논문 2004-41SP-5-24

해상도 절감 3차원 룩업 테이블을 이용한 실시간 색역폭 매핑 방법

(Real-Time Color Gamut Mapping Method Based on the Three-Dimensional Reduced Resolution Look-Up Table)

한 동 일*

(Dongil Han)

요 약

본 논문에서는 기존에 PC 모니터와 프린터 사이의 색 재현성 향상을 위해서 사용되던 색역폭 매핑 방법을 디스플레이 장치에 적용하기 위한 실시간 색역폭 매핑 방법을 제시하였다. 다양한 디스플레이 장치가 개발됨에 따라서 각 디스플레이 별로 색 재현성의 차이가 발생하고 있으며 기존의 색역폭 매핑 방법을 디지털 TV의 디스플레이 장치에 적용하기 위해서는 수 나노초 단위의 처리 속도가 필요하다. 이를 위하여 본 논문에서는 절감 3차원 룩업 테이블(three-dimensional reduced resolution look-up table)의 하드웨어 구조를 제안하고 이를 이용하여 색역폭 매핑을 실시간으로 수행하는 방법을 제공한다. 제안된 하드웨어 구조는 먼저 simulation을 통하여 타당성을 검증하였으며 이후 FPGA와 ASIC으로 구현되어 디지털 TV의 디스플레이 장치의 화질 개선에 성공적으로 적용되었다.

Abstract

A novel real-time color gamut mapping method is described. The color gamut mapping method that is used for enhancing the color reproduction quality between PC monitor and printer devices is adopted for digital TV display quality enhancement. The high definition digital TV display devices operate at the clock speed of around 70MHz ~ 150MHz and permit several nano seconds for real-time gamut mapping. Thus, the concept of three-dimensional reduced resolution look-up table is introduced for real-time processing. The required hardware can be greatly reduced by look-up table resolution adjustment. The proposed hardware architecture is successfully implemented in FPGA and ASIC and also successfully adopted in digital TV display quality enhancement purposes.

Keywords: Color gamut mapping, Image processing, Display quality enhancement

I. 서 론

최근 디지털 TV 방송이 시작됨에 따라서 고해상도의 디스플레이 장치를 장착한 디지털 TV의 보급이 확대되고 있다. 그리고 기존의 아날로그 TV의 대다수를 차지하고 있던 CRT 디스플레이 장치의 경우 대형화가 어려운 단점으로 인해서 디지털 TV 시대에서는 점차 사용 비율이 줄어들 것으로 예측되는 반면 PDP TV, LCD TV, LCD 프로젝션 TV, DLP 프로젝션 TV 등 대화면이 용이한 디스플레이 장치들의 사용이 점차 증가될 것으로 예측되고 있다. 그러나 새롭게 개발되는 디스플레이 장치들의 경우 대화면이 용이한 반면 CRT 디스플레이 장치와 비교하여 화질, 시야각, 명암비, 밝기, 수명 등 여러 가지 면에서 CRT의 성능에 못 미치고 있는 것도 사실이다. 특히 제일 중요한 화질의 경우 CRT 디스플레이 장치의 화질과 비교하면 그 수준이 떨어지고 있는 실정이며 영상 처리 기법을 이용한 화질

* 정회원, 세종대학교 컴퓨터공학과
(Department of Computer Engineering, Sejong University)

※ 본 논문은 한국과학재단 목적기초연구(R01-2003-000-10785-0)지원과 IDEC의 Tool지원으로 수행되었음.

접수일자: 2004년4월1일, 수정완료일: 2004년5월20일

개선과 관련하여 다방면의 연구 개발이 이루어지고 있다[1~4].

특히 본 논문에서는 기존에 PC용 컬러 모니터와 컬러 프린터 간의 색 재현성의 차이를 보정하는 데에 적용되는 색역폭 매핑(color gamut mapping)의 개념을 디지털 TV 분야에 적용하여 디스플레이 장치의 색 재현성을 개선하는 방법을 제안하고자 한다.

기존의 색역폭 매핑 방법[5~10]들은 대부분 모니터에서 재현되는 색을 프린터 장치에 효과적으로 표현하기 위한 색역폭 축소 기법에 대한 해결책 들을 제시하고 있다. 이러한 기법 중에서 공간 영역의 색역폭 매핑 기법[5]은 각 화소별 색역폭 매핑 방법의 단점을 극복하기 위하여 제안된 방법이다. 이 방식은 기존의 화소별 색역폭 매핑을 수행한 후에 밝기 정보와 원 영상의 밝기 정보와의 차이를 계산하고 차이 신호에 대해서 필터링을 수행하여 그 결과를 더하는 방법을 사용한다. 그리고 필터링에 의해서 색역폭의 외부로 매핑되는 부분을 내부로 매핑시키는 과정을 추가적으로 사용한다. 이 방법은 기존 화소별 매핑 방법의 여러 가지 문제점을 줄이면서 좋은 실험 결과를 제공한다.

위의 방법과 다른 접근 방법으로 다양한 앵커 점들에 대한 밝기 정보 매핑[6] 방법이 있다. 밝기 정보 매핑을 통하여 두 색역폭 간의 최대의 크로마 값을 갖는 위치에서의 밝기 값의 차이를 최소화하는 기법이다. 또 다른 접근 방법으로서 3차원의 색역폭 축소 기법[7]이 제안되었다. 원 영상과 프린터 장치간의 색역폭의 관계를 고려한 후 L*a*b* 공간에서 색역폭을 축소하는 방법을 사용하고 있으며 기존의 이차원의 색역폭 매핑 기법에 비해서 우수한 결과를 제공한다.

이 밖에 다양한 색역폭 매핑 기법이 제안되고 우수한 성능을 제공하고 있지만 대부분의 색역폭 매핑 기법이 디스플레이 장치에 실시간으로 적용되기에는 매우 큰 어려움이 존재한다. 색역폭 매핑 알고리즘이 디지털 TV의 디스플레이 장치에 적용이 되기 위해서는 약 10 나노 초 정도의 처리 속도가 필요하며 기존의 알고리즘을 직접 이용해서 구현하기는 거의 불가능하다.

따라서 본 논문에서는 해상도 절감 3차원 룩업 테이블의 개념을 이용하여 색역폭 매핑을 실시간으로 구현할 수 있는 하드웨어 구조를 제안하고 이를 통해서 기존의 다양한 색역폭 매핑 알고리즘을 적용하여 디지털 TV의 화질을 실시간으로 개선할 수 있는 색역폭 매핑 방법을 제안한다.

Ⅱ. 해상도 절감 3차원 룩업 테이블

대부분의 디스플레이 장치들은 Red, Green, Blue의 삼원색의 선형 조합으로 다양한 색을 만들어 내게 된다. 그리고 삼원색이 재현하는 색상의 정확한 표현을 위하여 ITU-R BT. 709[11]에서는 CIE 1931의 색 좌표계인 x, y 좌표계에서의 삼원색의 위치를 정의하고 있다. 그러나 디스플레이 장치의 제작에 사용되는 컬러 필터나 형광체의 종류, 영상 신호의 이득 등에 따라서 각각의 디스플레이 장치는 표준 신호와는 서로 조금씩 다른 삼원색을 제공하게 된다. 따라서 같은 컬러 영상 정보가 인가됨에도 불구하고 서로 다른 삼원색과 신호 이득 등에 따라서 디스플레이 장치 별로 서로 다른 컬러 신호를 디스플레이 하고 있다.

이러한 차이를 표현하는 방법으로 SMPTE RP177-1993[12]에서 기술한 XYZ 색 공간을 들 수 있다. XYZ 색공간은 RGB 색공간의 스펙트럼 영역에 존재하는 음수 부분을 없애기 위하여 도출한 가상적인 공간이다. 그러나 RGB 색 공간과 마찬가지로 이 좌표계에서의 거리의 차이가 인간이 느끼는 색의 차이와 일치하지 않는 단점이 존재한다. 이러한 문제를 해결하기 위하여 CIE에서 정의한 균등 색공간인 L*a*b*, L*u*v* 색공간[13~15] 이 도입되었으며 인간이 느끼는 색의 차이를 이러한 균등 색공간에서 기하학적인 거리의 차이로 계산할 수 있는 척도를 제공해 준다.

비록 L*a*b* 및 L*u*v* 색 공간이 직접 디스플레이 용도로는 사용되지 않으나 색을 표현하는 장치인 컬러 모니터, 컬러 스캐너, 컬러 프린터 등의 색의 표현 특성을 정확하게 표현하는 장점을 가지고 있다. 따라서 디스플레이 장치의 화질 개선을 위해서는 대부분의 색역폭 매핑 알고리즘들은 다음 그림 1과 같은 구조를 이용하여 색역폭 매핑을 구현하게 된다.

이러한 색역폭 매핑 방법을 실시간으로 처리하기 위해서는 전체 블록의 하드웨어화를 고려할 수 있다. 하드웨어로 구현하는 첫 번째 방법으로서 두 개의 색좌표

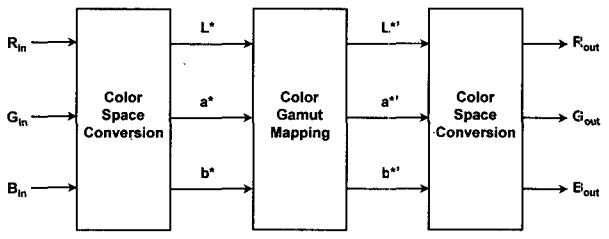

그림 1. 전형적인 색역폭 매핑 단계
Fig. 1. A typical color gamut mapping stage.

변환기와 색역폭 매핑부를 직접 하드웨어로 구현하는 방법을 생각할 수 있다. 그러나 RGB 색 공간과 L*a*b* 색 공간의 변환식의 경우 아래에 나타낸 바와 같이 매우 비선형적이기 때문에 이를 실제로 하드웨어로 구현하기는 매우 어렵다.

즉 식 (1)의 경우는 RGB 색공간을 XYZ 색공간으로 변환하는 식으로 SMPTE RP177-1993[12]에 정의되어 있으며 3x3 매트릭스 연산으로 쉽게 구현이 가능하다. 그러나 식(2) ~ 식(6)에 나타낸 바와 같이 XYZ 색공간을 L*a*b* 색공간으로 변환시키기 위해서는 매우 복잡한 비선형 연산을 하여야 하며 하드웨어로 직접 구현하는 데에는 어려움이 존재한다.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

$$L^* = 116 \ f\left(\frac{Y}{Y_n}\right) - 16 \quad (2)$$

$$a^* = 500 \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] \quad (3)$$

$$b^* = 200 \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right] \quad (4)$$

여기서, $f(q) = q^{1/3} \quad for \quad q > 0.008856$ (5)

$$f(q) = 7.787\, q + \frac{16}{116} \quad for \quad q \le 0.008856 \quad (6)$$

이고 X_n, Y_n, Z_n 값은 기준 백색의 X, Y, Z 값을 각각 나타내며 CIE 색도계에서 x = 0.3127, y = 0.3290 에서의 백색 값을 나타낸다[13].

색역폭 매핑부 또한 하드웨어로 구현하기에 매우 어려운 특성이 있다. 그림 2에 CIE L*a*b* 색공간에 표현된 ITU-R BT. 709[11] 표준 HDTV display 장치의 색역폭을 메쉬 형태로 표현하였으며 테스트 대상 PDP 장치의 색역폭을 볼륨 형태로 구분하여 나타내었다.

이 그림에서 알 수 있는 바와 같이 두 디스플레이 장치의 색역폭이 일치하지 않으며 다양한 종류의 컬러 영역에서 디스플레이 특성의 차이를 나타내고 있다. 이 예에서 나타낸 바와 같이 디스플레이 장치 간의 색재현성을 보정하기 위해서는 비선형적인 색역폭 매핑 방법을 필요로 하고 있으며 이로 인하여 하드웨어적인 구현이 매우 어렵게 된다. 또한 필요에 따라서 기존에 개발된 알고리즘의 보완 및 대체가 필요한 경우가 발생할 수 있다. 그리고 디스플레이 장치 별로 다른 형태의 색

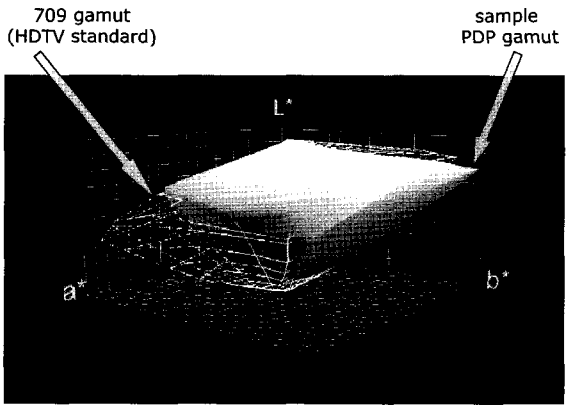

그림 2. 색역폭 차이 예
Fig. 2. A example of color gamut difference.

역폭을 가지게 되며 색역폭 매핑 또한 다른 알고리즘이 필요할 경우가 생기게 된다. 이러한 경우 색역폭 매핑 알고리즘을 바로 하드웨어로 구현할 경우 대처가 곤란한 문제가 생기게 된다. 이와 같이 색좌표 변환부나 색역폭 매핑 부를 직접 하드웨어로 구현하는 데에는 여러 가지 문제점이 존재한다.

색역폭 매핑 방법을 하드웨어로 구현하는 또 다른 방법으로서 두 개의 색좌표 변환기능과 색역폭 매핑 기능을 테이블로 만들어서 3차원 룩업 테이블로 구성하는 방법을 들 수 있다. 즉 입력되는 R_{in}, G_{in}, B_{in} 값에 대해서 두 번의 색좌표 변환과 색역폭 매핑을 통하여 생성되는 새로운 R_{out}, G_{out}, B_{out} 값을 각각 저장하여 사용함으로써 다양한 색역폭 매핑 알고리즘을 수용할 수 있는 하드웨어 구조가 얻어지게 된다. 그러나 이 경우 필요한 기억 장치의 양은 8비트의 영상을 고려하였을 때에 256x256x256x3 byte = 50,331,648 byte로서 하드웨어로 구현하기에는 거의 불가능한 용량이 필요하게 된다.

따라서 본 논문에서는 3차원 룩업 테이블의 매핑 해상도를 낮추고 나머지 부분은 보간을 통하여 계산해 냄으로써 삼차원 룩업 테이블과 거의 유사한 성능을 제공하는 해상도 절감 3차원 룩업 테이블(Three- Dimensio -nal Reduced Resolution Look-up Table: RRLT) 구조를 제안하고 이를 이용하여 필요한 하드웨어의 양을 대폭 줄이면서 실시간으로 색역폭 매핑을 구현 가능한 방법을 제안한다.

그림 3에 해상도 절감 3차원 룩업 테이블의 개념도를 나타내었다. 여기에서는 R_{in}, G_{in}, B_{in} 각각 256 계조를 갖는 색 성분의 MSB n 비트 정보를 이용하여 룩업 테이블을 구성한다. 그림 3에는 n이 3인 경우를 나타내었으며 이 경우 그림 4에 나타낸 바와 같이 해상도 절감 룩업 테이블을 이용하여 색역폭 매핑이 가능하기 위해

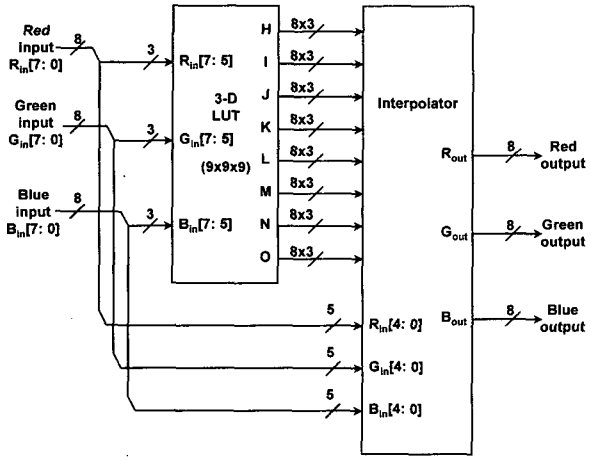

그림 3. 해상도 절감 3차원 룩업 테이블의 개념도
Fig. 3. A block diagram of RRLT when n = 3.

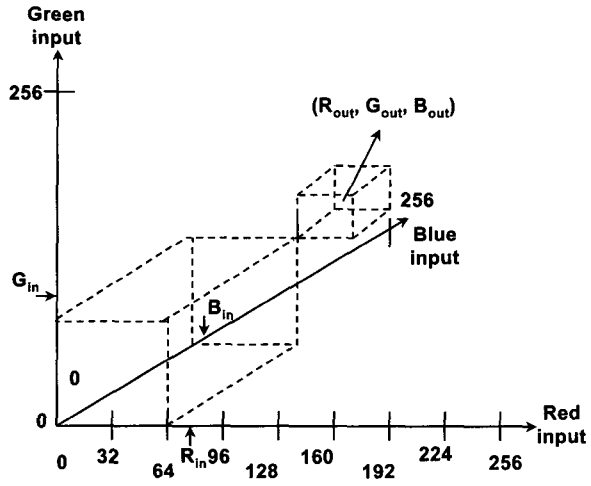

그림 4. 해상도 절감 시의 매핑 예
Fig. 4. Three dimensional mapping example.

표 1. n에 따른 RRLT의 저장 공간 비교
Table. 1. The required memory space for RRLT.

n	Required Memory
8	50.3 MBytes
7	6.44 MBytes
6	824 KBytes
5	108 KBytes
4	14.7 KBytes
3	2.19 KBytes
2	375 Bytes
1	81 Bytes

서는 R_{in}, G_{in}, B_{in} 각각 MSB 3비트 정보를 이용하여 각각의 구간을 8개의 구간으로 나눈 뒤 매핑이 이루어져야 할 화소를 포함하는 입방체에서의 색역폭 매핑 값을 제공해 줄 수 있어야 한다.

각각의 n 값에 따라서 필요로 하는 저장 공간의 크기는 크게 달라지며 표 1에 8 비트의 영상을 기준으로 한 n에 따른 해상도 절감 3-차원 룩업 테이블 크기를 나타

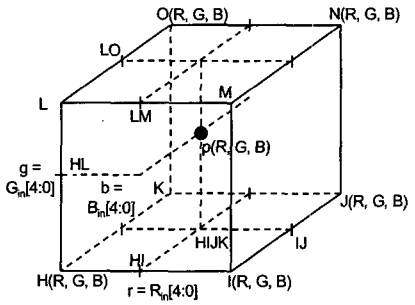

그림 5. 보간을 이용한 색역폭 매핑
Fig. 5. Three dimensional interpolati -on for color gamut mapping.

내었다.

위의 표에서 알 수 있는 바와 같이 5 이상의 n 값을 가질 경우는 필요한 저장 공간의 크기가 너무 커서 하드웨어로 구현하기에는 어려움이 존재할 수 있다. 그리고 1, 2 정도의 작은 n 값일 경우에는 필요로 하는 하드웨어의 크기는 매우 작으나 이 경우 색역폭 매핑에 필요로 하는 해상도를 얻을 수 없어서 직접 적용하기에는 어려움이 있다. 다양한 실험을 통해서 테스트를 수행한 결과 n=3인 경우 필요로 하는 해상도의 색역폭 매핑을 수행할 수 있었으며 필요한 하드웨어 양도 2 KByte 정도로 1차원 룩업 테이블과 거의 같은 정도의 하드웨어 양으로 구성할 수 있다.

n이 3인 경우 RRLT를 구성하기 위해서는 그림 4에 나타낸 바와 같이 Red, Green, Blue 신호에 대해서 각각 9개의 위치에서 새로운 Red, Green, Blue 신호에 대한 매핑 값을 저장하고 있어야 하며 총 9x9x9x3 바이트의 저장 공간으로 구성할 수 있다. 그리고 RRLT를 이용하여 최종적인 R_{out}, G_{out}, B_{out} 값을 생성하기 위해서는 그림 3에 나타낸 보간기(interpolator)를 이용하여 최종적인 변환 값을 제공하게 된다. 이때 RRLT에서는 주어진 R_{in}, G_{in}, B_{in} 신호의 MSB 3비트 값을 입력 받아서 해당 화소를 둘러싸는 8개의 좌표 점에 대해서 새로운 색역폭 매핑 값을 제공해 주어야 하며 이때 Red, Green, Blue 신호 각각에 대해서 새로운 매핑 값을 동시에 제공해 주어야 한다.

그리고 나머지 LSB 5비트 값을 이용하여 최종 색역폭 매핑 값을 보간을 통하여 구성하게 되는데 그림 5에 3차원 보간 기능을 자세히 나타내었다. 9x9x9 RRLT에서 8개의 정육면체 꼭지점에 해당하는 색역폭 매핑 값을 $R_{in}[7:5]$, $G_{in}[7:5]$, $B_{in}[7:5]$ 각각에 대해서 제공하게 되며 이 값과 R_{in}, G_{in}, B_{in} 각각의 LSB 5 비트 값을 이용하여 해당 위치에서의 색역폭 매핑 값을 보간을 이용해서 계산해 내게 된다.

그림 5에서는 처음에 H점과 I점에서의 Red, Green, Blue 룩업 테이블 값과 $R_{in}[4:0]$ 값을 이용하여 HI 점에서의 색역폭 매핑 값을 계산할 수 있으며 마찬가지 방법으로 J점과 K점에 대해서도 $R_{in}[4:0]$ 값을 이용하여 JK점에서의 매핑 값을 계산해 낼 수 있다. 그 이후 HI 점과JK 점에서의 매핑 값과 $B_{in}[4:0]$ 값을 이용하여 HIJK 점에서의 매핑 값을 계산해 낼 수 있다. 즉 임의의 점에서의 각각의 컬러 성분에 대한 매핑 함수를 $gm_{component}(.)$라고 정의하면 다음과 같은 보간 식을 이용하여 p 점에서의 Red 신호에 대한 새로운 매핑 값 R_p를 다음과 같이 계산해 낼 수 있다.

$$R_{HI} = (R_H \times (32 - r) + R_I \times r) / 32 \quad (7)$$

$$R_{KJ} = (R_K \times (32 - r) + R_J \times r) / 32 \quad (8)$$

여기서

$$r = R_{in}[4:0] \quad (9)$$

$$R_H = gm_{red}(H(R,G,B)) \quad (10)$$

$$R_I = gm_{red}(I(R,G,B)) \quad (11)$$

이다. 그리고 다음 단계에서는

$$R_{HIJK} = (R_{HI} \times (32 - b) + R_{KJ} \times b) / 32 \quad (12)$$

를 얻을 수 있으며 마지막으로

$$R_p = (R_{HIJK} \times (32 - g) + R_{LMNO} \times g) / 32 \quad (13)$$

을 얻을 수 있다. 같은 방법으로 Green, Blue 신호의 p 점에서의 보간값은 다음 식으로 구해진다.

$$G_p = (G_{HIJK} \times (32 - g) + G_{LMNO} \times g) / 32 \quad (14)$$

$$B_p = (B_{HIJK} \times (32 - g) + B_{LMNO} \times g) / 32 \quad (15)$$

이와 같은 방법으로 최종적으로 p 점에서의 새로운 매핑 값을 Red, Green, Blue 성분 각각에 대해서 계산해 낼 수 있으며 이런 작업을 매 화소마다 반복적으로 수행함으로 해서 인가되는 R_{in}, G_{in}, B_{in} 화소 데이터에 대해서 색역폭 매핑 된 화소 값 R_{out}, G_{out}, B_{out} 값을 매 화소마다 계산해 낼 수 있게 된다.

Ⅲ. 실시간 처리용 하드웨어 구현

그림 3의 3차원 룩업 테이블을 실제 하드웨어로 구현하는 것은 또 다른 문제를 야기한다. 주어진 R_{in}, G_{in}, B_{in} 입력 값에 대해서 그 점의 색역폭 매핑 결과 값을 계산하기 위해서는 동시에 8개의 좌표에서의 색역폭 매핑 값을 추출해 내어야 하는 문제가 존재한다. 즉 입력 값이 가끔씩 정육면체의 꼭지점이나 모서리, 혹은 변에 대응될 때도 있으나 대부분의 경우는 주어진 R_{in}, G_{in}, B_{in} 값에 의해서 정육면체 내부의 한 점이 선택되고 이를 보간을 이용하여 매핑 값을 계산해 내기 위해서는 한 점을 둘러싸는 8개의 꼭지점의 매핑 값을 이용하여 새로운 보간 값을 계산해 내야하며 이때 8개의 매핑 값을 동시에 사용하여 보간을 수행하여야 한다.

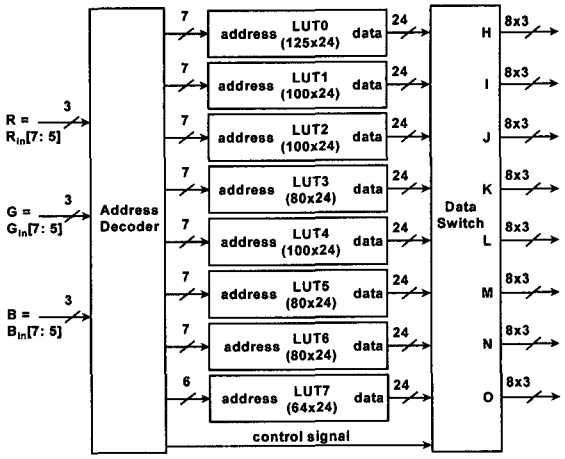

그림 6. 9x9x9 룩업 테이블의 실제 구현
Fig. 6. Real implementation of 9x9x9 look-up table.

이때 단순히 3차원 룩업 테이블을 구성 시 매 화소별로 8개의 꼭지점 좌표의 매핑 값을 동시에 제공하도록 구현하는 것은 매우 어렵다. 이러한 문제를 해결하기 위하여 보간부를 파이프라인 구조로 변경하더라도 현재 화소와 다음 화소의 매핑 값을 동시에 제공할 수 있어야 하기 때문에 같은 문제를 야기하게 된다. 이와 같은 문제를 해결하기 위해선 그림 3의 9x9x9 3차원 룩업 테이블(3-D LUT)을 실제적으로는 그림 6과 같이 8개의 1차원 룩업 테이블과 어드레스 디코더, 그리고 데이터 스위치로 구성하면 된다.

즉 9x9x9 = 729개의 좌표 점들은 각각 8개의 1차원 룩업 테이블로 나뉘어서 저장이 된다. 예를 들면 R_{in} = 0, G_{in} = 0, B_{in} = 0점에 대한 매핑 값은 LUT0의 첫 번째 어드레스에 저장이 되며 R_{in} = 32, G_{in} = 0, B_{in} = 0 점에 대한 매핑 값은 LUT1의 첫 번째 어드레스에 저장이 되며 원점을 포함하는 정육면체의 각 점이 LUT0 ~ LUT7 의 첫 번째 어드레스에 골고루 저장이 된다. 이와 같은 방법으로 전체 3차원 데이터를 8개의 일차원 룩업 테이블에 저장할 수 있으며 룩업 테이블의 선택 예를 그림 7과 그림 8에 자세히 나타내었다. 그림 7의

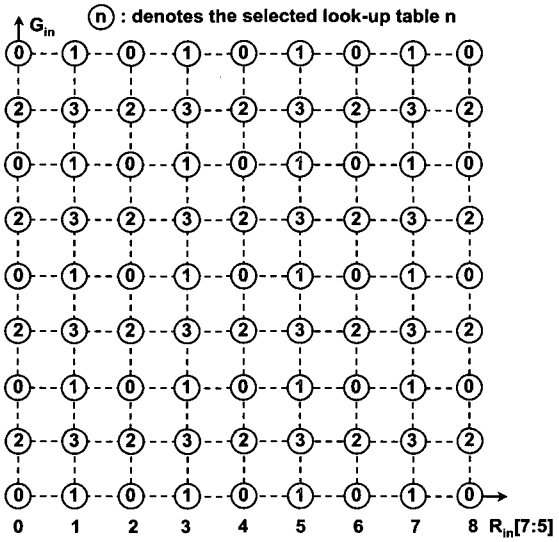

그림 7. Bin[7:5] = 0, 2, 4, 6, 8(특별히 고려)일 경우에 선택되는 1차원 룩업 테이블

Fig. 7. The selected one dimensional LUT when Bin[7:5] = 0, 2, 4, 6, and special value 8.

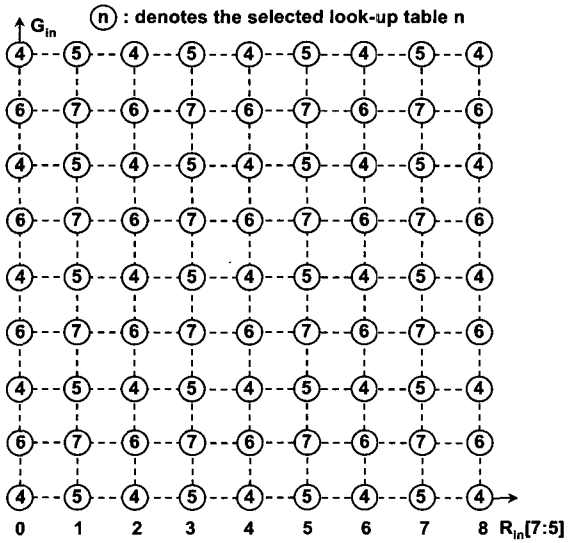

그림 8. Bin[7:5] = 1, 3, 5, 7일 경우에 선택되는 1차원 룩업 테이블

Fig. 8. The selected one dimensional LUT when Bin[7:5] = 1, 3, 5, 7.

경우는 B_{in} 값이 0, 64, 128, 192, 256(보간을 위해 특별히 고려) 인 경우에 전체 3차원 영역의 매핑 값이 저장되는 룩업 테이블의 위치를 나타내었다. 이 그림에서 알 수 있는 바와 같이 LUT0 ~ LUT3 까지의 1차원 룩업 테이블을 이용하여 특정 Bin 값에 대한 각 정육면체의 해당 좌표의 매핑 값을 저장함을 알 수 있다. 그리고 그림 8의 경우는 B_{in} 값이 32, 96, 160, 224 인 경우에 전체 3차원 영역의 매핑 값이 저장되는 룩업 테이블의 위치를 나타내었으며 LUT4 ~ LUT7이 이용됨을 알 수 있다.

그리고 그림 7과 그림 8의 형태를 교대로 사용하여 9개를 중첩함으로써 전체 입방체의 룩업 테이블 선택 영역을 구성할 수 있으며 예를 들어서 R_{in} = 10, G_{in} = 10, B_{in} = 10인 입력에 대해서는 그림 4에서 원점을 포함하는 정육면체가 선택되며 그림 7과 그림 8을 조합해서 고려하면 그림 5의 H점이 LUT0, J점이 LUT3, L점이 LUT4, N점이 LUT7에서 색역폭 매핑 데이터를 가져오게 한다. 또 다른 예로서 그림 4에서 R_{in} = 80, G_{in} = 100, B_{in} = 80인 입력에 대해서는 그림 5의 H점이 LUT2, J점이 LUT1, L점이 LUT6, N점이 LUT5에서 데이터를 가져오게 한다.

일차원 룩업 테이블을 위와 같이 구성함으로 해서 3차원 룩업 테이블을 1차원으로 최적으로 구성할 수 있으며 이 경우 각각 크기가 다른 8개의 1차원 룩업 테이블로 구성이 된다. 예를 들면 LUT0의 경우 그림 7에서 알 수 있는 바와 같이 LUT0에 저장되는 색역폭 매핑 데이터는 5x5개가 존재하며 이러한 룩업 테이블 데이터가 B_{in}[7:5] = 0, 2, 4, 6 에서 존재하고 또한 적절한 보간을 수행하기 위하여 B_{in} = 256에서의 매핑 데이터가 존재하며 총 5x5x5 = 125개의 어드레스로 이루어진다. LUT7의 경우 그림 8에서 나타낸 바와 같이 4x4개가 존재하며 이러한 룩업 테이블이 B_{in}[7:5] = 1, 3, 5, 7 에서 존재하므로 총 64개의 어드레스로 이루어짐을 알 수 있다.

이렇게 구성하였을 때 발생하는 또 다른 문제점은 각각의 일차원 룩업 테이블의 출력이 정육면체의 특정 위치 꼭지점 데이터를 출력하지 않고 임의의 꼭지점 위치의 데이터가 출력되게 된다. 즉 앞의 예에서와 같이 LUT0의 출력이 그림 5의 정육면체의 H 점에 해당할 수도 있고 L 점에 해당할 수도 있으며 입력 데이터 값에 따라서 정육면체의 모든 위치에 해당하는 값을 제공할 수 있다. 이 경우 추후 보간기의 복잡도를 증가시키게 된다. 따라서 그림 6에 나타낸 바와 같이 8개의 일차원 룩업 테이블의 출력에 데이터 스위치부를 두어서 임의의 입력 데이터에 대해서도 데이터의 위치를 적절히 변경하여 항상 같은 규칙으로 보간을 수행하게 함으로써 보간부를 최적으로 구성할 수 있다.

그리고 8개의 1차원 룩업 테이블의 선택과 동시에 입력 영상에 대해서 각각의 룩업 테이블의 어드레스가 적절히 선택되어야 한다. 또한 색역폭 매핑 데이터를 저장하는 단계와 입력 영상에서 색역폭 매핑 데이터가 실시간으로 출력되는 단계에서도 적절한 어드레스가 선

택되어야 한다. 이를 위한 8개의 룩업 테이블의 어드레스 디코딩 로직은 다음 식들로 표현되며 이 경우 색역폭 매핑 데이터를 저장하는 단계와 실시간으로 색역폭 매핑이 이루어지는 단계 모두 같은 어드레스 디코딩 로직을 사용할 수 있다.

$$LUT0\_addr=(R+1)/2+5*((G+1)/2)+25*((B+1)/2) \quad (16)$$

$$LUT1\_addr=(R\ )/2+4*((G+1)/2)+20*((B+1)/2) \quad (17)$$

$$LUT2\_addr=(R+1)/2+5*((G\ )/2)+20*((B+1)/2) \quad (18)$$

$$LUT3\_addr=(R\ )/2+4*((G\ )/2)+16*((B+1)/2) \quad (19)$$

$$LUT4\_addr=(R+1)/2+5*((G+1)/2)+25*((B\ )/2) \quad (20)$$

$$LUT5\_addr=(R\ )/2+4*((G+1)/2)+20*((B\ )/2) \quad (21)$$

$$LUT6\_addr=(R+1)/2+5*((G\ )/2)+20*((B\ )/2) \quad (22)$$

$$LUT7\_addr=(R\ )/2+4*((G\ )/2)+16*((B\ )/2) \quad (23)$$

위의 디코딩 연산은 모두 정수 연산으로 계산되며 R = R_{in}[7:5], G = G_{in}[7:5], B = B_{in}[7:5]로 나타내었다. 그리고 색역폭 매핑 데이터의 저장을 위해서는 총 729 개의 정육면체 꼭지점에서의 색역폭 매핑 데이터를 저장하여야 하며 실제로 영상으로 입력되지 않지만 보간을 위해서 R_{in} = 256, G_{in} = 256, B_{in} = 256 등의 좌표점에서의 색역폭 매핑 데이터도 같이 저장을 하여야 한다. 모든 R_{in}, G_{in}, B_{in} 값이 동시에 256일 경우는 그림 7에 나타낸 바와 같이 LUT0에 매핑 값이 저장되며 이때 상위 3bit의 MSB를 고려한 R, G, B 값은 8이 되며 이 경우 식 (16)에 대입해 보면 최종 매핑 데이터는 LUT0의 어드레스 124로 매핑 됨을 알 수 있다.

이와 같은 방법으로 최종적으로 p 점에서의 새로운 매핑 값을 Red, Green, Blue 성분 각각에 대해서 계산해 낼 수 있으며 이런 작업을 매 화소마다 반복적으로 수행함으로 해서 인가되는 R_{in}, G_{in}, B_{in} 화소 데이터에 대해서 색역폭 매핑 된 화소 값 R_{out}, G_{out}, B_{out} 값을 매 화소마다 계산해 낼 수 있게 된다.

Ⅳ. 실 험 결 과

앞에서 제안한 하드웨어 구조는 먼저 simulation을 통해서 알고리즘의 정당성을 확인하였다. 색역폭 매핑을 수행함에 있어서 총 729개 위치에서의 매핑 값을 저장하고 나머지 부분은 보간을 통해서 계산해 내게 되는데 이 연산 과정이 영상의 해상도를 저하시키지 않음을 확인할 수 있었다. 즉 보간 과정을 통해서 색역폭 매핑의 정밀도가 떨어지지만 그 결과 영상의 해상도에는 아무런 영향을 주지 않는 것을 확인할 수 있었다. 또한 제안된 하드웨어 구조는 1차로 VHDL로 구현하였고 VHDL simulation 결과 C simulation 결과와 같은 결과를 제공함을 확인하였다. 그리고 하드웨어 Emulation을 통해서 다양한 영상에 대해서 동작함을 확인하였다. 최종적으로 FPGA와 ASIC으로 구현하였으며 총 gate count는 표 2에 나타내었으며 총 10만 게이트 이내로 구성됨을 알 수 있다.

표 2. 구현 시 사용된 게이트 수
Table. 2. The used gate counts.

구분	Memory	Logic	Total
Gate Count	47,303	45,658	92,961

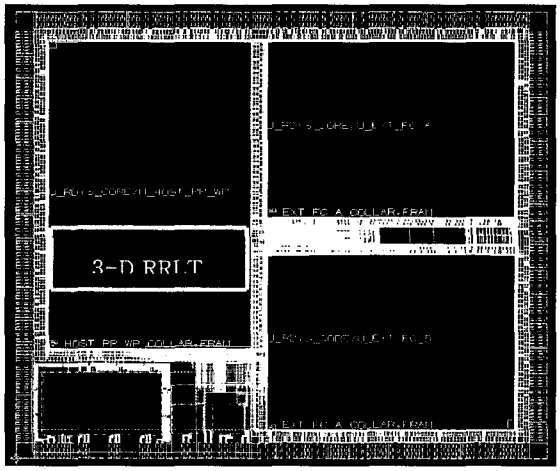

그림 9. 3-D RRLT 기능을 갖는 HDTV 영상 처리기의 레이아웃 도면
Fig. 9. The layout plot of the HDTV video processor including 3-D RRLT

동작 속도로는 HDTV 포맷인 1920x1080 순차주사 화면으로 디스플레이가 가능하도록 162HMz의 동작 속도를 가지도록 설계되었으며 0.18 mm의 공정이 사용되었다. 3-D RRLT를 포함하는 HDTV용 영상 처리 칩의 레이아웃 도면을 그림 9에 나타내었으며 3-D RRLT 부분을 별도의 표시로 나타내었다.

본 논문에서 제안한 하드웨어 구조는 색역폭 매핑 알고리즘의 종류와 무관하게 사용할 수 있으며 실제 테스트를 위해서는 참고문헌 [10]의 색역폭 확장 알고리즘을 사용하였으며 그림 2의 색역폭 차이를 줄이도록 알고리즘을 개발하고 이 계수들을 이용하여 실시간 매핑 테스트를 수행하였다. 즉 그림 2의 테스트 PDP에 대하여 색역폭 매핑 결과 화면의 한 예를 그림 10에 나타내었으며 잔디 부분의 녹색이나 하늘색이 좀 더 정확하게 재현됨을 확인할 수 있다.

(a) 원 영상 (b) 색역폭 매핑된 영상
(a) original image (b) gamut mapped image

그림 10. 실시간 색역폭 매핑 결과 예
Fig. 10. An Example of real-time color gamut mapping.

V. 결 론

본 논문에서는 기존에 모니터와 프린터 간의 색재현 문제를 개선시키는 데에 적용되던 색역폭 매핑 문제를 디스플레이 장치에서의 색 재현성을 향상시키는 데에 적용하였다. 그러나 디스플레이 장치에 적용하기 위해서는 수 나노 초 단위의 처리 속도가 필요하여 RRLT의 개념을 도입하고 이를 이용하여 실시간 색역폭 매핑이 가능한 하드웨어 구조를 개발하고 설계 및 검증을 통해서 제안된 구조의 정확성을 검증하였다.

또한 본 논문에서 제안한 하드웨어 구조는 색역폭 매핑 알고리즘의 기법의 종류에 무관하게 실시간으로 적용할 수 있는 장점이 있다. 또한 제안된 하드웨어 구조는 가능한 하드웨어 크기에 따라서 해상도를 조절할 수 있으며 하드웨어나 ASIC으로 쉽게 구현이 가능한 구조이다. 제안된 하드웨어 구조는 추후 색역폭 매핑 뿐만이 아니라 색좌표 변환, 감마 변환 등 기존의 색 처리 알고리즘을 동시에 구현이 가능하여 다양한 색 신호 처리 알고리즘에 범용으로 적용이 가능한 장점이 있다.

참 고 문 헌

[1] LeRoy DeMarsh, "Colorimetry for HDTV," IEEE Trans. on Consumer Electronics, vol. 37, no. 1, pp. 1-6, 1991.

[2] D. Han, C.-Y. Shin, S.-J. Choi, and J.-S. Park, "A Motion Adaptive 3-D De-interlacing Algori-thm based on the Brightness Profile Pattern Difference," IEEE Trans. on Consumer Electro-nics, vol. 45, no. 3, pp. 690-697, 1999.

[3] G. de Haan, J. Kettenis and B. De Loore, "IC for motion compensated 100 Hz TV with a smooth motion movie-mode," IEEE Trans. on Consumer Electronics, Vol. 42, pp.165-174, May 1996.

[4] S.-Y. Kim, D. Han, S.-J. Choi and J.-S. Park, "Image Contrast Enhancement Based on the Piecewise-Linear Approximation of CDF," IEEE Trans. on Consumer Electronics, Vol. 45, no. 3, pp.828-834, August 1999.

[5] Raja Bala, Ricardo deQueiroz, Reiner Eschach and Wencheng Wu, "Gamut Mapping to Preser-ve Spatial Luminance Variations," Journal of Image Science and Technology, Vol. 45, no. 5, pp.436-443, September/October 2001.

[6] Chae-Soo Lee, Yang-Woo Park, Seok-Je Cho 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/June 2001.

[7] Hung-Shing 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, pp44-52, January/February 2002

[8] B. Pham and G. Pringle, "Color Correction for an Image Sequence," IEEE Computer Graphics and Applications, pp.38-42, 1995.

[9] 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, 1993.

[10] Byoung-Ho Kang, Jan Morovic, M. Ronnier Luo, and Maeng-Sub Cho, "Gamut Compression and Extension Algorithms Based on Observer Experimental Data," ETRI Journal, Vol. 25, no. 3, pp.156-170, 2003.

[11] Rec. ITU-R BT 709, "Parameter Values for the HDTV Standards for Production and Internatio-nal Programme Exchange," 2000

[12] SMPTE Recommended Practice RP177-1993, "De-rivation of Basic Television Color Equations," 1993.

[13] CIE, "Uniform Color Spaces - Color Difference Equations Psychometric Color Terms," Commis-sion Internationale de L'Eclairage, Publication No. 15, Supplement No. 2, Paris, 1978.

[14] CIE, "Colorimetry," 2nd edition. CIE Publication No. 15.2. Vienna, Austria: CIE. 1986.

[15] Phil Green and Lindsay MacDonald, "Color Engineering : Achieving Device Independent Colour," Wieley SID Series in Display Techno -logy, 2002.

저 자 소 개

한 동 일(정회원)
1988년 2월 고려대학교 전자전산공학과 졸업(학사)
1990년 2월 한국과학기술원 전기 및 전자공학과 졸업(석사)
1995년 2월 한국과학기술원 전기 및 전자공학과 졸업(박사)
1995년 2월~2003년 2월 LG전자 디지털 TV 연구소 책임연구원.
2003년 3월~현재 세종대학교 컴퓨터공학과 부교수
<주관심분야: 영상 처리, 컴퓨터 비전, 디지털 TV, 시스템 온 칩>