FPGA implementation for real-time Chroma-key effect using Coarse and Fine Filter

Conference Paper · January 2013

DOI: 10.1109/ComManTel.2013.6482383

2 authors:

Sang Nguyen
University of Information Technology, VNU-HCM
7 PUBLICATIONS 5 CITATIONS
SEE PROFILE

Vinh Truong Quang
Ho Chi Minh City University of Technology (HCMUT)
26 PUBLICATIONS 77 CITATIONS
SEE PROFILE

FPGA Implementation for Real-time Chroma-key Effect Using Coarse and Fine Filter

Nguyen Thanh Sang
Faculty of Electronic and Telecommunication
Ho Chi Minh City University of Science,
Vietnam National University - Ho Chi Minh City
Ho Chi Minh city, Viet Nam
E-mail: nguyenthanhsang85@gmail.com

Truong Quang Vinh
Faculty of Electrical and Electronics Engineering
Ho Chi Minh City University of Technology
Vietnam National University - Ho Chi Minh City
Ho Chi Minh city, Viet Nam
E-mail: tqvinh@hcmut.edu.vn

Abstract— **Chroma-key is a robust and important technique for processing image or video that is widely used in cinema films, magazine covers, video game industries as well as television programs such as weather forecast, live talk show, ect. This paper presents study of Chroma-key method and proposes a hardware architecture for Chroma-key effect in real-time. Based on K-means algorithm, we propose an improved method namely *Coarse and Fine Filter*, which is more robust on segmentation and more appropriate for hardware design thanks to usage of small buffer. A VLSI architecture for the proposed method is implemented on the Altera DE2 FPGA board. Experimental results show that the proposed design can perform Chroma-key effect with pleasing quality in real-time.**

Index Terms—**Chroma-key effect, K-means clustering, VLSI architecture, FPGA.**

I. INTRODUCTION

Chroma-key is a technique that a foreground object extracted from a foreground frame will be combined with a background frame to create a new composite frame for a special effect. The foreground frame has two parts consisting of foreground and background objects, in which the background object is a solid color, usually green or blue.

Fig. 1 shows an example of a foreground frame with green screen of background and a background frame. The frames are then processed by using Chroma-key technique.

Chroma-key effect is often used in films, television programs, and in the weather forecasting, especially. The weather reporter generally stands in front of a green or blue screen. Meanwhile, the green or blue background color will be replaced with a weather map when weather report is telecasting. In general, the Chroma-key tools are used widely in creation alien worlds and impossible situations to increase more special effects or save a big amount of money for the sections that can perform in studio of the movie in the modern entertainment industry.

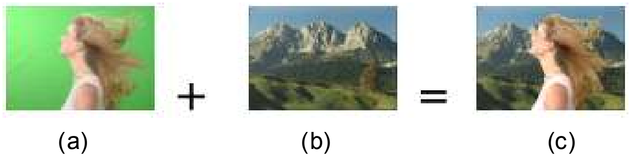


Figure 1. Illustration of the Chroma-key effect. (a) Foreground frame, (b) background frame, (c) composite frame.

The Chroma-key effect can be done in post-production or in real-time. In post-production, one is used almost by software [5, 7] and in real-time, one is commonly used as a Chroma-key device to connect to or include in a camera [4, 8, 10, 11, 14].

The Chroma-key effect includes a part of science and a part of art. Thus, in order to achieve the best result, techniques in studio and in processing Chroma-key effect by software or device should be combined together.

This paper introduces a Chroma-key core designed and implemented on FPGA to perform the Chroma-key effect in real-time.

The content of the paper consists of six sections. Section I introduces the general problem. Section II presents some reference methods which can perform the Chroma-key effect on the Matlab software. In section III, an improved method which is intended for the hardware solution in real-time is presented. Then, a proposed VLSI architecture for the Chroma-key effect in real-time is described in section IV. Section V shows the experimental results obtained from the implementation of the design on the Altera DE2 board. The final section, the conclusion of this paper is shown in section VI.

II. STUDY OF CHROMA-KEY METHODS

A common chroma-key method is to find a mask representing for the foreground region extracted from foreground frame, in which the pixels belong to foreground object will be equal to 1, and ones belong to background object will be equal to 0. Therefore, the mask will show up two colors, white and black. The values that are equal to 1 are white, and ones that are equal to 0 are black.

After the mask has been obtained, the Chroma-key effect is performed on foreground and background frames with the mask as below equation:

$$C = Fg.{*}mask + Bg.{*}(\sim mask) \qquad (1)$$

where C is composite frame, *mask* is the mask of foreground frame, *Fg* and *Bg* are value of all pixels in the *RGB* color space of the foreground and background frame, respectively.

Some methods that can perform the Chroma-key effect are referred and presented as follows:

A. *Fixed Key Method*

This method is demonstrated by Thilina Sammera [16]. He proposed a simple method to find out a mask of the foreground frame. However, the use of the method to the Chroma-key process has to need a *key* value first. A manipulation scans overall values of the red (256 values), green (256 values), blue (256 values) components in each pixel of foreground frame to reach the *key* value. It takes many times to do that, so he has built a GUI (Graphic User Interface) to help finding the key value easily.

This method has proposed a simple equation to find out the mask of foreground frame. But if this one is used, finding a *key* value must be performed first by scanning over *256x256x256=16.777.216* values. It is hence very time-consuming and difficult in the finding key exactly; therefore, this method is not chosen to implement by hardware. The simulation result of this method is seen in Fig. 7(b) with *key*=[75 170 80] for green background object and *key*=[130 172 80] for blue background object.

B. *K-Means Clustering Method*

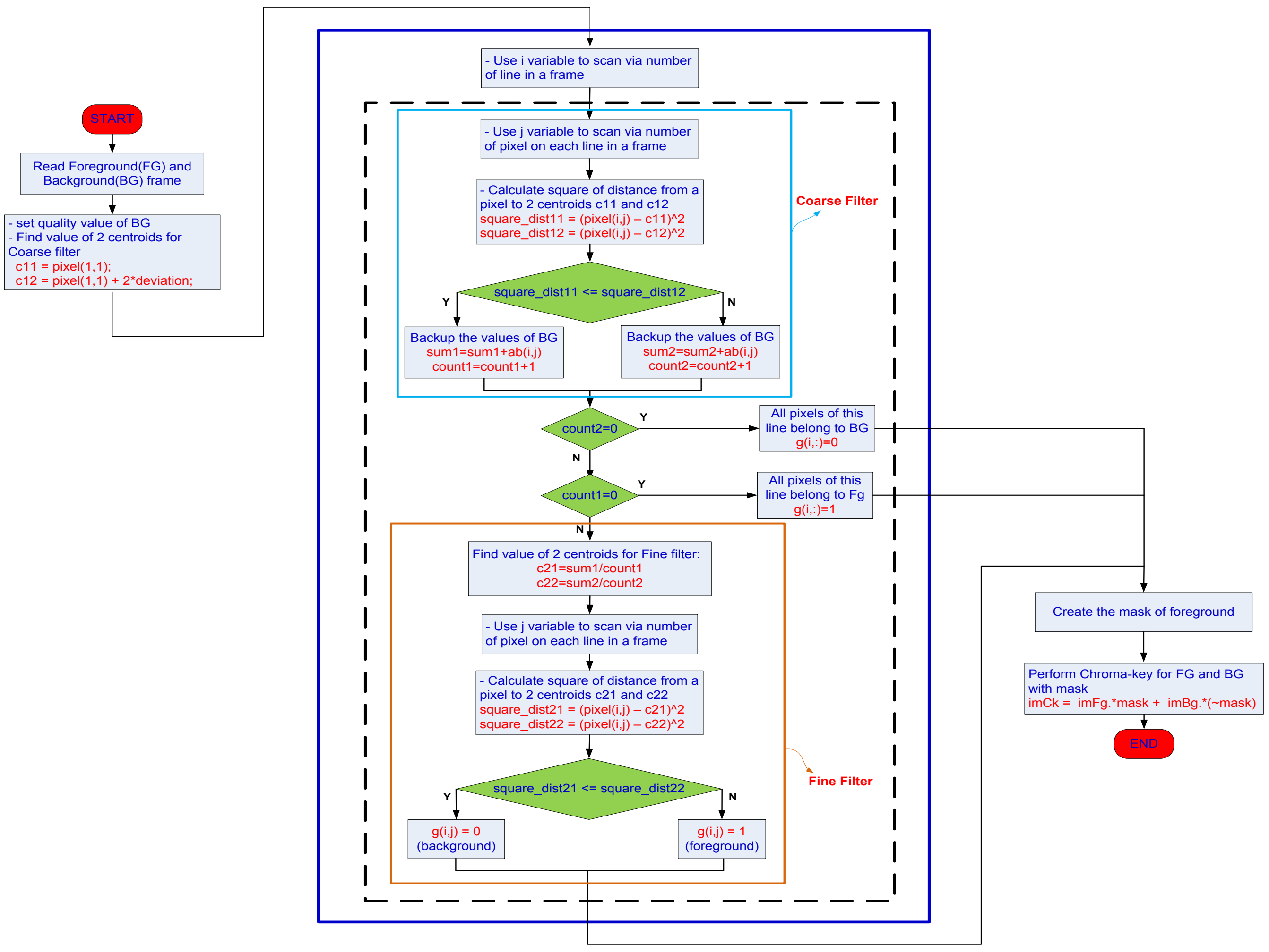
In order to find a mask for Chroma-key method, key-value based methods cannot provide pleasing result due to the noise of the solid color in background object. A clustering method must be used to find the mask for Chroma-key method. Since the number of cluster is predefined as two, K-Means, a simple and effective clustering algorithm is an appropriate choice.

In the Matlab software, a K-Means function is ready and how to use this function to classify color clustering is guided at an example in [18]. In that example, the image is converted from *RGB* to *L*a*b** color space, *a** and b* components are used to find the mask of foreground, and then create composite frame. Chroma-key effect performed by K-Means clustering has fairly good result, as shown in Fig. 7(c).

Kardi Teknomo proposed a modified K-Means function for the Chroma-key effect which uses some iterations before the clustering has reached stability [17]. The simulation result on Matlab is shown in Fig. 7(d). This method has improved the performance of clustering process. However, the number of iterations in each different foreground frames is large, about eight or nine iterations in the testing foreground frames. Thus, the required buffer for the algorithm is extremely large. Moreover, the number of iterations is not constant for every different foreground frames; hence, the size of buffer needed for hardware solution is not fixed.

III. Proposed Chroma-Key Method

The required buffer for the K-Means Clustering method is very large, and it is a really hard constraint for real-time



START
Read Foreground(FG) and Background(BG) frame
- set quality value of BG
- Find value of 2 centroids for Coarse filter
c11 = pixel(1,1);
c12 = pixel(1,1) + 2*deviation;
- Use i variable to scan via number of line in a frame
- Use j variable to scan via number of pixel on each line in a frame
- Calculate square of distance from a pixel to 2 centroids c11 and c12
square_dist11 = (pixel(i,j) – c11)^2
square_dist12 = (pixel(i,j) – c12)^2
square_dist11 <= square_dist12
Y
N
Backup the values of BG
sum1=sum1+ab(i,j)
count1=count1+1
Backup the values of BG
sum2=sum2+ab(i,j)
count2=count2+1
Coarse Filter
count2=0
Y
All pixels of this line belong to BG
g(i,:)=0
N
count1=0
Y
All pixels of this line belong to Fg
g(i,:)=1
N
Find value of 2 centroids for Fine filter:
c21=sum1/count1
c22=sum2/count2
- Use j variable to scan via number of pixel on each line in a frame
- Calculate square of distance from a pixel to 2 centroids c21 and c22
square_dist21 = (pixel(i,j) – c21)^2
square_dist22 = (pixel(i,j) – c22)^2
square_dist21 <= square_dist22
Y
N
g(i,j) = 0
(background)
g(i,j) = 1
(foreground)
Fine Filter
Create the mask of foreground
Perform Chroma-key for FG and BG with mask
imCk = imFg.*mask + imBg.*(~mask)
END

Figure 2. Flowchart of the Coarse and Fine Filter method.

hardware implementation. A requirement of decreasing the buffer size is considered.

On the other hand, in the weather forecasting programs or movies that use the Chroma-key effect will have that the first line of the composite frame always includes all the pixels of background frame.

The main idea of this method is to improve the K-Means function modified by Kardi Teknomo. In the proposed method, the number of iterations is fixed on two, coarse and fine filter processes, and the iteration on each frame is replaced by on each line of frame. So, the size of buffer will be decreased considerably from a frame size to a line size of a frame in hardware solution. For example, a frame with *640x480* sizes, the size of buffer is just equal *640*.

In addition, the clustering just done for all pixels on each line of a frame instead of all pixels of a frame will execute more accurately. A value that uses to set the permissible minimum deviation between foreground and background object in foreground frame is applied to the coarse filter process as input value for the setting two centroids differently in the first iteration to help this filter perform better. The fine filter updates values of the two centroids and applies clustering process based on Euclidean distances from those centroids.

The flow chart of this method is seen in Fig. 2, and the simulation result of this one on Matlab software shown in Fig. 7(e)(f) is positive and encouraging.

The most important things is that this method with the advantages such as small buffer size, high accuracy in the clustering is to appropriate for implementing the Chroma-key effect by hardware in real-time.

IV. VLSI ARCHITECTURE FOR CHROMA-KEY EFFECT IN REAL-TIME

This section presents a VLSI architecture that can perform the Chroma-key effect in real-time on a FPGA chip by using *Coarse and Fine Filter* method. The FPGA chip used here is the Cyclone II chip of Altera Corporation integrated on the Altera DE2 board. In implementing process, the Chroma-key core will be designed and embedded in the TV Box module to perform the Chroma-key effect.

A. TV Box Module

In the design process, finding a system which can demonstrate the implementation of the design in real-time is concerned. A TV Box module developed by the Altera Corporation for studying about the multimedia technologies of students or engineers is selected. The performance of the TV Box module is efficiently demonstrated on the Altera DE1 and DE2 boards.

One of the most important reasons for using this module is that its specification and Verilog code are released free to all users by Altera Corp. It is thus easy to understand the activity of all blocks of TV Box module, and suitable to study and design a new module that can be integrated in this module to perform a special function or visional effect as the Chroma-key effect.

Fig. 3 illustrates a proposed system to perform the Chroma-key effect on the Altera DE2 board. The input video signal is retrieved from a camera, and then it is transferred to the Altera DE2 board to perform the Chroma-key effect by a Chroma-key chip, which involves the designed Chroma-key core embedded in the ready TV Box module. After the Chroma-key process, the output video signal is displayed on a VGA (LCD/CRT) monitor. The block diagram of the Chroma-key chip is shown in Fig. 4 in detail.

*B. L*a*b*, RGB and YCbCr Color Space*

All foreground frames utilized in the simulation on Matlab software are the *RGB* color space, then they are converted into the *L*a*b** color space (except the Fixed Key method) to find out the mask. The *a** component is used for green color, and the *b** component is used for blue color of background object in the foreground frame.

Applying the *Coarse and Fine Filter* method for performing the Chroma-key effect is that the change of color space from *RGB* to *L*a*b** should be done first.

Since there is no function that converts directly *RGB* into *L*a*b** color space, this converting must be realized through two the following steps:

- *RGB* into *XYZ* color space
- *XYZ* into *L*a*b** color space

The equation to convert *RGB* into *XYZ* and *XYZ* into *L*a*b** color space are very complicated and difficult for hardware implementation. Therefore, the *Cb* and *Cr* components of the *YCbCr* color space are used to implement the Chroma-key effect in real-time [9, 10].

Besides, inside the TV Box module, there are the *YUV442_to_444* and *YCbCr_to_RGB* blocks to perform the conversion *YUV_442* into *YUV_444 (YCbCr)* and *YCbCr* into *RGB* color space, respectively. So, the *YCbCr* color space is studied whether it can be used to find mask of the foreground frame in the Chroma-key process or not.

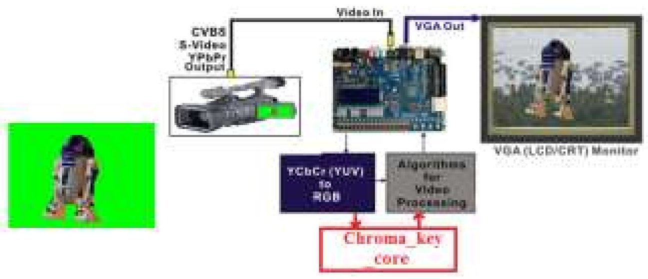


Figure 3. A proposed system for implementing Chroma-key effect in real-time.

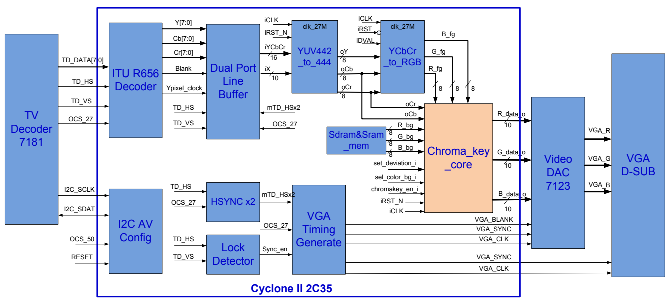


Figure 4. Block diagram of the Chroma-key chip in the FPGA CycloneII2C35 chip on Altera DE2 board.

The formula converted the RGB into YCbCr color space is shown in Table I.

TABLE I. A FORMULA CONVERT RGB INTO YCBCR COLOR SPACE [26]

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16$$
$$Cr = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128$$
$$Cb = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128$$

Some features of the foreground frame can be used to implement the Chroma-key effect:

- In the foreground frame whose background object is green color, the red and blue intensities of the pixels in background region are very small, while the green intensity of those pixels is very high. Ideally, R:G:B value of these pixels are 0:255:0, respectively. Thereby, according to the formula in Table I., *Cr* component of these pixels will be much less than ones that belong to the foreground object region. Therefore, the *Cr* component should be used to apply K-Means Clustering combined with the *Coarse and Fine Filter* method.

- In the foreground frame whose background object is blue color, the red and green intensities of the pixels in the background region are very small, while the blue intensity of those pixels is very high. Ideally, R:G:B value of these pixels are 0:0:255, respectively). Hence, according to the formula in Table I., *Cb* component of these pixels will be much greater than ones that belong to the foreground object region. Therefore, the *Cb* component should be noticed. In this case, in order to apply properly K-Means Clustering combined with the *Coarse and Fine Filter* method, the (*255 - Cb)* value should be used.

The implementation of the Chroma-key effect in *YCbCr* color space with the *Coarse and Fine Filter* method is done and the simulation result on Matlab is shown in Fig. 7(g)(h).

C. *Architecture of the Chroma_key Core*

The Chroma_key core is designed and embedded in the TV Box module to create the Chroma-key chip. The *Coarse and Fine Filter* method is applied to this design.

1) Block Diagram: The Chroma-key core consists of five blocks as shown in Fig.5.

a) Coarse_filter block: This is one of two main blocks of the core. Its function is to perform coarsely the clustering from the first two centroids, one is the *Cr* or *Cb* value of the first pixel of the foreground frame, based on the background object that is green or blue; and another is the *Cr* or *Cb* value of the first pixel plus amount of *2*set_deviation_i*. The *sum1, count1, sum2, count2* values created by this block will be transfered to the *Fine_filter* block to calculate for two new centroids of the fine filter process in next iteration. Besides, this block also creates signals to control the FIFOs of the other blocks such as *rd_fifo, wr_fifo* and *reset_fifo.*

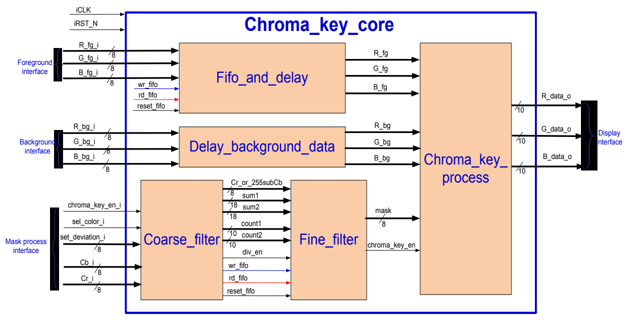


Figure 5. Block diagram of the Chroma-key

b) Fine_filter block: This is another main blocks of the core. Depending on the *sum1, count1, sum2* and *count2* values that get from the *Coarse_filter* block, two centroids for the fine filter process will be computed after the *div_en* signal is active.

The value of two centroids will obtain after the two cycles through a divider inside this block. The clustering realized with two new centroids and the *Cr_or_255subCb* value of the *Coarse_filter* block will create a *mask* of the foreground frame, and then the *mask* will be transfered to the *Chroma_key_process* block to perform the Chroma-key effect.

c) Fifo_and_delay block: This block consists of three FIFOs to store red, green, blue components of each pixel in the foreground frame. The values that read out from the FIFOs will be delayed one cycle to synchronize with the *mask* of the *Fine_filter* block. The values after delay will be transfered to the *Chroma_key_process* block to perform the Chroma-key effect.

d) Delay_background_data block: This block performs the delay on the background frame data which read from memory device to synchronize with the foreground frame data of the *Fifo_and_delay* block and the *mask* of the *Fine_filter* block. The output of this block will be also transfered to the *Chroma_key_process* block to perform the Chroma-key effect.

e) Chroma_key_process block: Depending on the foreground frame data, background frame data and the *mask* value, this block will perform the Chroma-key process.

2) Pipeline Structure: The specification of the *Chroma_key* core *is* shown in detail in Fig. 6 (the part is surrounded by the red line). The *Coarse and Fine Filter* method is applied to this design; hence, the two most important blocks of this core, *Coarse_filter* and *Fine_filter* blocks, are corresponding to two main processes in the hardware design.

In the first process, through studying the performance of the *YCbCr_to_RGB* block in the TV Box module, this block will take to 5 clock cycles. Since the *Cr* and *Cb* components are input both *Coarse_filter* and *YCbCr_to_RGB* blocks, and the *Cr_or_255subCb* value (output of the *Coarse_filter* block) need to be synchronous with the *R, G, B* (outputs of the *YCbCr_to_RGB* block), the *Coarse_filter* block is designed as a five-stage pipeline structure.

In the second process, the *Fine_filter* block is designed as a three-stage pipeline structure, and then the *Chroma_key_process* block spends one clock cycle; therefore, this process will be done in four clock cycles.

V. EXPERIMENTAL RESULTS

A demonstration system has been set up for testing the implementation of design in real-time on the Altera DE2 board as shown in Fig. 8. This system consists of a camera used to record a video that its each frame is a foreground frame in Chroma-key process, a Altera DE2 board with the FPGA Cyclone-II chip, a monitor used to display the video result after the Chroma-key effect is implemented, a green or blue cardboard utilized to create the background object of the foreground frame, and a computer (laptop) used to download the Chroma-key program to the Altera DE2 board. Besides, a person also need to work as a foreground object of the foreground frame.

The experimental results shown in Fig. 9(a)(b) are encouraging and satisfactory despite using a demonstration system in which a cardboard is not enough qualification as desired background for the Chroma-key process system. Especially, since the lighting condition is not perfect, it still remains the shadow of the foreground object on the background object. Thus, the noises still exist at the shadow places in the implementation results of this system.

In general, the proposed design can perform the Chroma-key effect in real-time properly, and the experimental results are fairly good and satisfactory. They can be better and higher quality if the design is implemented in a professional studio or television system.

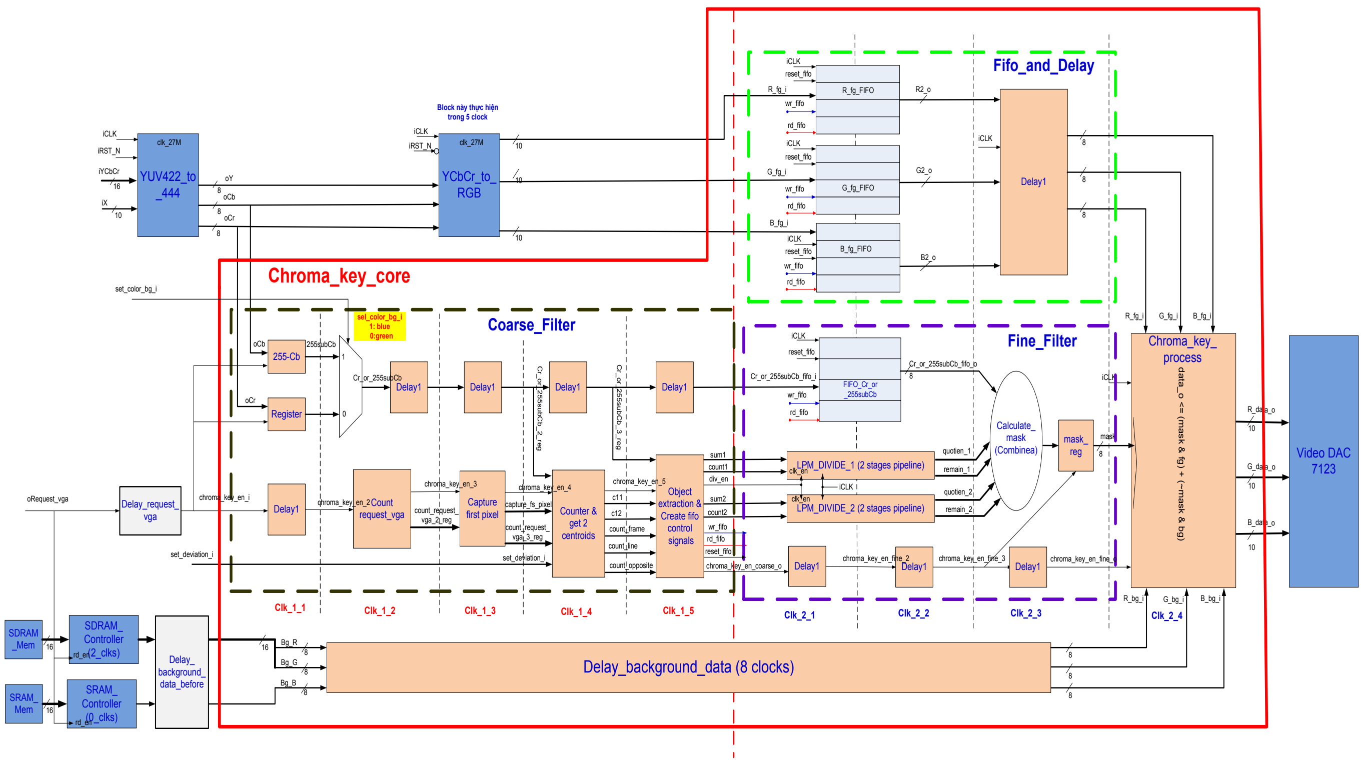

Chroma_key_core
Fifo_and_Delay
Coarse_Filter
Fine_Filter
YUV422_to_444
YCbCr_to_RGB
R_fg_FIFO
G_fg_FIFO
B_fg_FIFO
Delay1
255-Cb
Register
Delay_request_vga
Count request_vga
Capture first pixel
Counter & get 2 centroids
Object extraction & Create fifo control signals
FIFO_Cr_or_255subCb
LPM_DIVIDE_1 (2 stages pipeline)
LPM_DIVIDE_2 (2 stages pipeline)
Calculate_mask (Combinea)
mask_reg
Chroma_key_process
data_o <= (mask & fg) + (~mask & bg)
Video DAC 7123
SDRAM_Mem
SDRAM_Controller (2_clks)
SRAM_Mem
SRAM_Controller (0_clks)
Delay_background_data_before
Delay_background_data (8 clocks)
Block này thực hiện trong 5 clock
sel_color_bg_i 1: blue 0:green
Clk_1_1
Clk_1_2
Clk_1_3
Clk_1_4
Clk_1_5
Clk_2_1
Clk_2_2
Clk_2_3
Clk_2_4

Figure 6. The pipeline structure of the Chroma_key_core.

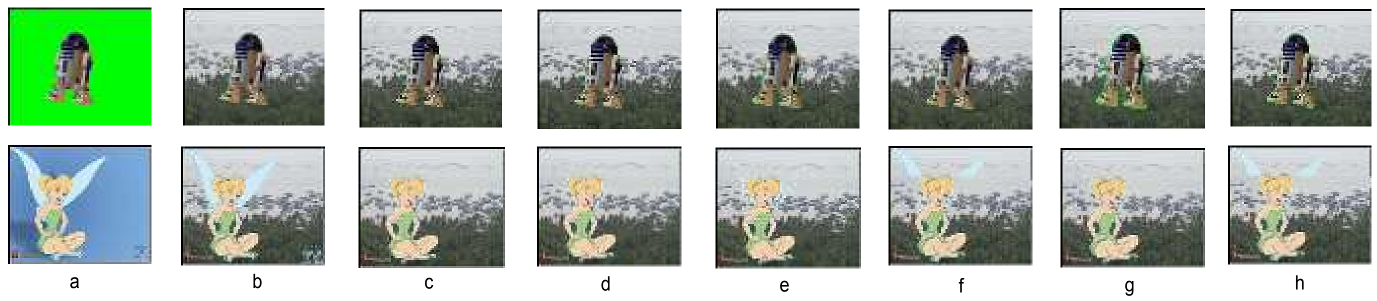


Figure 7. Simulation results on Matlab software. (a) foreground frames, (b) composite frames of the fixed key method, (c) composite frames of the method using K-Means clustering function in Matlab software, (d) composite frames of the method using K-Means clustering function modified by KardiTeknomo, (e) and (f) composite frames after coarse filter and fine filter process of the Coarse and Fine Filter method in L*a*b* color space, (g) and (h) composite frames after coarse filter and fine filter process of the Coarse and Fine Filter method in YCbCr color space.

VI. Conclusion

In this paper, we have proposed Coarse and Fine Filter method the Chroma-key effect. We also have proposed a VLSI architecture for Chroma-key effect in real-time. The whole design is separated in two main processes, the coarse filter and the fine filter processes, in which each process is designed with pipeline structure. And then, the design is combined with the TV Box module, developed by the Altera Corporation to implement the Chroma-key effect in real-time on the Altera DE2 board. A demonstration system is also set up to show the performance of the proposed design in real-time. The experimental results show that the quality of the output composite frames by the proposed system is better than Fixed key and K-mean methods.

The results of this paper are initial achievements in order to propose a Chroma-key IP core, which can be applied for the professional studios or television systems in the future.

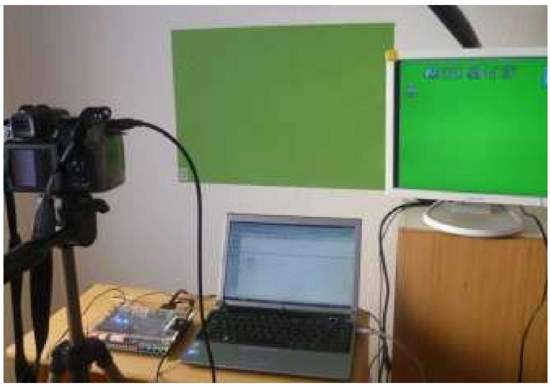
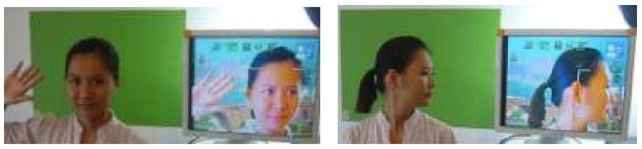
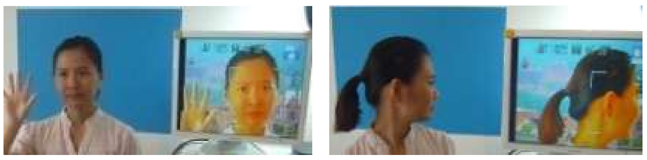


Figure 8. A demonstrated system for the Chroma-key effect in real-time.



(a)

.



(b)

Figure 9. The implementation results in real-time. (a) green background, (b) blue background.

References

[1] Alvy Ray Smith and James F. Blinn (1996), "Blue Screen Matting", *SIGGRAPH, pp. 259–268*

[2] Changick Kim, Jungwoo Park, Jaeho Lee, and Jenq-Neng Hwang (2007), "Fast Extraction of Objects of Interest from Images with Low Depth of Field", *ETRI Journal*, 29(3)

[3] Christopher Schultz (2006), *Digital Keying Methods,* Technical report, University of Bremen, Germany

[4] *David Yamnitsky (2009), Real-Time Chroma Keying on the GPU,* A White Paper by *David Yamnitsky* Boris FX.

[5] F. van den Bergh, V. Lalioti (1999), "Software Chroma Keying in an Immersive Virtual Environment", *South African Computer Journal*, pp. 155-162

[6] Gilbert A (2008), *Scalable and Adaptable Tracking of Humans in Multiple Camera Systems*, PhD. Thesis, University of Surrey Guildford, Surrey GU2 7XH, U.K.

[7] Hiroki Agata, et al., "Region Extraction Using Chroma Key with a Checker Pattern Background", Faculty of Engineering, Shizuoka University, Japan

[8] Kosmas Dimitropoulos, et al. (2010), "Improve 3D video synthesis combining graph cuts and chroma key technology", *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1-4

[9] Keith Jack (2001), *Video Demystified*, LLH Technology Publishing, United States of America

[10] Krit Panusopone, Xuemin Chen (1999), "Shape coding with a modified chroma key technique", *42nd Midwest Symposium*, pp. 422 – 425

[11] Moshe Ben-Ezra (2000), "Segmentation with Invisible Keying Signal", *Proc. Of the IEEE Conference on Computer Vision and Pattern Recognition*, *pp.* 32–37.

[12] Ronen Gvili, et al. (2003), "Depth keying", *Proceedings of SPIE-IS&T Electronic Imaging*, pp. 564-574

[13] Seoksoo Kim (2011), "Virtual Studio System for Augmented Reality & Chroma Key Processing", *Advanced Communication Technology (ICACT)*, pp. 762-765

[14] Shiaeru Shirnoda, et al. (1989), "New Chroma-key lmaging Technique with Hi-Vision Background", *IEEE Transactions on broadcasting*, 35(4), *pp. 357-361.*

[15] T. Fukaya, et al. (2002), "An effective Interaction Tool for Performance in the Virtual Studio – Invisible Light Projection System", *IBC, Amsterdam*, pp. 389-396

[16] Fixed key method, available: http://thilinasameera.wordpress.com/2010/12/03/chroma-keying-matlab-implementation-1-0/

[17] KardiTeknomo's method, available: http://people.revoledu.com/kardi/tutorial/kMean/NumericalExample.htm

[18] Available: http://www.mathworks.com/products/image/examples.html?file=/products/demos/shipping/images/ipexhistology.html