A Real Time Color Gamut Mapping Using Tetrahedral Interpolation for Digital TV Color Reproduction Enhancement

Hak-Sung Lee, Kyung-Seok Kim, and Dongil Han, *Member*, IEEE

Abstract — *In this paper, a tetrahedral interpolation technique is utilized to build new type architecture for real time color gamut mapping which has been known to be one of promising methods to enhance display quality of digital and/or high-definition TV. It is common practice to simplify the required complex computations for the mapping by adopting three-dimensional interpolation techniques. If carefully arranged, the tetrahedral interpolation can be computed with simpler operations compared to the other types of interpolation such as a cubic or three-bilinear interpolation in the conventional reduced resolution look-up table. With these simplified operations, the proposed method reduces the required computational cost and the complexity of hardware implementation. The proposed method is implemented in VHDL code and the simulation results show that the proposed method reduces the number of gates needed to hardware implementation without loss of color reproduction quality*[1].

Index Terms — Color Gamut Mapping, Display Quality Enhancement, Digital TV, Tetrahedral Interpolation.

I. INTRODUCTION

The recent technological advance in color display device entails the advent of several new kinds of display device such as LCD, PDP, ELD(Electro Luminescent Display), DLP(Digital Light Processor), LCoS(Liquid-crystal on Silicon). However, most of the newly developed display devices offer lower display quality compared to the conventional CRT display which has better quality in most features such as image quality, viewing angle, contrast ratio, brightness, durability, response time, and many other aspects. Since these shortcomings mostly come from their own color production mechanism of the newly developed display devices, in order to enhance display quality, one may resort to an approach to develop phosphor which has enhanced purity and generates brighter colors for each component or to develop color filter which has better transmissivity for each components. However, the development of new material is very difficult and often time consuming task.

[1] This work was supported by the Korea Science and Engineering Foundation under Grant R01-2003-000-10785.

Hak-Sung Lee is with the Department of Electronics Engineering, Sejong University, Seoul, Korea. (e-mail:hslee@sejong.ac.kr).

Kyung-Seok Kim is with Grandport Co., Ltd., Seoul, Korea (e-mail: kks@hanmail.net).

Dongil Han is with the Department of Computer Engineering, Sejong University, Seoul, Korea. (e-mail:dihan@sejong.ac.kr).

Contributed Paper
Manuscript received October 25, 2008

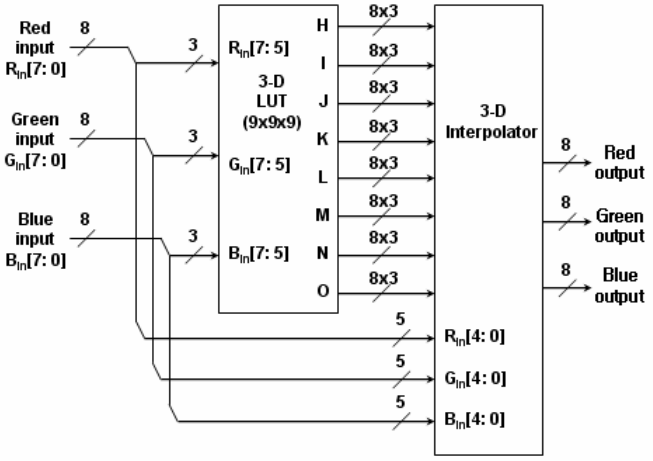


Fig. 1. The three-dimensional reduced resolution look-up table architecture

In another approach, color signal transform, such as color gamut mapping, can be adopted to enhance color reproduction quality. There are many color gamut mapping algorithms which can be used for reducing any differences that might exist between the sets of colors obtainable on different display devices[1]-[6]. However, despite of their excellence, it is very difficult to directly apply these outstanding gamut mapping algorithms to high speed display application such as digital TV where the processing time for the color transformations is limited to about several nano seconds. In order to come up with this high speed real-time requirement, Han[7] proposed the concept of three-dimensional reduced resolution look-up table(RRLT) as shown in Fig. 1.

Instead of storing all transformed color signals which require memory space of 256x256x256x3 = 50,331,648 bytes, RRLT only stores reduced number of transformed color signal with the spatial sampling over the entire input space and the output color signals out of RRLT are computed via cubic or three bilinear interpolation. This method can greatly reduce the required size of look-up table and it is possible to have reduced hardware cost as a result. For example, Han reported that only 9x9x9x3 = 2,187 bytes are sufficient to perform a color transformation for display device without severe loss of resolution[7][8].

Furthermore, by dividing the three-dimensional lookup table into 8 one-dimensional lookup tables (Fig. 2), the interpolation can be computed in parallel and this makes it possible for overall processing to meet the high speed time requirement for digital TV[7][8].

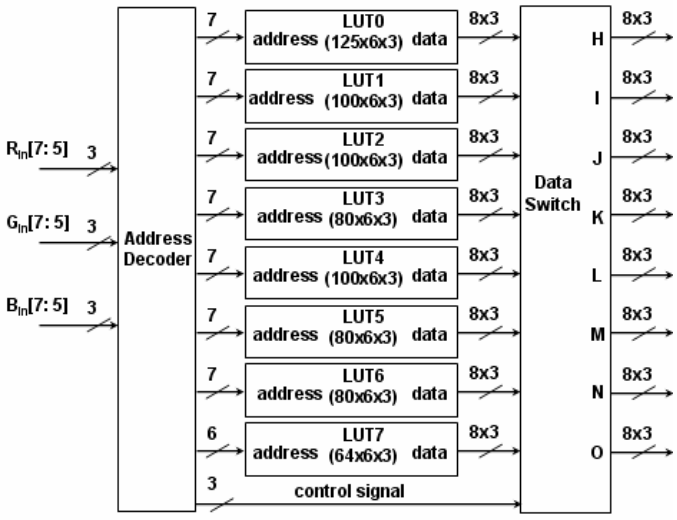


Fig. 2. The one-dimensional decomposed architecture of three dimensional lookup table.

In this paper, a tetrahedral interpolation technique will be applied to build new type real time gamut mapping for digital TV color reproduction quality. A tetrahedral interpolation is widely used interpolation method for various kinds of color space conversion. When the output signal is calculated, this interpolation uses 4 vertexes points of a tetrahedron which encloses the input point, whereas the cubic interpolation uses 8 vertex points. Therefore, if carefully arranged, the tetrahedral interpolation can be computed with simpler operations compared to the cubic interpolation. By positively using this computational simplicity, the proposed method reduces the required computational cost.

This paper is organized as follows. More detailed explanation on the tetrahedral interpolation will be given in section II. And section III will show the proposed hardware architecture for high speed gamut mapping. Simulation result and comparison with RRLT will be shown in section IV. Finally, conclusion will be mentioned in section V.

II. TETRAHEDRAL INTERPOLATION

The three-dimensional interpolation is a method to approximate a given function with three-dimensional input space using a lookup table which stores values for the function evaluated at discrete points in the input space[9][10]. When presented with a input, the output is approximately calculated by the interpolation with the input and only a few of the stored values from the lookup table.

There exist many types of interpolation techniques such as cubic interpolation[9], PRISM interpolation[11], tetrahedral interpolation[10], Kanamori interpolation[12] and so on. For example, a cubic interpolation uses the pre-calculated output points at 8 vertexes points of a cube which encloses a specific input in input space. The corresponding output to the input is calculated as a weight sum of volumes of 8 sub-cubes as follows.

$$\boldsymbol{O}_{out} = \frac{1}{V}\sum_{i=0}^{7} \boldsymbol{O}_i \times V_i \tag{1}$$

where $V = V_0 + V_1 + ... + V_7$ and V_i is the volume of the sub-cube which has the vertex directly opposite to point $\boldsymbol{I}_i$ and the input point as two diagonal vertexes with sides coincident or parallel to the sides of the original cube as shown in Fig.3. $\boldsymbol{O}_i$ is the mapping value at the point $\boldsymbol{I}_i$ such that $\boldsymbol{O}_i = f(\boldsymbol{I}_i)$ where f is the function to be approximated with the interpolation.

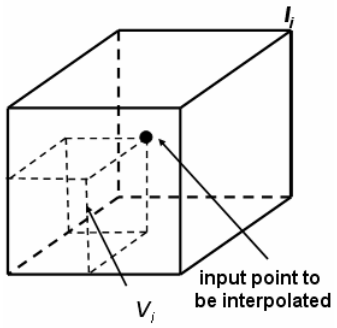


Fig. 3 Geometrical Interpretation for Cubic Interpolation

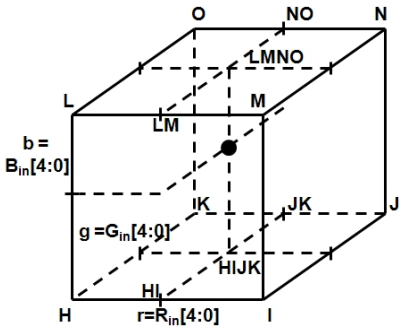


Fig. 4 Three Bi-linear Interpolations

In RRLT[7] which adopts the cubic interpolation, each R, G, B component is divided into 8 intervals. And the whole 3-dimensional color space is segmented into 8x8x8 individual cubes. The tree most significant bits of each component are used to select the corresponding cube position and the eight mapping values of each vertex of the cube. Five least significant bits of each component are used for interpolating the new mapping value for each R, G, B input image. In performing the interpolation, RRLT uses three simple bi-linear interpolations for each component. Let *r, g, b* be five least significant bits of R, G, B component, respectively. In the first step, the intermediate mapping values for R component at *HI* and *KJ* points are calculated as follows.

$$\begin{aligned} R_{HI} &= (R_H \times (32 - r) + R_I \times r)/32 \\ R_{KJ} &= (R_K \times (32 - r) + R_J \times r)/32 \end{aligned} \tag{2}$$

where R_X (X=$H,I,...,O$) is the pre-calculated mapping value for R component at X point. In the second step, we have

$$R_{HIJK} = (R_{HI} \times (32 - g) + R_{KJ} \times g) / 32 . \quad (3)$$

And finally, we can get

$$R_{out} = (R_{HIJK} \times (32 - b) + R_{LMNO} \times b) / 32 . \quad (4)$$

Whereas the cubic interpolation uses 8 vertexes, a tetrahedral interpolation[10] uses 4 vertexes of a tetrahedral which encloses the input point as illustrated in Fig.5. Let V_X, (X=A,B,C,D) be the volume of the tetrahedron defined by the input point and the face opposite to the vertex X. Then, the corresponding output is calculated as a weight sum of volumes as follows.

$$\boldsymbol{O}_{out} = \frac{1}{V}(V_A \times \boldsymbol{O}_A + V_B \times \boldsymbol{O}_B + V_C \times \boldsymbol{O}_C + V_D \times \boldsymbol{O}_D) \quad (5)$$

where V=(V_A+V_B+V_C+V_D) and $\boldsymbol{O}_X$ (X=A,B,C,D) is the mapping value for the input X. Although the computation of a volume of a tetrahedron may involve a complex calculation, the interpolation equation can be simplified since the ratio of the volumes sharing a same face is equal to the ratio of the heights. For example, V_D/V can be replaced with H_D/L_D because the tetrahedron $ACBD$ and the tetrahedron $ABCI$ share the face ABC in Fig.5. With this method, (5) can be simplified as follows.

$$\boldsymbol{O}_{out} = \frac{H_A}{L_A}\boldsymbol{O}_A + \frac{H_B}{L_B}\boldsymbol{O}_B + \frac{H_C}{L_C}\boldsymbol{O}_C + \frac{H_D}{L_D}\boldsymbol{O}_D . \quad (6)$$

In order to apply the above interpolation to the whole input space, the input space should be appropriately divided by a set of tetrahedron with a proper size. For this, as in cubic interpolation, the whole input space is segmented into cubes with same size and then each individual cube is divided into 6 tetrahedrons using the diagonal division as shown in Fig. 6. Tetrahedrons T1 through T6 share vertexes H and N. The output is interpolated from the vertexes forming the tetrahedron into which the input value falls. For example, if the input value falls in tetrahedron T1 ($g \geq b \geq r$), the output is interpolated with vertexes H, K, O and N. With tetrahedrons in Fig. 6., the interpolation equation (6) can be rewritten as follows[13].

$$\boldsymbol{O}_{out} = \frac{1}{\Delta}(H_0 \times \boldsymbol{O}_0 + H_1 \times \boldsymbol{O}_1 + H_2 \times \boldsymbol{O}_2 + H_3 \times \boldsymbol{O}_3) \quad (7)$$

where Δ is the length of one side of the original cube and H_0, ..., H_3, $\boldsymbol{O}_1$, $\boldsymbol{O}_2$ are defined in Table. I. $\boldsymbol{O}_0$ and $\boldsymbol{O}_3$ are the output values calculated at vertexes H and N, respectively, which are shared by all tetrahedrons T1 through T6.

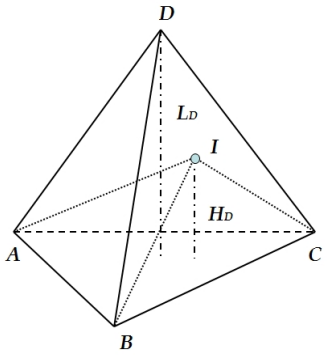
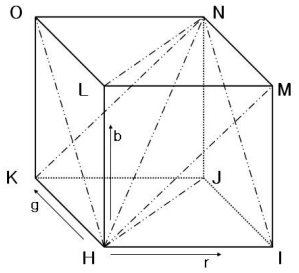


Fig. 5 Tetrahedral Interpolation



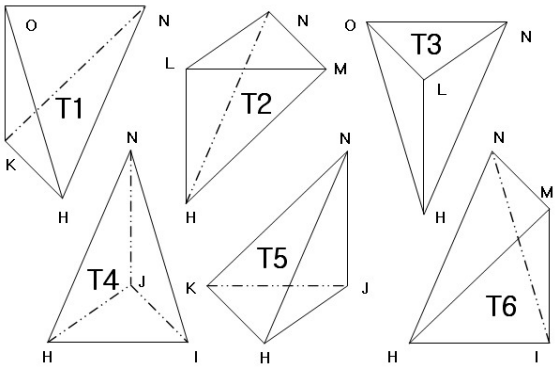


Fig. 6. Diagonal Division

TABLE I
TETRAHEDRAL INTERPOLATION

	Condition	H_0	H_1	H_2	H_3	$\boldsymbol{O}_1$	$\boldsymbol{O}_2$
T1	$g \geq b \geq r$	$\Delta - g$	$b - r$	$g - b$	r	$\boldsymbol{O}_O$	$\boldsymbol{O}_K$
T2	$b > r > g$	$\Delta - b$	$r - g$	$b - r$	g	$\boldsymbol{O}_M$	$\boldsymbol{O}_L$
T3	$b > g \geq r$	$\Delta - b$	$g - r$	$b - g$	r	$\boldsymbol{O}_O$	$\boldsymbol{O}_L$
T4	$r \geq g > b$	$\Delta - r$	$g - b$	$r - g$	b	$\boldsymbol{O}_J$	$\boldsymbol{O}_I$
T5	$g > r \geq b$	$\Delta - g$	$r - b$	$g - r$	b	$\boldsymbol{O}_J$	$\boldsymbol{O}_K$
T6	$r \geq b \geq g$	$\Delta - r$	$b - g$	$r - b$	g	$\boldsymbol{O}_M$	$\boldsymbol{O}_I$

The calculation of the resulting interpolation equation (7) involves only 4 multiplications for each R, G, B component and totally 12 multiplications whereas the cubic interpolation equation (2)-(4) use 14 multiplications for each component and totally 42 multiplications. Since multiplication is the one of complex modules in hardware implementation, (7) can greatly reduce the complexity of the high speed color transform hardware. Based on this simplicity of the tetrahedral interpolation, the hardware architecture for high speed color gamut mapping will be given in the next section.

III. HARDWARE ARCHITECTURE BASED ON TETRAHEDRAL INTERPOLATION

The proposed hardware architecture based on tetrahedral interpolation discussed in the previous section is illustrated in Fig.7. The proposed hardware architecture consists of "3-D LUT", "Weight Calculator" and "Tetrahedral Interpolation". The "3-D LUT" part generates $\boldsymbol{O}_0 \sim \boldsymbol{O}_3$ which are pre-calculated and stored in lookup tables. And the "Weight Calculator" computes the required weight values $H_0 \sim H_3$ in (7) as defined in Table I. Finally, the "Tetrahedral Interpolation" part performs the interpolation and returns the output color signal. As reported in [7][8], 9x9x9 pre-calculated mapping data sufficiently meet the requirement of accuracy with little deterioration in the image quality. In hardware implementation of this paper, the number of stored mapping data is assumed to be same as in [7]. But, it can be easily modified to adopt the other size of three-dimensional look-up table.

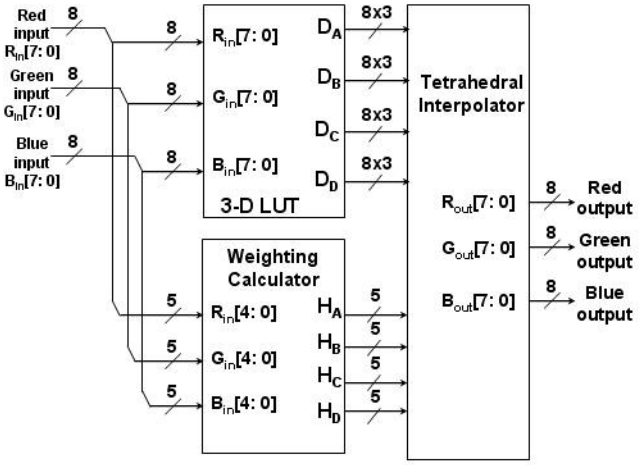


Fig. 7. Architecture of Proposed Hardware Based on Tetrahedral Interpolation

A. 3-D LUT

Since the mapping values at vertex points are used to calculate a new mapping value in every pixel clock, $\boldsymbol{O}_0 \sim \boldsymbol{O}_3$ are required simultaneously to handle the interpolation in a high speed. In this paper, the 3-D LUT part is implemented by using 4 one-dimensional look-up tables and an address decoder, as shown in Fig. 8. The 9x9x9 = 729 vertex points are separated and stored into 4 one-dimensional look-up tables (*LUT0 ~ LUT3*) in order to easily generate the mapping values in parallel. The rule for the separation is as follows.

$$N_{LUT} = (R_{in}[7:5] + G_{in}[7:5] + B_{in}[7:5]) \bmod 4 \qquad (8)$$

where N_{LUT} refers to the selected one-dimensional look-up table where the mapping point is stored and $X_{in}[7:5]$ (X=R or G or B) represent the three highest bits of X. By convention, $X_{in}[7:5] = 8$ if X=255.

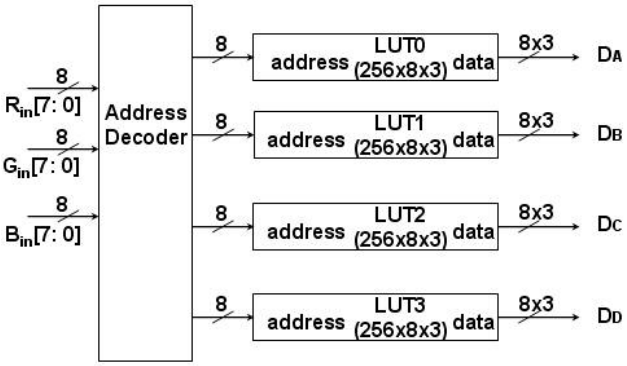


Fig. 8. Implementation of 3-D LUT part Using 4 One-dimensional Lookup Table

The lookup table separation rule (8) does not only separates 729 vertex points into *LUT0 ~ LUT3* but also uniquely defines the positions of 8 vertex points of a cube containing the input point in *LUT0 ~ LUT3*. Fig. 9. shows the configuration of *LUT0 ~ LUT3* depending on N_{LUT} . In Fig. 9., the number in the circle at the vertex denotes the look-up table which stores the mapping value at the point. For example, assume that an input is given as (R_{in}, G_{in}, B_{in}) = (67, 5, 129) where R_{in}[7:5] = 2, G_{in}[7:5] = 0, and B_{in}[7:5] = 4. Then, N_{LUT} = 2 and the 8 vertex points of a cube containing the input are stored as shown in Fig. 3(c). Note that, in each four cases in Fig. 3., two one-dimensional look-up tables store three points as *LUT0* and *LUT3* in Fig. 3(c). But, as described in Table. I., only one point from those lookup tables is recalled to handle the tetrahedral interpolation. The above case where r=3, g=5 and b=1 is corresponding to T5 in Table. I. and only $\boldsymbol{O}_J$ from *LUT0* and $\boldsymbol{O}_K$ from *LUT3* are required to the tetrahedral interpolation.

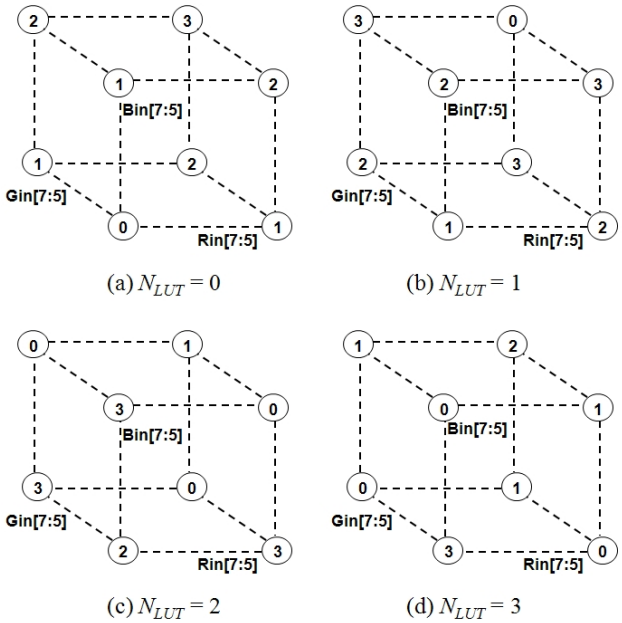


Fig. 9. Look-up table configuration on 8 vertex points depending on N_{LUT}

B. Address Decoder

The address decoder is designed to simplify the complexity of hardware implementation. For this, define the base address A_B as follows.

$$A_B = 27 \times B_{in}[7:5] + 3 \times G_{in}[7:5] + (R_{in}[7:5] >> 2) \quad (9)$$

where ($>> n$) implies n bits right shift. With the lookup table separation rule (8) and the base address (9), 729 vertex points are distributed and stored in *LUT0 ~ LUT3* at the address A_B . According to (9), four vertex points in $R_{in}[7:5]$ axis may have the same base address, but they are stored in different lookup tables. With the base address of (10), the addresses of eight vertex points of a cube which encloses an input can be simply obtained by adding offsets to A_B. The address decoding logic can be described as follows.

$$\begin{aligned}
&\text{address for vertex } H := A_B \\
&\text{address for vertex } I := A_B+\alpha \\
&\text{address for vertex } J := A_B+3+\alpha \\
&\text{address for vertex } K := A_B+3 \\
&\text{address for vertex } L := A_B+27 \\
&\text{address for vertex } M := A_B+27+\alpha \\
&\text{address for vertex } N := A_B+30+\alpha \\
&\text{address for vertex } O := A_B+30
\end{aligned} \quad (10)$$

where vertex H ~ O is defined as in Fig. 6. and α is defined as follows.

$$\alpha = \begin{cases} 1 & \text{if Rin}[7:5] = 3 \text{ or } 7 \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

TABLE II
ADDRESS DECODING LOGIC (PHASE I)

Condition	$addr_A$	$addr_B$	$addr_C$	$addr_D$
$g \geq b \geq r$	A_B	A_B+3	A_B+30	$A_B+30+\alpha$
$b > r > g$	A_B	A_B+27	$A_B+27+\alpha$	$A_B+30+\alpha$
$b > g \geq r$	A_B	A_B+27	A_B+30	$A_B+30+\alpha$
$r \geq g > b$	A_B	$A_B+\alpha$	$A_B+3+\alpha$	$A_B+30+\alpha$
$g > r \geq b$	A_B	A_B+3	$A_B+3+\alpha$	$A_B+30+\alpha$
$r \geq b \geq g$	A_B	$A_B+\alpha$	$A_B+27+\alpha$	$A_B+30+\alpha$

TABLE III
ADDRESS DECODING LOGIC (PHASE I)

N_{LUT}	*LUT0*	*LUT1*	*LUT1*	*LUT1*
0	$addr_A$	$addr_B$	$addr_C$	$addr_D$
1	$addr_D$	$addr_A$	$addr_B$	$addr_C$
2	$addr_C$	$addr_D$	$addr_A$	$addr_B$
3	$addr_B$	$addr_C$	$addr_D$	$addr_A$

The process for address decoding involves two phases. In the first phase (Table. II.), depending on the three highest bits from the input signal, A_B is calculated as in (9). And then, temporary four addresses for four one-dimensional look-up tables are assigned according to the condition on the diagonal division of 6 tetrahedral. This process discriminates three points stored in the same one-dimensional look-up table. In second phase (Table. III.), using the value of N_{LUT} as in (8), the actual addresses for *LUT0 ~ LUT3* are assigned based on the configuration of the four lookup tables as shown in Fig. 9.

C. Weight Calculator

As defined in Table I., "Weight Calculator" calculates $H_0 \sim H_3$ which are required to compute the tetrahedral interpolation in (7). In the computation of (7) H_0 should be multiplied to the stored mapping value at the vertex H ($\boldsymbol{O}_0$) and the other weights should be multiplied to the mapping values at the appropriate points. However the "3-D LUT" returns the stored mapping values in an arbitrary way. For example, the stored value in *LUT0* could be a different value from $\boldsymbol{O}_0$ depending on N_{LUT} as illustrated in Fig. 9. In order to handle this mismatch, a weight switching block changes all weight values $H_0 \sim H_3$ according to the value of N_{LUT} as shown in Fig. 10.

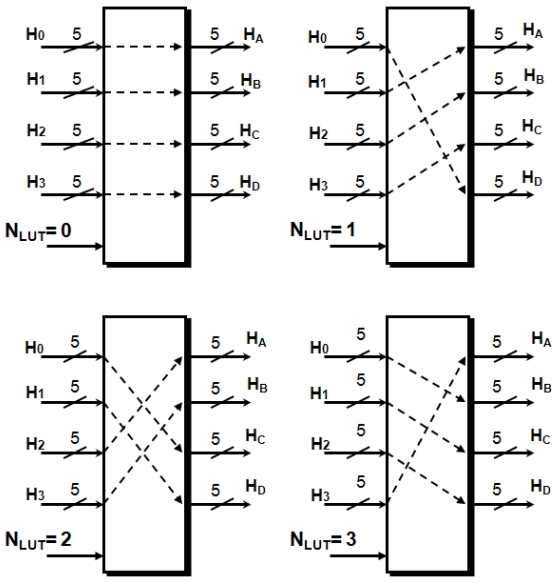


Fig. 10. Weight Switching Block

D. Tetrahedral Interpolation

Using the data from "3-D LUT" part and weights from "Weight Calculator", "Tetrahedral Interpolation" part computes the output value as follows.

$$\boldsymbol{O}_{out} = \frac{1}{\Delta}(H_A \times \boldsymbol{O}_A + H_B \times \boldsymbol{O}_B + H_C \times \boldsymbol{O}_C + H_D \times \boldsymbol{O}_D) \quad (12)$$

IV. EXPERIMENTAL RESULTS

The proposed hardware architecture is simulated in C and matlab. In the simulation, the 729 direct mapping data in [7][8] which is obtained for adjusting a sample PDP is utilized to build the lookup table. Fig. 11. illustrates the color gamut

mapping result and it shows that the proposed method has the almost same color mapping quality compared to [7].

(a) Original Image (b) Gamut Mapped Image

Fig. 11. Matlab Simulation Result

(a) Original Image (b) Gamut Mapped Image

Fig. 12. VHDL Simulation Result

The proposed hardware architecture is implemented in VHDL and the VHDL simulation shows the same results as matlab simulation. Table IV. compares the FPGA implementation of the proposed method to RRLT in [7]. As described in this table, the proposed method makes less use of hardware elements than RRLT. Especially the usage of BELs(basic elements) and multipliers is greatly reduced. This VHDL simulation shows that the proposed method can be adopted to reduce the hardware cost. Fig. 12. illustrates the gamut mapping of VHDL simulation and shows the similar result.

TABLE IV
COMPARISON OF FPGA IMPLEMENTATION

	RRLT	Proposed
Registers	200	144
Multiplexers	49	45
Adders/Subtracters	34	43
Multipliers	48	14
BELS	2678	1574
Filpflops/Latches	703	491
MULTs	42	12
MULT18X18s	42	12

V. CONCLUSION

The three dimensional interpolation is widely used for many kinds of color space transformation. With acceptable usage of memory and computing resources, this interpolation technique promises accuracy that meets most requirements of color signal transformation. And, recently, a hardware architecture for three dimensional interpolation has been successfully implemented to handle color signal enhancement for high definition digital TV devices where high speed real-time color gamut mapping is required[7].

In this paper, tetrahedral interpolation is adopted to design new hardware architecture for the high speed real-time color gamut mapping. Since the interpolation equation is simpler than the cubic interpolation technique used in [7][8], the proposed method reduces the complexity of the required hardware and implementation cost.

As in [7], the proposed method can accommodate an arbitrary color signal mapping with simple change of look-up tables. The number of intervals for approximating can be changeable and it can be selected by affordable memory size and required color mapping precision. In addition, the concept of RRDLT in [8] can be easily applied to the proposed method. The hardware cost can be further reduced by adopting the concept of RRDLT. The proposed method is expected to achieve more efficient hardware structure for high-end digital TV application.

REFERENCES

[1] Raja Bala, Ricardo deQueiroz, Reiner Eschach and Wencheng Wu, , "Gamut mapping to preserve spatial luminance variations," *Journal of Image Science and Technology,* vol. 45, no. 5, pp. 436-443, Sep/Oct. 2001.

[2] Chae-Soo Lee, Yang-Woo Park, Seok-Je Oh and Yeong-Ho Ha, "Gamut mapping algorithm using lightness mapping and multiple anchor points for linear tone and maximum chroma reproduction," *Journal of Image Science and Technology,* vol. 45, no. 3, pp. 209-223, May/Jun. 2001.

[3] Hung-Shin Chen and Hiroaki Kotera, "Three-dimensional gamut mapping method based on the concept of image dependence," *Journal of Image Science and Technology,* vol. 46, no. 1, pp. 44-52, Jan/Feb. 2002.

[4] B. Pham and G. Pringle, "Color correction for an image sequence," *IEEE Computer Graphics and Applications,* vol. 15, no. 3, pp. 38-42, August 1995.

[5] H. Haneishi, K. Miyata, H. Yaguchi and Y. Miyake, "A new method for color correction in hardcopy from CRT images," *Journal of Image Science and Technology,* vol. 37, no. 1, pp. 30-36, Jan/Feb. 1993.

[6] Byoung-Ho Kang, Jan Morovic, M. Ronnier Luo, and Maeng-Sub, "Gamut compression and extension algorithms based on observer experimental data," *ERTI Journal,* vol. 25, no. 3, pp. 156-170, 2003.

[7] Dongil Han, "Real-time color gamut mapping method for digital TV display quality enhancement," *IEEE Trans. on Consumer Electronics,* vol. 50, no. 2, pp. 691-699, 2004.
[8] Dongil Han, "A cost effective color gamut mapping architecture for digital TV color reproduction enhancement," *IEEE Trans. on Consumer Electronics,* vol. 51, no. 1, pp. 168-174, 2005.
[9] Po-Chieh Hung, "Colorimetric calibration in electronic imaging devices using a look-up-table model and interpolations", " *Journal of Electronic Imaging",* vol. 2, no. 1, pp. 53-61, 1993.
[10] James M. Kasson, Sigfredo I. Nin, Wil Plouffe and James L. Hafner, "Performing color space conversions with three dimensional linear interpolation", " *Journal of Electronic Imaging",* vol. 4, no. 3, pp. 226-250, 1995.
[11] K. Kanamori, H. Kotera, O. Yamada, H. Motomura, R. Iikawa, and T. Fumoto, "Fast color processor with programmable interpolation by small memory," *Journal of Electronic Imaging,* vol. 2, no. 3, pp. 213-224, Jul. 1993.
[12] K. Kanamori and H. Kotera, "Color correction technique for hard copies by 4-neighbors interpolation method," *J. Imaging Sci. Technol.,* vol. 36, no. 1, pp. 73-80, Jan/Feb. 1992.
[13] Heidi M. Schoolcraft, Manish S. Kulkarni and Moshe Broudo, "Streamlined Tetrahedral Interpolation," U.S. Patent 6 466 333, January 26, 1998

Hak-Sung Lee received the B.S., M.S. and Ph.D degree in electrical and electronics engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 1989, 1991 and 1996 respectively. From 1996-1998, he was a chief research engineer at Machine Intelligence Group in LG Electronics Institute of Technology, Seoul, Korea. From 1998-2000, he was a chief research engineer in LG Innotech Inc., Kyunggi-do, Korea. He is currently an associate professor in the Department of Electronics Engineering at Sejong University in Seoul, Korea. His research interests include intelligent control, learning control, display quality enhancement for digital TV and robot control. He is a member of IEEK(The Institute of Electronics Engineers of Korea) and KRS(Korea Robotics Society).

Kyung-Suk Kim received the B.S. and M.S. degree in electronics engineering from Sejong University, Seoul, Korea in 2005 and 2007. He is currently a research engineer in Grandport Co., Ltd. in Seoul, Korea. His research interests include robot control and color gamut mapping. He is a member of IEEK(The Institute of Electronics Engineers of Korea) and KRS(Korea Robotics Society).

Dongil Han received the B.S. degree in electronics and computer engineering from Korea University, Seoul in 1988 and the M.S. and Ph.D degree in electrical and electronics engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 1990 and 1995 respectively. From 1995-2003, he was a chief research engineer at digital TV R\&D Laboratories in LG Electronics Inc., Seoul, Korea. He is currently an associate professor in the Department of Computer Engineering at Sejong University in Seoul, Korea. His research interests include image processing, display quality enhancement for digital TV, system on chip, and robot vision. He is a member of IEEE, IEEK(The Institute of Electronics Engineers of Korea) and KRS(Korea Robotics Society).