SDRAM and the even line from the second SDRAM, the combination of the two video sources can be seen at each frame in the output video.

The design used in frame sequential video is modified and used for the field sequential video implementation. As seen from Figure 6.4, *Frame Controller* block controls the reading video data from the SDRAM frame buffers. *VGA Controller* block counts the number of row in current time and generates *VGA_Y* row counter signal. *Frame Controller* block uses *VGA_Y* information and sends the read request to the first SDRAM for the odd lines and to the second SDRAM for the even lines.

In addition to the FPGA implementation, field sequential view of the two images is observed by implementing in MATLAB. Odd lines of the left images and even lines of the right image are combined in a new image.

MATLAB Code:

```
%% Read Images
img_left  = imread('scene_l.bmp');
img_right = imread('scene_r.bmp');
```

```

%% Convert the images to YCbCr format
img_left = rgb2ycbcr(img_left);
img_right = rgb2ycbcr(img_right);

%% Size of Images
row = size(img_left,1);
col = size(img_left,2);

%% Line Sequential
odd_lines = 1:2:row;
even_lines = 2:2:row;

img_line_seq(odd_lines, :, :) = img_left(odd_lines, :, :);
img_line_seq(even_lines, :, :) = img_right(even_lines, :, :);

img_line_seq = ycbcr2rgb(img_line_seq);
figure, imshow(img_line_seq)
title('Result of Line Sequential')

```

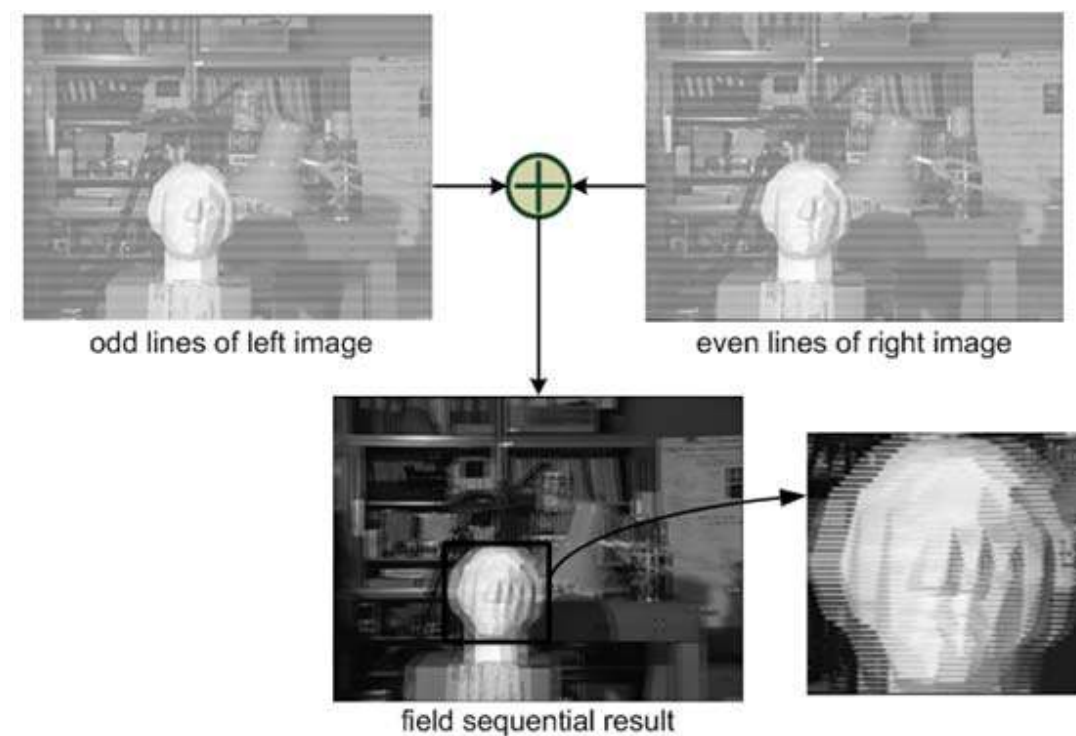


Figure 6.5 MATLAB result of Field Sequential image

6.4 Chroma Keying Implementation

In the chroma keying application two video cameras are used. One of video sources is for foreground video and the other video source is for the background video. In order to realize this application properly, the backdrop color of the foreground video should be solid colored. So far in our design, the video is just read

from the SDRAM and transmitted to the output. While designing the chroma keying application, both of the SDRAMs should be read at the same time. In the design there is a decision block for each pixel. YCbCr and RGB color spaces can be used in chroma keying applications. As explained in previous chapters, YCbCr color space is converted to RGB color space in *YCbCr_to_RGB* block. In YCbCr color space, the color information is transmitted on Cb and Cr components and the luminance information is transmitted on Y component. The required key color can be obtained by defining only the Cb and Cr colors. Also by using the luminance information, realism of the chroma keying function can be increased. On the other hand, in RGB color space all of the components should be checked to detect a specific chroma key color. Therefore YCbCr color space is chosen in this study. The design made for the frame sequential video application is modified. Cb and Cr components of the foreground video frame are compared with the determined chroma key interval. This comparison is performed for each pixel of the foreground video. If the compared pixel is between the required chroma key color limits, instead of the foreground pixel, the background pixel from the second video source is transmitted. Figure 6.6 shows the block diagram of the chroma keying. Any pixel from the foreground or background frame can be switched according to this comparison.

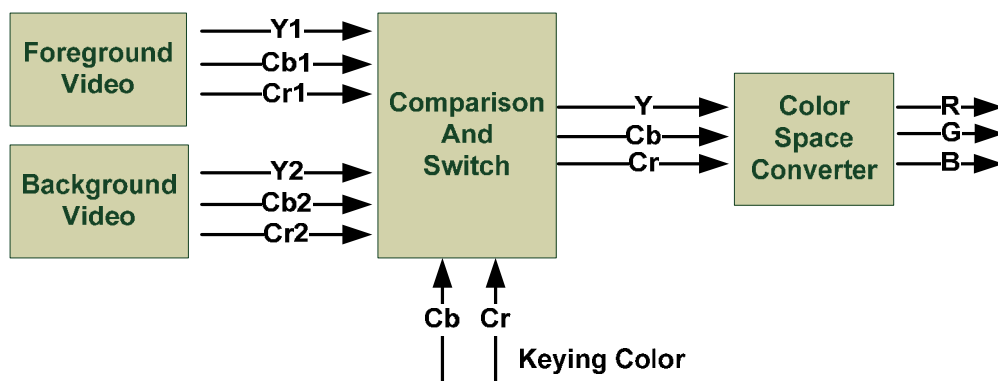


Figure 6.6 Chroma keying comparison

In chroma keying applications, the backdrop color can be in different green color tones because of several factors. Therefore threshold levels of the chroma keying are not fixed in this work. Toggle switches on DE2-70 are used for the adjustment of the

threshold levels. In this way, a more flexible solution is found for the chroma keying color and the most suitable levels can be chosen simultaneously.

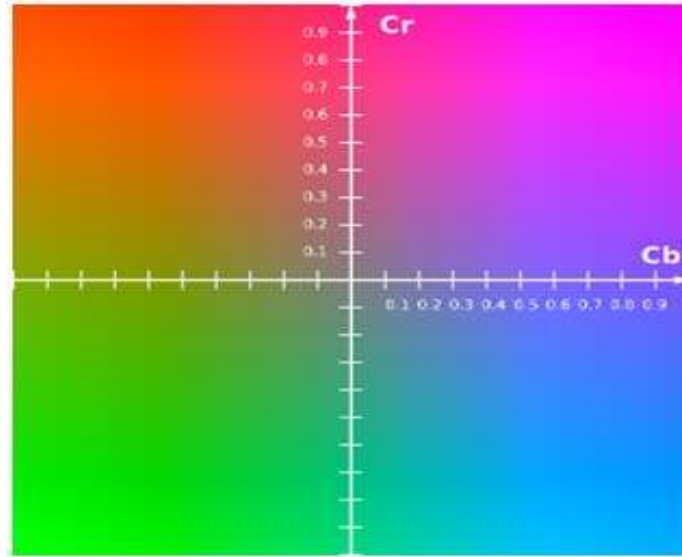


Figure 6.7 Cb-Cr plane at constant luminance value ($Y'=0.5$)

The Figure 6.7 shows the Cb-Cr color distribution with a constant luminance value ($Y=0.5$). A threshold for the green color key can be detected from this figure. As seen in figure, green color tones correspond to the low values of the Cb and Cr values. For example, according to the YCbCr to RGB calculations mentioned in previous chapters, pure green color RGB (0, 255, 0) is equal to YCbCr (145, 54, 34).

In this work, 14 toggle switches are used to define the Cb and Cr values. Because the number of the toggle switches is limited in DE2-70 board, only the upper limit thresholds are defined. The function of the switches is illustrated as in Figure 6.8. Toggle switches cause logic 0, when in the down position and logic 1 when in the up position.

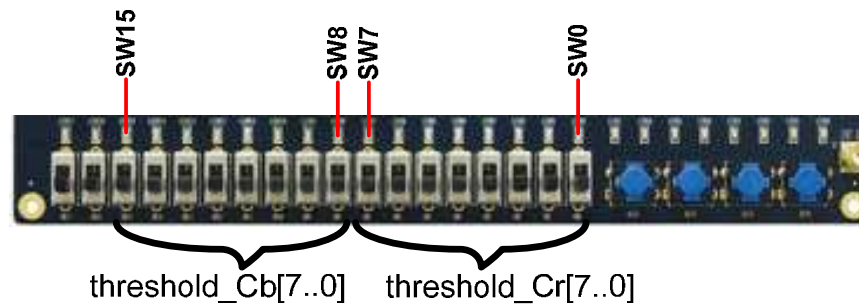


Figure 6.8 Function of toggle switches for chroma keying application

If the Cb and Cr values of the foreground image is smaller than $threshold_Cb[7..0]$ and $threshold_Cr[7..0]$, it means that the green color is detected and the pixels belong the background video is transmitted instead of foreground video. Chroma keying method is implemented in MATLAB. The threshold for the Cb and Cr is chosen as “110”.

MATLAB Code:

```
%% Read Images
img_foreground = imread('foreground1.jpg');
img_background = imread('background1.jpg');
img_foreground = rgb2ycbcr(img_foreground);
img_background = rgb2ycbcr(img_background);

row = size(img_background,1);
col = size(img_background,2);
% Thresholding
Cb_th = 110;
Cr_th = 110;

img_chroma_key = img_foreground;
% Chroma Keying
for i = 1:row
    for j = 1:col
        if ( (img_foreground(i,j,2) < Cb_th) &&
            (img_foreground(i,j,3) < Cr_th) )
            img_chroma_key(i,j,:) = img_background(i,j,:);
        end
    end
end

figure,imshow(ycbcr2rgb(img_chroma_key))
title('Result of Chroma Keying')
```

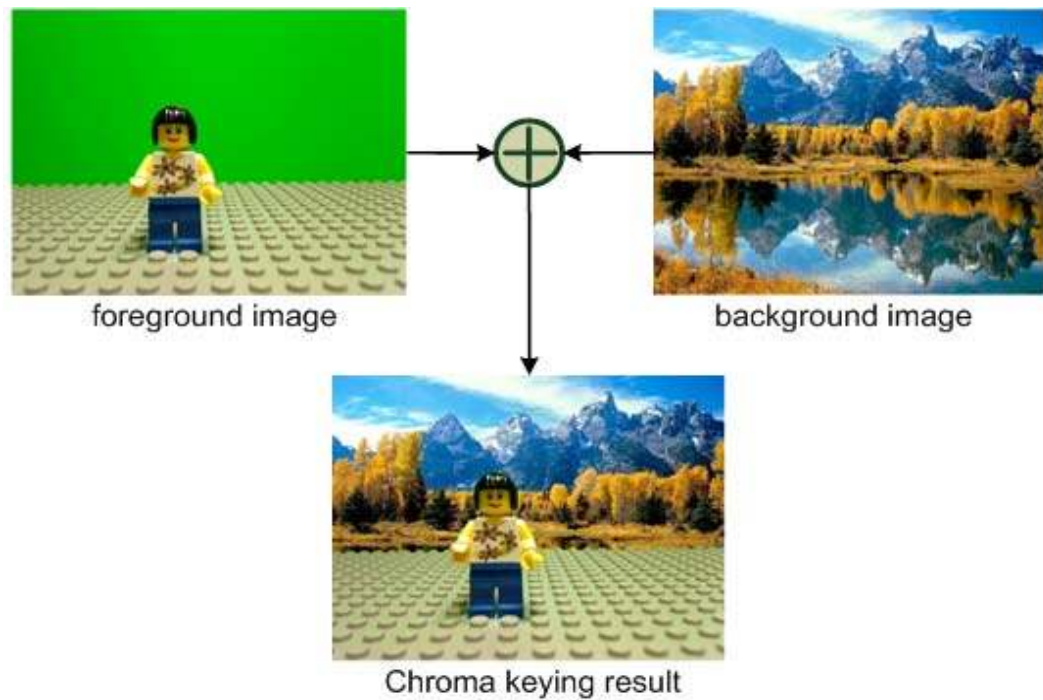


Figure 6.9 MATLAB result of chroma keying($Cb_{th}=110$, $Cr_{th} = 110$)

As seen from the MATLAB result, the green pixels of the foreground image are replaced with the background image. In the following figure a photograph from a VGA monitor can be seen while FPGA implementation is performing the chroma keying application.



Figure 6.10 Chroma keying result at the output of the FPGA

6.5 Anaglyph Implementation

In anaglyph application, RGB color spaces of the two video frames should be used. In the previous applications, *Frame_Controller* block processes the YCbCr video signals. Now the RGB signals of the both video sources are combined for an anaglyph video. In addition to the previous designs, *YUV422_to_YUV444* and *YCbCr_to_RGB* blocks are also duplicated as seen in Figure 6.11. *Frame_Controller* block controls the RGB signals come from the both video sources.

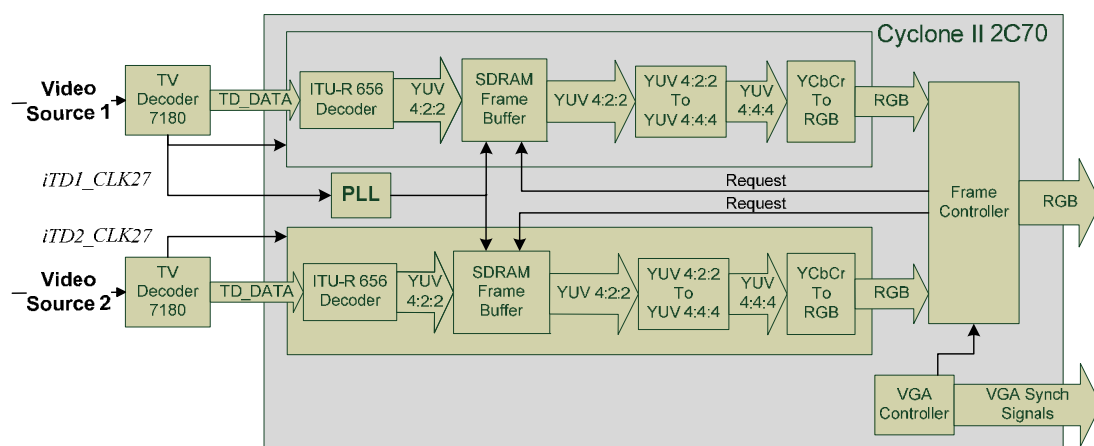


Figure 6.11 Block diagram of Anaglyph video design

In the *Frame_Controller* block RGB signal coming from the *video source1* and *video source2* are controlled by the toggle switches. Six toggle switches (*SW5* to *SW0*) are assigned for this task. Figure 6.12 illustrates the function of each toggle switch. These switches allow or block the RGB signals. The toggle switches for the *video source1* have a higher priority. As a result, *SW5*, *SW4*, *SW3* has a higher priority than *SW2*, *SW1*, *SW0*. It means that, when the *SW5* is logic 1, The R (red) signal of the first video source is transmitted whatever the status of *SW2* is.

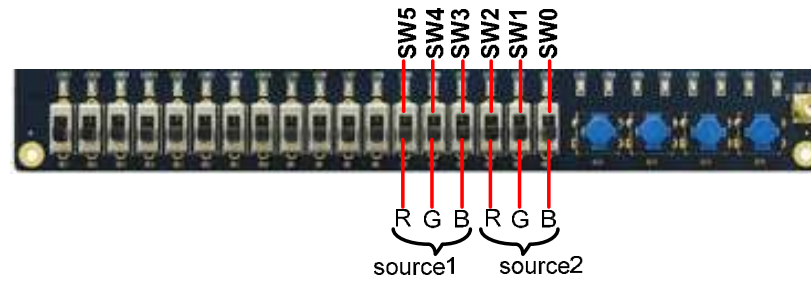


Figure 6.12 Function of toggle switches for anaglyph application

In order to see the anaglyph video with Red-Cyan glasses, the switch combination in the Table 6.1 should be performed. In addition, different anaglyph color variations can be realized by changing the status of the switches as seen in the table.

Table 6.1 Settings for the anaglyph application

Anaglyph Method	VIDEO IN 1			VIDEO IN 2		
	Red	Green	Blue	Red	Green	Blue
	SW5	SW4	SW3	SW2	SW1	SW0
Red-Cyan	ON	OFF	OFF	OFF	ON	ON
Cyan-Red	OFF	ON	ON	ON	OFF	OFF
Red-Blue	ON	OFF	OFF	OFF	OFF	ON
Blue-Red	OFF	OFF	ON	ON	OFF	OFF

Also the anaglyph algorithm is implemented in MATLAB.

MATLAB Code:

```
%% Read Images
left  = 'scene_l.bmp';
right = 'scene_r.bmp';
img_left  = imread(left);
img_right = imread(right);

%% Convert the images to YCbCr format
img_left  = rgb2ycbcr(img_left);
img_right = rgb2ycbcr(img_right);

%% Anaglyph
img_left_rgb  = ycbcr2rgb(img_left);
img_right_rgb = ycbcr2rgb(img_right);
% R Channel
img_anaglyph(:, :, 1) = img_left_rgb(:, :, 1);
% G Channel
img_anaglyph(:, :, 2) = img_right_rgb(:, :, 2);
% B Channel
img_anaglyph(:, :, 3) = img_right_rgb(:, :, 3);
```

```
figure,imshow(img_anaglyph),title('Result of Anaglyph(Red-Cyan)')
```

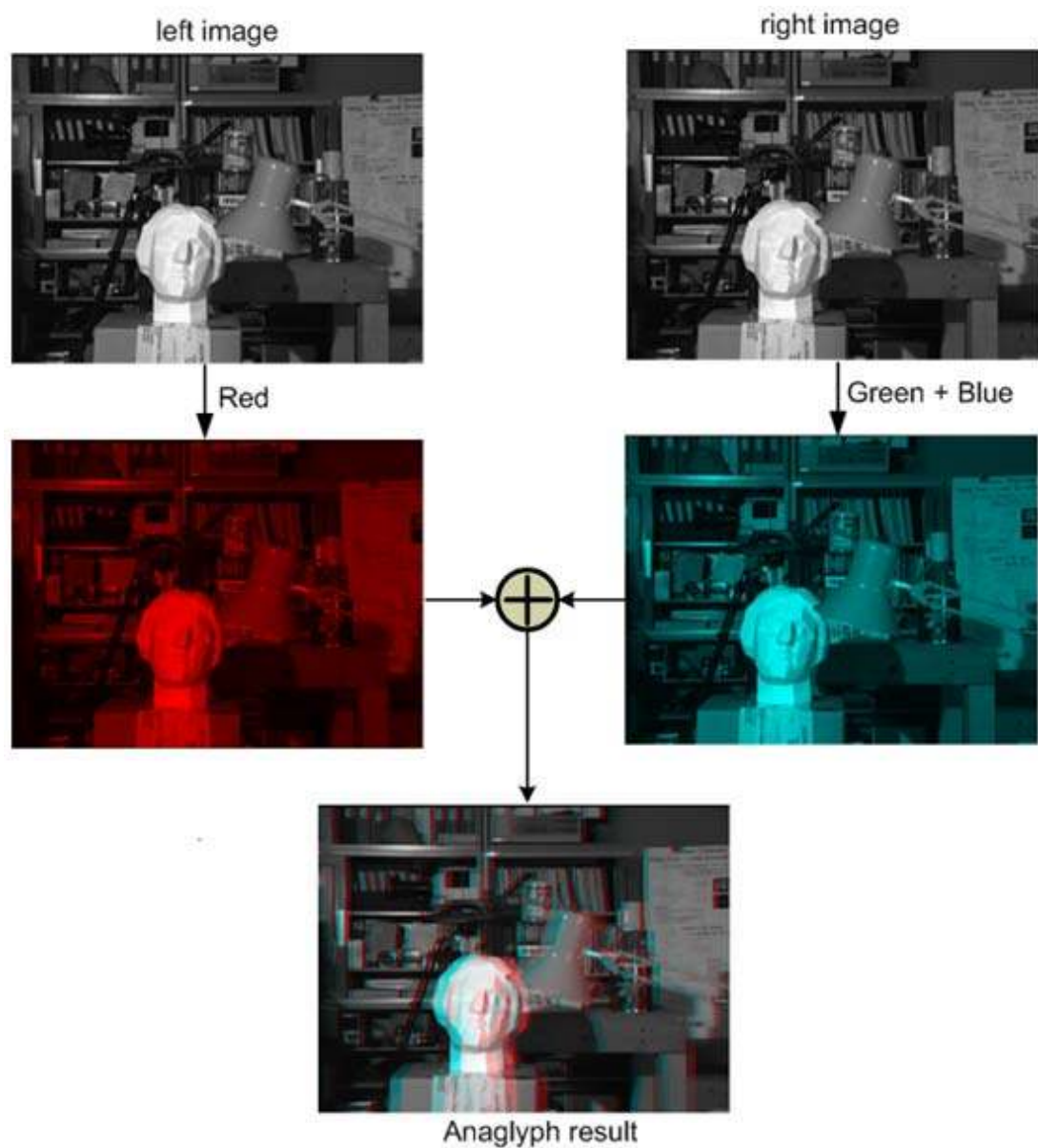


Figure 6.13 MATLAB result of anaglyph (Red/Cyan)

As seen from the MATLAB result in Figure 6.13, 3D effect can be seen with Red/Cyan glasses. In the following figure a photograph from a VGA monitor can be seen while FPGA implementation is performing the anaglyph application.



Figure 6.14 Anaglyph video result at the output of the FPGA

6.6 PIP (Picture in Picture) Implementation

In this application, two different video source signals from the TV decoders are multiplexed and both video signals are displayed on the CRT/LCD monitor using picture in picture mode. One video is displayed on the full screen while the other video is displayed in a small sub window.

Firstly a memory RAM block is designed with 8-bit word size to store the luminance (Y) values of the sub window video. For sub window video we have to the scaling and no need to store every pixel in RAM blocks. It is decided to scale the sub-window to the 1/4 of the real sizes. Therefore, for each four pixel one pixel is sampled in vertical and horizontal directions. Figure 6.15 illustrates this operation. Red dots are the stored pixels in RAM block. In this way, 640x480 pixel resolution video is scaled to 160x120 pixel resolution.

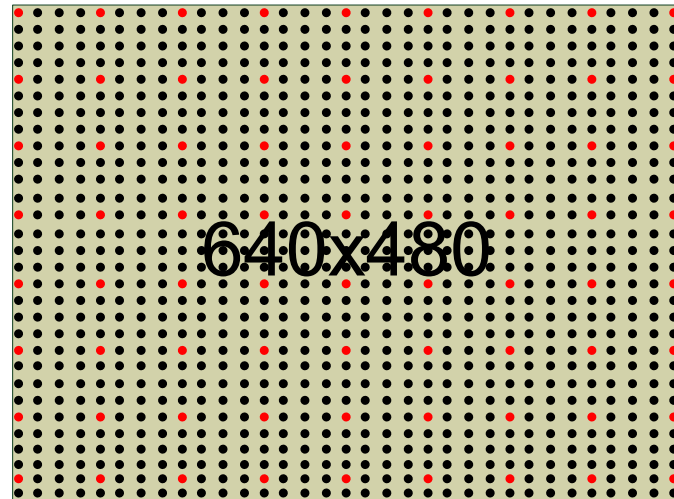


Figure 6.15 Sub Sampling the sub-window video. (Red dots are the sampled pixels)

VGA_X and VGA_Y are *VGA Controller* block signals which hold the row and column number in current time. When the $0 < VGA_X \leq 160$ and $0 < VGA_Y \leq 120$, then the RAM block is read and the sub window pixels are transmitted at that timing intervals.

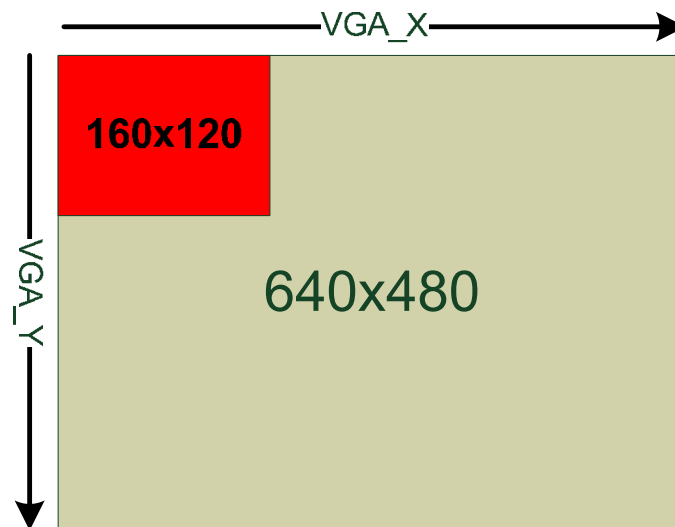


Figure 6.16 Video resolutions of the main video and sub-window

Because of the storing only the Y component of the video source, the image in sub-window was in gray scale. In order to make a colored sub-window, the RAM block word size is extended to 16-bit.

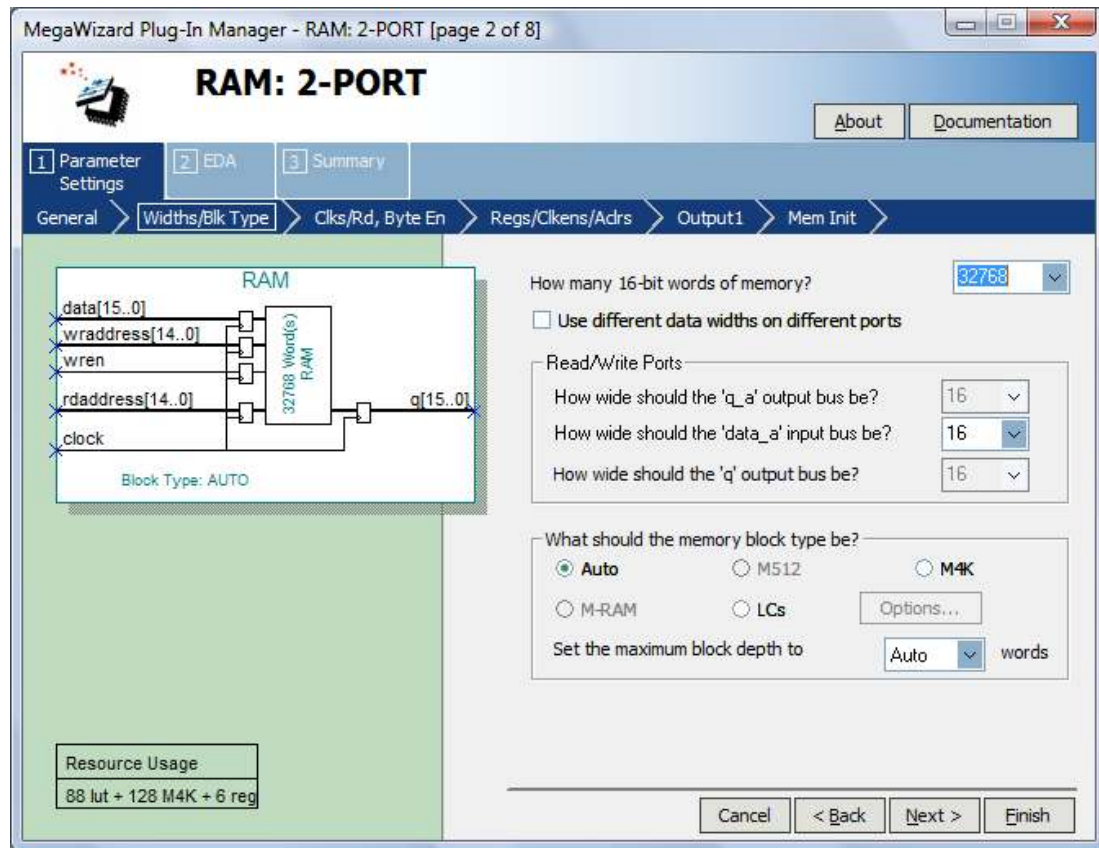


Figure 6.17 General settings for the function of memory in RAM mode

The used RAM block megafunction is illustrated in Figure 6.17. The memory length and width are static parameters supported in memory core. The minimum size of the memory should be $160 \times 120 = 19200$ word size. The word size of the RAM is 16-bit but our YCbCr data is 24-bit. Therefore, we just stored the most significant four bit of the chrominance components (Cb, Cr). The stored 16 bit data for each pixel is as in the following table.

Table 6.2 Values of each bits of the word in the RAM block

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	Y[7]	Y[6]	Y[5]	Y[4]	Y[3]	Y[2]	Y[1]	Y[0]	Cb[7]	Cb[6]	Cb[5]	Cb[4]	Cr[7]	Cr[6]	Cr[5]	Cr[4]

The rest of the Cb and Cr bits are filled as “0111” which is the average of the corresponding bits. Therefore the colored sub-window image is obtained. Because of the scaling in the sub-window video, the loss in chrominance component is acceptable and is not bothering the viewers.

Another implemented feature of this application is that, by using the toggle switches, the video sources displayed on the full screen and sub window can change place. As seen from the Table 6.3, when the *SW0* is logic 1, main window video is *video source1* and sub-window is *video source2*, and vice versa when the *SW0* is logic 0.

Table 6.3 The setup for the PIP application

<i>SW0</i>	PIP Display Mode
ON	Main window : Video in 1 Sub window : Video in 2
OFF	Main window : Video in 2 Sub window : Video in 1

In the following figure a photograph from a VGA monitor can be seen while FPGA implementation is performing the picture in picture application.



Figure 6.18 PIP application result at the output of the FPGA

6.7 Combination of the All Designs in One Implementation

All of the video applications mentioned above are distinct implementations in FPGA. One of them can be implemented and used at a time. In this section, all of these implementations are combined in one design. All the controls are made in *Frame Controller* block as seen in Figure 6.20.

Also a user interface is needed for these applications. A menu interface is developed. The user can select the wanted application via push-buttons on DE2-70 board.

Firstly, the appearance of the menu is designed using a paint program. The size of the menu is defined as 300x200 pixel. It is decided to store the information about the menu screen pixels in memory blocks.

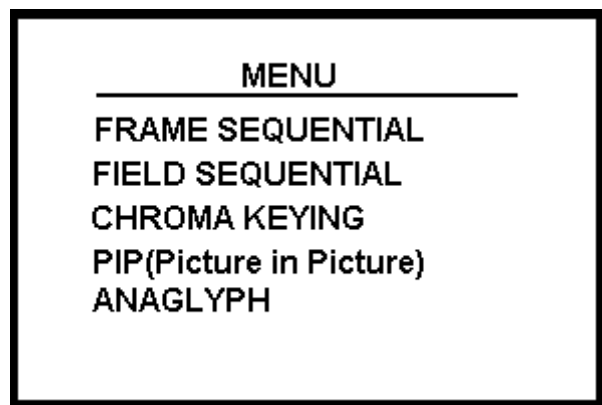
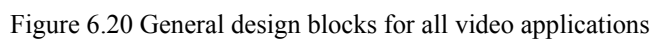


Figure 6.19 Shape of the menu screen

Cyclone II memory blocks support ROM mode. A memory initialization file (.mif extension) specifies the initial condition of these memory blocks. Initial values for each address in the memory are assigned during the project compilation. A MIF (memory initialization file) contains these initial values. In addition to initial values, memory depth and memory width are required in this file. Also, radices can be specified.



An example of MIF file:

```
DEPTH = 32; -- The size of data in bits
WIDTH = 8;  -- The size of memory in words
ADDRESS_RADIX = HEX; -- The radix for address values
DATA_RADIX = BIN; -- The radix for data values
CONTENT -- start of (address : data pairs)
BEGIN
00 : 00000000; -- memory address : data
01 : 00000001;
02 : 00000010;
03 : 00000011;
04 : 00000100;
05 : 00000101;
06 : 00000110;
07 : 00000111;
08 : 00001000;
09 : 00001001;
0A : 00001010;
0B : 00001011;
0C : 00001100;
END;
```

In order to differentiate the menu items, menu background and menu frame, a colored menu is drawn. While initializing the menu different colors are coded as different values from 0 to 7. For each pixel these coded values are stored. Because of these values, the width of the word should be at least 3-bit. Also the depth of the required memory is 60000 (300x200).

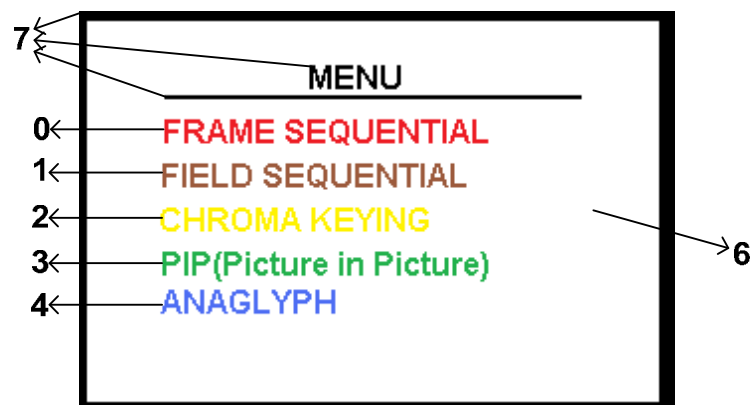


Figure 6.21 Values of the menu items in memory

In order to create this memory initialization file automatically, a MATLAB program is developed. In this way, when any change on menu items is performed, it is easy to create a new memory initialization file by using this program.

Here is the MATLAB code:

```
fid = fopen('mask.mif','w');
img = imread('test_mask.bmp');
y=rgb2gray(img);
figure,imshow(y)
img = 6*ones(200,300);%% Size of Menu screen (width:300 height:200)
img(y==255)=6; % Menu Background
img(y==0) = 7; % Menu Heading and menu frame
img(y==91) = 0; % Menu item 1 --> Frame Sequential
img(y==106)=1; % Menu item 2 --> Field Sequential
img(y==218)=2; % Menu item 3 --> Chroma Keying
img(y==123)=3; % Menu Item 4 --> PIP
img(y==115)=4; % Menu Item 5 --> Anaglyph

[row col] = size(img);
% Mif Configuration Parameters
WIDTH=3; % number of bits of data per word
DEPTH= row * col; % number of addresses

fprintf(fid,'WIDTH=%d;\n',WIDTH);
fprintf(fid,'DEPTH=%d;\n',DEPTH);
fprintf(fid,'ADDRESS_RADIX=UNS;\n');
fprintf(fid,'DATA_RADIX=UNS;\n');
fprintf(fid,'CONTENT BEGIN\n');

address = 0;
data = 0;
X = 1;
Y = 1;
for i = 0:(DEPTH-1)
    data = (img(X,Y));
    Y = Y + 1;
    if ( Y == (col+1) )
        Y = 1;
        X = X + 1;
    end
    fprintf(fid,'%10d : %d;\n',i,data);
end

fprintf(fid,'END;\n');
fclose(fid);
```

A megafunction is created for the memory initialization file. General settings of this megafunction which is a memory block in ROM mode are as in the Figure 6.22. Memory initialization file created by the MATLAB program is given as input to this

megafunction. While compiling the design the MIF is read and initialization values are assigned to the memory while programming the FPGA.

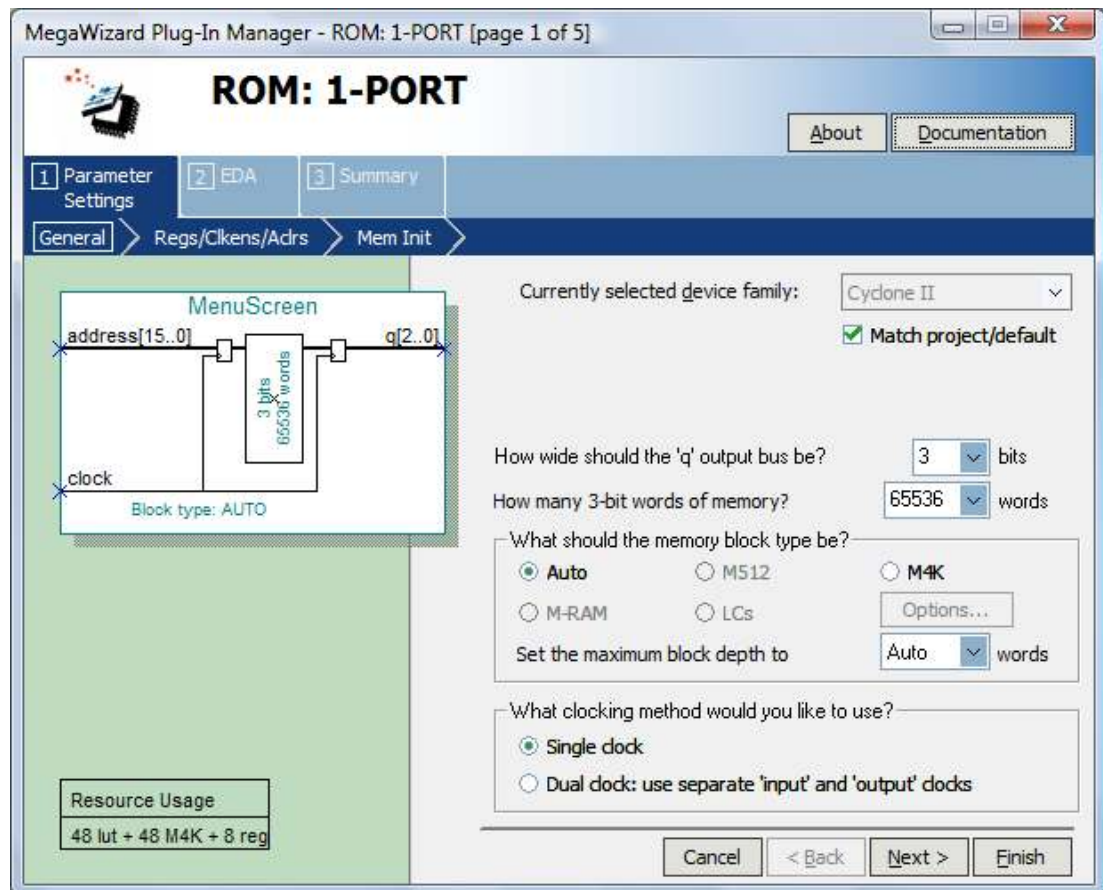


Figure 6.22 General settings for the function of memory in ROM mode

The output video resolution is 640x480 and the menu size is 300x200 pixel. The menu screen is located to the center of the video. As seen from the Figure 6.23, when the $170 \leq VGA_X < 470$ and $141 \leq VGA_Y \leq 340$, the memory block is read and the menu screen pixels are displayed at between these row and column intervals.

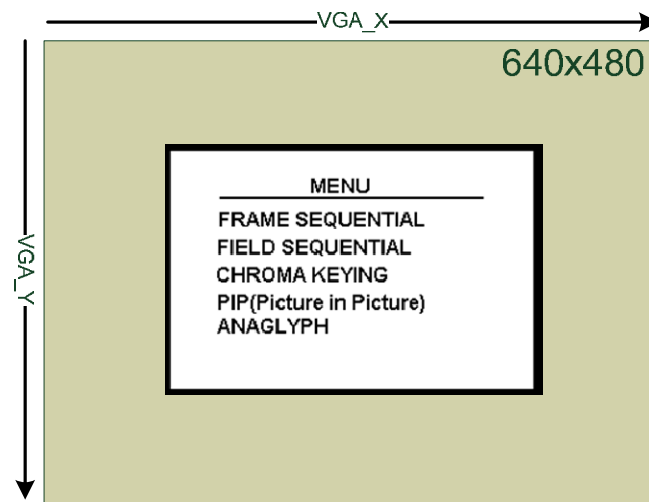


Figure 6.23 Position of the menu screen

In menu design, there are three colors for the background. Black is used for the menu background, green is used for the menu items and menu frame, purple is used for the selected menu item. By using the 4 push-buttons on DE2-70 board, the required video application can be performed. *KEY3* is used to apply the selected application at that time. Also when any application is being performed, *KEY3* is used to return back to the menu screen. *KEY2* is used to move the selected item to the upper menu item. *KEY1* is used to move the selected item to the lower menu item. *KEY0* is used to reset the system.

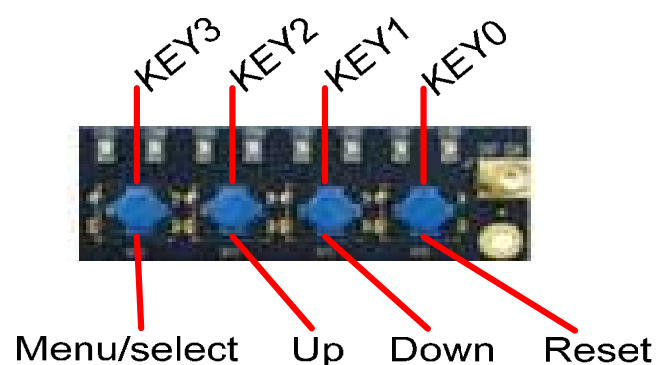


Figure 6.24 Function of push-buttons for menu usage

A counter in the design holds the number of selected menu item, according to this selected value counter is changing. The real appearance of the menu screen is

illustrated in Figure 6.25. As it is seen, menu background looks like transparent. In order to make the transparency effect, menu background color is displayed only on the odd lines. In the even lines the *video source1* is displayed.



Figure 6.25 Menu interface at the output of the FPGA

6.8 Project Design Report

The Figure 6.26 indicates the compilation report of the project designed in Quartus II Web Edition.

Flow Summary	
Flow Status	Successful - Wed Aug 31 12:24:48 2011
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	FPGA_TOP
Top-level Entity Name	FPGA_TOP
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	3,412 / 68,416 (5 %)
Total combinational functions	2,894 / 68,416 (4 %)
Dedicated logic registers	1,956 / 68,416 (3 %)
Total registers	1988
Total pins	530 / 622 (85 %)
Total virtual pins	0
Total memory bits	827,264 / 1,152,000 (72 %)
Embedded Multiplier 9-bit elements	36 / 300 (12 %)
Total PLLs	1 / 4 (25 %)

Figure 6.26 Compilation report of the final design

CHAPTER SEVEN

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this thesis, real time video applications are studied and implemented in FPGA. The detailed information about the video background and programmable logic devices are presented and the main parallel video applications are introduced briefly.

Firstly, an FPGA development board is investigated in the market to meet parallel video processing requirements with two video input and one video output. Altera DE2-70 development board is preferred which is the best for parallel video applications in terms of performance and cost effectiveness.

The implemented video applications are frame sequential video, field sequential video, chroma keying application, anaglyph video and picture in picture video. They are both implemented and tested in MATLAB and FPGA environments. The applications are implemented in Cyclone II (EP2C70) FPGA on DE2-70 board. All the workload of the implementations is in hardware part. Therefore the processor NIOS II in Cyclone II device is not used in this thesis.

The first approach during the design period is implementation of the video editing applications using only one video source. TV Decoder IC is used to convert the analog video input to digital video format. Also the system clock 27MHz is generated from the TV decoder IC. SDRAM is used for video frame buffering and VGA DAC IC is used to convert the digital video to analog video.

After a great deal of knowledge is gained on video blocks for one video source, implemented video blocks are duplicated for the second video source. The second TV Decoder IC and the second SDRAM is used with the second video source. Frame sequential video, field sequential video, chroma keying, anaglyph video and picture in picture applications are realized one by one. The implemented methods are tested

with two video input sources which are in NTSC video format. Frame sequential application is tested by using a head mounted display (HMD). Chroma keying application is tested with a foreground video of which backdrop color is green. Anaglyph video is tested with red/cyan colored glasses. Field sequential video and picture in picture video are just observed in VGA monitor.

Finally, all the designs are combined into one implementation and a user friendly menu interface is designed and implemented in FPGA. By using the push-buttons, the user can select the required video application on menu.

The major advantage of the FPGA in this study is the parallel processing features in real time video processing. Two video can be processed easily in parallel video blocks at the same time. FPGAs are good for large number of image processing applications. In this project, video applications do not include complex mathematical expressions. In mathematical operations, DSP chips are better than FPGAs and designers should prefer a system consisting of both FPGAs and DSPs for these complex designs. The implemented design is not PC based. Thus, it is a low-cost and convenient platform for parallel video applications.

7.2 Future Work

In our design, the video inputs have to be in analog format because DE2-70 board has composite analog video inputs. In order to process video in FPGA, it should be converted to the digital format. Analog to digital conversion, de-interlacing and chroma resampling operations decrease the video quality. As a future work, these implementations can be performed on a different FPGA board which has digital video inputs instead of analog video inputs.

In this study, the refresh rate of the output video is 60Hz. However, minimum required refresh rate is 85Hz for a frame sequential video in 3D mode. As a future work, refresh rate can be increased to 85Hz or higher refresh rates to prevent the flickering effects.

REFERENCES

ADV7180 Datasheet. (2006) *Analog Devices*. Rev. B.

Altera (2007). *Cyclone II Device Handbook*, Volume 1, Chapter 2 (ver 3.1).

Best 3DTVs. (n.d.). *How 3D TVs work*. Retrieved September 12, 2011 from <http://www.best-3dtvs.com/guides/how-3d-tvs-work/>

Brown S. & Rose J. (n.d.). *Architecture of FPGAs and CPLDs: A Tutorial*. Department of Electrical and Computer Engineering, University of Toronto

Color Vision: One of Nature's Wonders. (n.d.). Retrieved September 12, 2011 from <http://www.diycalculator.com/sp-cvision.shtml#A1>

Digital Creation Labs Incorporated. (April, 2004). *Digital Video Overview* (Rev 1.0). Retrieved September 12, 2011 from http://www.digitalcreationlabs.com/docs/AN10_digital_video_overview.pdf

Dumbreck A.A. & Smith C.W. (1991, 1992). *3-D TV Displays for Industrial Applications*. Copyright AEA Technology 1991, 1992

Dubois E. (2000). *Generation of Anaglyph Stereoscopic Images*.

Dubois E. (n.d.). *A Projection Method to Generate Anaglyph Stereo Images*. School of Information Technology and Engineering, University of Ottawa

Engdahl T. (1997). *3D Glasses and Other 3D Display Devices*. Retrieved September 9, 2011 from <http://www.epanorama.net/documents/pc/3dglass.html>

Gallagher A. (2010). Detecting Anaglyph Images with Channel Alignment Features. *Proceedings of 2010 IEEE 17th International Conference on Image Processing*, September 26-29, 2010, Hong Kong.

- HMD User Manual (2004). *5DT HMD 800-26 Series User's Manual*. Fifth Dimension Technologies. Version 3.0, August, 2004.
- Hsiao H. H. & Jeng J. H. (n.d.). *Modified De-interlacing method based on edge direction*. Department of Information Engineering, I-Shou University, Taiwan
- Huh K.M. (1999). *The Two-Eyes Position Detection Algorithm for a Moving Viewer Using Color Camera Image*. Proceeding of the 1999 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Taipei, Taiwan, R.O.C. August 1999.
- Intersil Americas Inc. (July, 2002). *BT.656 Video Interface for ICs*. Application Note 9728.2
- Jack K. (2007). *Video Demystified* (Fifth Edition). United States of America: Elsevier Press
- Jackson C. (n.d.). *Flash + After Effects*. Focal Press
- McAllister D. (n.d). *Display Technology: Stereo & 3D Display Technologies*
- Neoh H. S. & Hazanchuk A. (n.d.). *Adaptive Edge Detection for Real-Time Video Processing using FPGAs*. ALTERA.
- Pellerin D. (2009). *Implementing Video Analysis in Xilinx FPGAs*. Embedded magazine, April 2009.
- Smith D. J. (n.d.). *VHDL & Verilog Compared & Contrasted – Plus Modeled Example Written in VHDL, Verilog and C*. BeriBest Incorporated.
- StereoGraphics Corporation. (1997). *The StereoGraphics Developers' Handbook*

Terasic (2009). *DE2-70 User Manual Version 1.08*, Terasic Technologies.

Wimmer P. (2005). *Anaglyph Methods Comparison*. Retrieved September 12, 2011 from http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx

Wood A. (n.d.). *Use Chroma Key Techniques to Capture Exquisite Images*. Retrieved September 12, 2011 from <http://ezinearticles.com/?Use-Chroma-Key-Techniques-to-Capture-Exquisite-Images&id=3543412>

WorldViz LLC. (n.d.). *Frame Sequential DDC/Blue-Line Coded Stereo*. Retrieved September 12, 2011 from http://docs.worldviz.com/vizard/Frame_sequential.htm

Xilinx (2006). *Programmable Logic Handbook*. Xilinx Technologies.

Zeidman B. (n.d.). *Introduction to CPLD and FPGA Design*. President of The Chalkboard Network.

Zhu Y. Zhen T. (2009). *3D Multi-View Autostereoscopic Display and Its Key Technologie*. 2009 Asia-Pacific Conference on Information Processing