논문 03-02-10

개선된 YUV신호를 RGB신호로 변환하는 단일칩 설계에 관한 연구

A Study on Simple chip Design that Convert Improved YUV signal to RGB signal

李 治 雨*, 朴 相 俸**, 陣 鉉 埈*, 朴 魯 京*
Lee Chi Woo*, Park Sang Bong**, Jin Hyun Jun*, Park Nho Kyung*

요 약

현행 TV 신호는 HDTV나 컴퓨터 모니터에서 사용하는 encoding 기술과는 완전히 다르다. 다시 말하면 기존의 아날로그 TV에서는 Interlace 방식을 사용하는 반면 HDTV 및 컴퓨터 모니터에서는 Interlace 방식 대신 Progressive 방식을 사용한다. 따라서 Interlace 방식을 사용하는 미디어의 신호를 Progressive 방식을 사용하는 미디어에서 사용하기 위해서는 보간 알고리즘이 필요하다. 보간 알고리즘으로는 ELA(Edge-Based Line Average) 알고리즘이 보편적으로 사용된다.

본 논문에서는 ELA 알고리즘의 수평 및 수직 윤곽선 복원 단점을 개선시킨 ADI(Adaptive De-interlacing Interpolation) 알고리즘을 사용하여 Interlace 방식의 YUV 디지털 신호를 Progressive 방식의 RGB 디지털 신호로 변환시켜 주는 주사선 변환 ASIC 칩을 설계하였다. 설계 언어로는 VHDL이 사용되었다.

Abstract

A current TV out format is quite different from that of HDTV or PC monitor in encoding techniques. In other words, a conventional analog TV uses interlaced display while HDTV or PC monitor uses Non-interlaced / Progressive-scanned display. In order to encode image signals coming from devices that takes interlaced display format for progressive scanned display, a hardware logic in which scanning and interpolation algorithms are implemented is necessary. The ELA(Edge-Based Line Average) algorithm have been widely used because it provided good characteristics.

In this study, the ADI(Adaptive De-interlacing Interpolation) algorithm using to improve the ELA algorithm which shows low quality in vertical edge detections and low efficiency of horizontal edge lines. With the De-interlacing ASIC chip that converts the interlaced Digital YUV to De-interlaced Digital RGB is designed. The VHDL is used for chip design.

Keyword : ADI algorithm, De-interlacing, YUV, RGB, VHDL

湖西大學校 情報通信工學,
(Dept. of Info.&Comm. Hoseo Univ.*)
世明大學校 情報通信工學**
(Dept. of Info.&Comm.Semyung Univ**)
接受日: 2003年 8月 16日 修正完了日: 2003年 11月 14日

I. 서 론

영상을 처리하는 방식에는 주사선 변환을 위한 보간 방식과 Color 신호처리를 용이하게 하기 위해 Color 신호를 변환하는 Color Space Converter가 있다. 현행 TV 표준 신호인 NTSC 신호에서는 Interlace 방식을 사용하고 있다. 그러나 차세대 영상 미디어인 HDTV 및 컴퓨터 모니터 등에서는 Progressive 방식을 사용하고 있다. 차세대 영상 미디어인 HDTV에서 Interlace 방식 대신에 Progressive 방식을 사용하고 있는 이유는 Interlace 방식을 사용할 때보다 Progressive 방식을 사용할 때에 화면내의 깜박거림 현상이 줄어들고, 수직해상도가 향상되며, 고스트 현상이 줄어들어 질 좋은 영상을 보여줄 수 있기 때문이다[1].

기존의 NTSC방식으로 제작된 방대한 영상 소스들을 차세대 영상 미디어인 HDTV 및 컴퓨터 모니터 등 Progressive방식을 사용하는 미디어에서 사용하기 위해서는 De-interlacing(주사선 변환) 과정이 꼭 필요하다. De-interlacing 방식에는 선형 보간 방법 및 비선형 보간 방법이 있다. 비선형 보간 방법은 선형 보간 방법에 비해 계산량이 많아 실시간 구현이 어렵고, 필드 메모리나 프레임 메모리의 사용으로 경제성이 떨어지는 이유로 본 논문에서는 선형 보간 방식 중 ELA 알고리즘의 수평 및 수직 윤곽선을 개선한 ADI 알고리즘을 이용하여 De-interlacing 방식을 적용하였다[2-4].

Color Space Converter란 색 공간 변환기로서 현행 TV 전송 방식인 NTSC 방식 및 PAL, SECOM 방식에서는 YUV Color Space를 사용한다. 그러나 컴퓨터 모니터 및 CRT에서 영상을 디스플레이 할 때에는 RGB Color Space를 사용하고 있다. 이러한 이유로 YUV Color Space를 사용하는 영상을 컴퓨터의 모니터 및 CRT에 디스플레이 하기 위해서는 Color Space Converter는 필수 불가결하다. 컴퓨터 모니터 및 CRT에서 RGB Color Space을 사용하는 이유는 RGB Color Space를 사용할 때보다 YUV Color Space를 사용할 때에 데이터 비와 저장공간이 1.5배 감소하기 때문으로 각각의 컬러에 대하여 8비트로 RGB신호가 사용되면, 각각의 RGB픽셀은 24비트로 표현할 수 있으며, 변환과 데시메이션을 한 후의 각각의 YUV 픽셀은 휘도에 대하여 8비트, 색차에 대하여 8비트로 총 16비트로 나타낸다. 대부분의 이미지 처리에서는 YUV 신호를 사용하지만 대부분의 출력장치에서는 RGB 신호를 사용하기 때문에, 영상신호변환이 필요하다[5-6].

본 논문에서는 NTSC 방식에서 사용되는 Interlace YUV신호를 입력받아 ADI 알고리즘을 적용 Progressive 영상으로 변환한 뒤 Color Space Converter를 이용 RGB 신호로 변환하여 출력함으로서 컴퓨터 모니터 및 CRT에 바로 디스플레이 할 수 있도록 하는 영상 칩을 설계하였다.

설계 및 검증은 Xilinx FPGA Express를 이용하여 구현한 후 검증을 통하여 정상 동작을 확인한 뒤 현대 0.6um CMOS 공정을 적용 Layout 설계 Tool 인 Cadence를 이용하여 Layout을 설계하였다. 설계한 칩은 IDEC의 MPW를 통하여 제작한다.

II. ADI 알고리즘 및 Color Space

2.1 ADI 알고리즘

ELA 알고리즘은 보간하고자 하는 화소의 주변 6개의 화소를 사용하여 보간하는 방식으로 수직과 대각선 각각 45。, 135。의 두 방향으로 화소의 방향성 상관관계를 계산하여 에지 방향을 검출 검출된 방향으로 두 화소값을 평균하여 보간 하는 알고리즘이다. ELA 알고리즘은 수직 및 대각선 두 방향에 대해서만 고려함으로 수평 윤곽선이 통과할 경우에는 판단기준이 없으므로 이미지에 열화가 발생하는 단점이 있다. 그림 1은 ELA 알고리즘의 3×3 윈도우를 보여주고 있다[3].

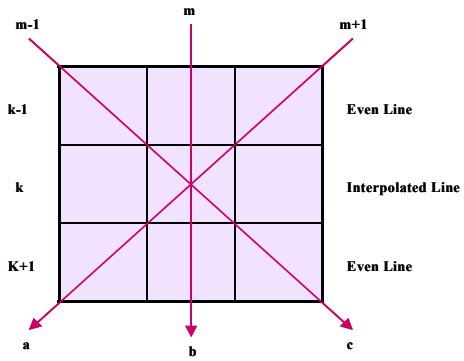

그림 1. ELA 알고리즘의 3×3 윈도우
Fig. 1. 3×3 Window for the ELA Algorithm

ADI 알고리즘은 ELA 알고리즘의 수평 및 수직 윤곽선 복원 단점을 개선시킨 알고리즘으로 다음 두 가지 측면을 고려하여 이미지를 시각적, 객관적인 면에서 개선시킨 알고리즘이다.

(1) ELA 보간 알고리즘은 화소 보간 시 윤곽선 방향을 검출하여 검출 한 방향의 두 화소를 평균하여 보간 값으로 적용하였다. 즉, 수직방향 90°와 대각선 방향 각각 45°, 135°대해서만 고려하여 보간 화소 데이터를 계산하였다. 따라서 그림 2의 (a)와 같은 형태의 윤곽선일 경우 이를 대각선 방향으로 계산을 하여 윤곽선 형태가 손상된다.

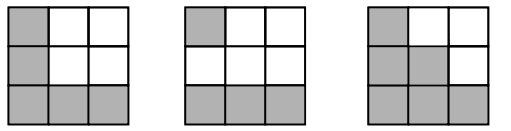

그림 2. ELA 보간 방법 사용 시 문제점
(수평 윤곽선)
Fig. 2. Problem using ELA Algorithm
(Horizontal Edge)

(2) 실제 화소 보간 값 계산 시 수평 방향으로 윤곽선이 통과하는지, 하는지는 알 수가 없다. 왜냐하면 라인이 없기 때문이다.

이러한 문제점을 보완하기 위해 ADI 알고리즘은 ELA 알고리즘의 3×3 윈도우를 윤곽선은 연속적인 특징이 있음을 착안하여 그림 3과 같이 5×3으로 확장하여 수직, 수평 윤곽선 방향을 판별하게 된다.

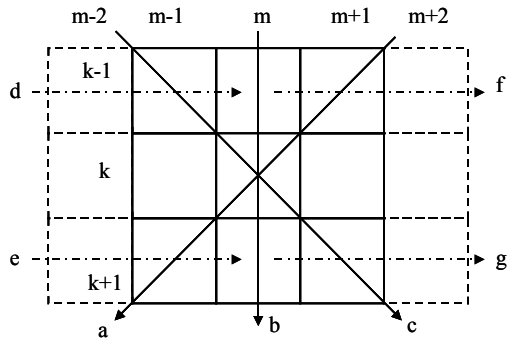

그림 3. ADI 알고리즘의5×3 윈도우
Fig. 3. A 5×3 Window of ADI Algorithm

ADI 알고리즘은 다음과 같이 두 가지 가정을 설정하였다.

(1) 5×3 윈도우에 존재하는 윤곽선 형태는 선형적이다.

(2) 계산 처리는 좌측에서 우측으로 진행한다.

이러한 가정으로부터 그림 4의 4개의 수평 방향 윤곽선 형태와 그림 5의 수평방향 윤곽선 형태를 나타낼 수 있다.

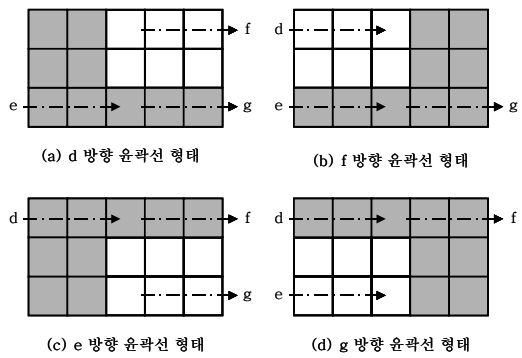

그림 4. 수평 방향 윤곽선 형태 1
Fig. 4. Case 1 : Horizontal Edge

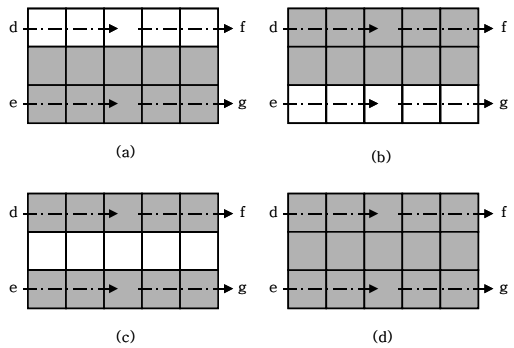

그림 5. 수평 방향 윤곽선 형태 2

Fig. 5. Case 2 : Horizontal Edge

ADI 보간 알고리즘은 기존의 ELA 형태를 가지면서 화소 보간 시 확장된 윈도우의 d, e, f, g의 화소 상관 관계를 구하여 윤곽선 형태를 파악한다. 예를 들어 e, f, g 일 경우(d 방향), d, e, g 일 경우(f 방향), d, f, g일 경우(e 방향) 그리고 e, f, g일 경우(g 방향)에 그림 4의 윤곽선 형태 중 하나로 판단하며 d, e, f, g가 모두 만족된다면 그림 5의 윤곽선 형태가 통과한다고 판단한다.

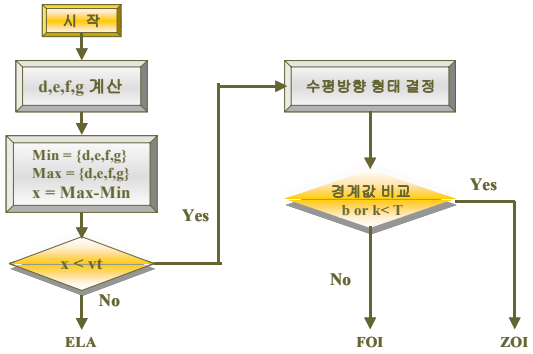

그림 6. 제안한 ADI 알고리즘 순서도

Fig. 6. Flowchart of Proposed the ADI Algorithm

그림 6과 같은 ADI 알고리즘 순서도에 따라 5×3 윈도우에서 d, e, f, g를 식(1)과 같이 계산한다.

$$d = |x(k-1, m-2) - x(k-1, m)|$$

$$e = |x(k+1, m-2) - x(k+1, m)|$$

$$f = |x(k-1, m) - x(k-1, m+2)|$$

$$g = |x(k+1, m) - x(k+1, m+2)|$$

$$Max = \{d, e, f, g\}, \quad Min = \{d, e, f, g\}$$

$$x = |Max - Min| \tag{1}$$

d, e, f, g의 4방향에 대해 상관도를 구하여 최대값과 최소값을 찾아, 그 차이값 x를 구한다. 만약 x가 주어진 vt 값보다 작다면 5×3 윈도우 안의 화소들은 거의 같은 색도의 화소값을 가지고 있다고 판단하여 수평 I 이나 수평 II방향의 윤곽선이 있다고 판단한다. 그렇지 않은 경우 즉 vt 보다 크거나 같은 경우는 기존의 ELA 방법이 사용된다. a, b, c 계산식은 식(2)와 같다.

$$a = |x(k-1, m+1) - x(k+1, m-1)|$$

$$b = |x(k-1, m) - x(k+1, m)|$$

$$c = |x(k-1, m-1) - x(k+1, m+1)| \tag{2}$$

ADI 알고리즘을 요약하면 먼저 그림 4의 수평 1 방향 (a), (b), (c), (d)형태를 파악하고 그 차 값이 주어진 vt와 비교해서 vt 보다 작으면 수평 1 방향과 수평 2 방향 윤곽선 형태로 파악한다. 반면 vt 보다 크다면 기존의 ELA가 사용되며, vt 값 보다 작은 경우 그림 5의 (a), (b), (c), (d) 형태의 윈도우를 파악하고 상하 평균값의 차로 경계값을 판별한다. 경계값 판별은 주어진 T값과 비교를 한다. 만약 T보다 작다면 ZOI 방법이 크거나 같은 경우는 FOI방법이 사용된다. 예로 그림 4의 (a)의 d 형태라면 식(3)으로 화소의 경계값(k)을 판별한다. 식(2)의 b값을 이용할 수도 있으나 식(3)을 사용 할 수도 있다. 이는 기존에 계산된 값을 이용하므로 하드웨어상의 복잡도를 줄이기 위해서이다. ADI 알고리즘은 복잡함과 평탄한 특성이 같이 있는 이미지에서는 성능이 우수하지만, 원만한 화소의 형태를 가지고 있는 이미지에서는 ADI알고리즘 부분 중 ZOI 방법이 많이 사용되어 ELA 알고리즘을 사용할 때보다 성능이 떨어지는 특성을 가지고 있다.

$$h = \{x(k-1, m) + x(k-1, m)\}/2$$

$$i = \{x(k+1, m) + x(k+1, m+2)\}/2$$

$$k = |h - i|, \quad k < T \text{ 또는 } b < T$$

(3)

가. ZOI 알고리즘

ZOI 알고리즘은 보간 방법 중에서 가장 간단한 형태의 알고리즘으로서 말 그대로 같은 화소값을 중복해서 사용하므로서 보간된 영상을 얻는다. ZOI 방법의 이용은 하드웨어 복잡도가 낮으며 메모리 사용이 적어 비용절감에 좋은 반면 이미지의 질적인 면에서 본다면 상당히 떨어지는 단점이 있다. 즉, 똑같은 화소값을 중복으로 사용하여 원 영상에서는 화소 하나였던 부분이 똑같은 화소값을 갖는 블록을 구성해서 특히 고주파 성분인 영상의 윤곽선 등에서 심각한 계단 현상을 낳게 된다. ZOI 보간 방법은 PSNR도 좋지 않고, 에지 보존 특성이 나쁘기 때문에 큰 영상에서 시각적으로 상당히 좋지 못하지만 수직 방향 윤곽선 특성만은 다른 선형필터 보다 좋은 특성을 가지고 있다[3-4].

나. FOI 알고리즘

FOI 방법은 인접한 라인의 상하 화소값을 평균하여 계산하며, 이 방법은 윤곽선의 경계값의 차가 클 경우 윤곽선의 열화가 심하게 되어 전체적으로 이미지를 흐릿하게 만드는 단점이 있다. 또한 하드웨어 측면에선 두 개의 라인 메모리가 필요하여 ZOI 함수의 사용보다 비용면에서 비싸다[3-4].

2.2 Color Space 개요

칼라 스페이스는 수학적으로 표현할 수 있는 색깔의 집합체를 말한다. 기본적인 칼라 모델에는 RGB, YIQ, YUV 또는 YCbCr 그리고 CMYK가 있다. 공통적인 칼라 스페이스는 카메라나 스캐너에서 공급되어지는 RGB를 사용한다.

가. RGB Color Space

RGB Color Space는 컴퓨터 그래픽과 이미지를 통하여 널리 사용되어져 왔다. 적색, 녹색, 청색은 데카르트 좌표 시스템에서 세 개의 영역으로 표현되는 3가지의 기본 색깔이다. 각각의 기본 요소와 같은 수를 가지고 있는 입방체의 대각선 표시된 것으로 다양한 그레이 레벨을 표현할 수 있다. RGB Color Space는 CRT에서 원하는 색깔을 만들기 위해서 적색, 녹색, 청색의 인광물질을 사용하기 때문에 그래픽 프레임 버퍼에서 가장 많이 선택하여 사용한다. 그렇기 때문에, 간단한 구조의 프레임 버퍼와 시스템 설계를 위하여 RGB 칼라 스페이스를 선택한다. 하지만 RGB는 실제로 보이는 이미지와 관계가 있을 때 매우 효과적이지 못하다

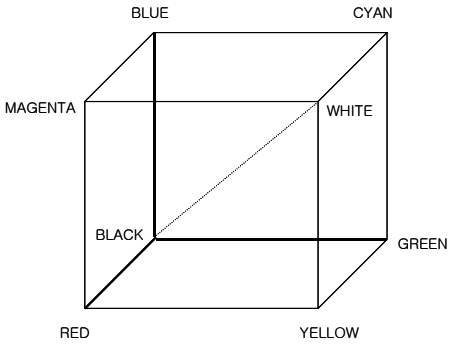

그림 7. RGB 칼라 입방체
Fig. 7. RGB Color Cube

.

세 가지의 RGB 요소는 RGB 칼라 입방체 안의 임의의 색깔을 만들 때 같은 대역폭이 필요하다. 이러한 결과는 그림 7과 같이 각각의 RGB 요소들을 위한 같은 픽셀 깊이와 같은 화면 해상도의 프레임 버퍼가 필요하다. 그러므로 RGB 칼라 스페이스에서 이미지를 처리하는 것은 효과적인 방법이 아니다[5-6].

나. YUV Color Space

YUV Color Space는 PAL, NTSC와 SECAM 신호에서 사용되어진 기본 칼라 스페이스이다. 흑백 시스템은 단지 휘도신호(Y)를 사용한다. 색차신호(U,V)는 보통의 흑백 그림을 나타낼 때 흑백 수신기와 같은 방법으로 칼라 신호를 추가하여 사용한다. 칼라 수신기는 칼라 그림을 나타내기 위하여 칼라 신호를 디코딩하여 사용한다. 감마 보정된 RGB 와 YUV 사이의 변환식은 식(4)와 같다.

$$R = Y + 1.371\ V$$

$$G = Y - 0.698\ U - 0.336\ V$$

$$B = Y + 1.732\ U$$

(4)

다. YCbCr Color Space

YCbCr 칼라 스페이스는 세계적으로 디지털 비디오 신호 표준을 개발하는 과정에서 ITU-R BT.601 권고안의 부분으로 발전되어졌다. Y는 16에서 235의 범위를 가지고 있으며, CbCr은 16에서 240의 범위를 가지며, 128은 0과 같다. YCbCr은 몇 개의 샘플링 형식(4 : 4 : 4, 4 : 2 : 2, 4 : 2 : 0)을 가지고 있다. RGB에서 YCbCr의 변환식은 식(5)와 같다[5-6].

$$R = Y + 1.371\ (\ V - 128\ )$$

$$G = Y - 0.698(\ V - 128\ ) - 0.336(\ U - 12\ )$$

$$B = Y + 1.732\ (\ U - 128\ )$$

$$Y = 0.299\ R + 0.587\ G + 0.113\ B$$

$$V = -0.172R - 0.339G + 0.511B + 128$$

$$U = 0.511R - 0.428 - 0.08\ B + 128$$

(5)

2.3 Color Space Converter

Color는 R, G, B의 삼원색을 여러 비율로 섞음으로서 표현할 수 있다. 컬러 TV 카메라는 렌즈를 통하여 촬영한 풍경을 프리즘을 통하여 삼원색으로 분해하고, 이것을 R, G, B 각각에 해당하는 세 개의 촬상소자로 받아들인 후 각 촬상소자에 대해 주사하면 R, G, B 세종류의 TV 신호가 얻어진다. 이 신호를 컴포넌트 신호라고 부른다. 그러나 이와 같은 방식은 하나의 컬러영상에 대해서 세 개의 전송로가 필요하고 전파를 효율적으로 이용하는 측면에서는 비경제적이다. 따라서 현재의 TV 방송에서는 R, G, B신호를 휘도신호(Y)와 색차신호(UV)로 변환하고 두 신호를 적당히 다중화하여 한 개의 신호로 만든 후 전송한다. 하지만 대부분의 출력장치에서는 Y, U, V 신호가 아닌 R, G, B 신호가 사용되고 있어 Y, U, V 신호를 출력하기 위해서는 색을 변환하는 Color Space Converter가 필요하다.

본 논문에서는 YUV 신호를 입력받아 RGB 신호로 변환하여 출력하는 Color Space Converter를 설계하였으며 YUV 신호를 RGB 신호로 변환하는 변환식은 식(6)과 같다[6].

$$R = Y + 1.371\ V = Y + Vr$$

$$G = Y - 0.698V - 0.336U = Y - Vg - Ug$$

$$B = Y + 1.732\ U = Y + Ub$$

(6)

III. 하드웨어 설계

3.1 VHDL을 이용한 하드웨어 설계

설계한 전체 블록은 그림 8과 같이 복합영상신호인 YUV 신호를 입력받아 ADI 알고리즘을 이용하여 De-Interlacing을 한 후에 보간된 영상을 Color Space Converter에서 입력받아 최종 출력 신호인 디지털 RGB 신호를 출력하는 회로이다.

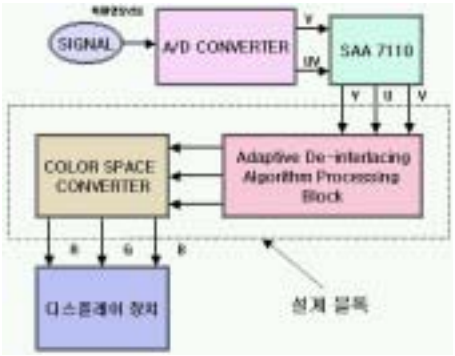

그림 8. 설계한 회로의 전체 블록도
Fig. 8. Block Diagram of Designed Total Circuit

가. ADI 알고리즘 블록 설계

그림 9는 ADI 알고리즘에 대한 블록도이다. 그림 9에서 보인 것과 같이 ADI 알고리즘은 기존의 공간 선형 필터를 최대한 이용하는 방법을 사용하고 있다.

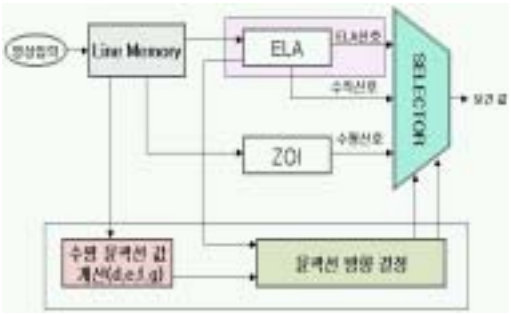

그림 9. ADI 알고리즘 블록도
Fig. 9. Block Diagram of ADI Algorithm

기존의 ELA 블록과 비교해서 ZOI 부분과 윤곽선 방향 결정 부분, 수평 윤곽선 값 계산 블록이 증가되었지만, ZOI 화소 값은 기존의 ELA 부분에서 계산 값을 사용함으로 별도의 화소 값 계산이 필요하지 않다. 즉, 윤곽선 방향 블록만 기존의 ELA에 비해 증가되었다. ADI 알고리즘 블록의 구성을 크게 분류하면 ELA 알고리즘 블록과 ZOI 알고리즘 블록, 윤곽선 방향 결정 블록으로 크게 나눌 수 있다. ELA 블록에는 가/감산기와 절대값 계산기, MUX, 그리고 비교기 등으로 구성되어 있으며, 윤곽선 결정 블록은 감산기, 최대/최소 값 비교기 등으로 구성되어 있다.

나. Color Space Converter 블록 설계

Color Space Converter 블록은 크게 전처리 블록, 행렬 연산 블록, 반올림 연산 블록으로 나눌 수 있다. 그림 10은 YUV신호를 RGB 신호로 변환하는 블록도를 나타낸 것이다.

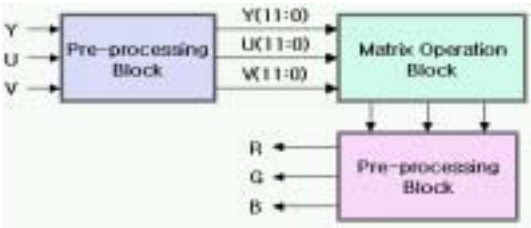

그림 10. YUV -> RGB 변환 블록도
Fig. 10. Block Diagram of YUV to RGB Conversion

전처리 블록에서는 행렬 계산에서의 소수점을 고려하여 8비트의 입력을 12비트로 만들어주는 블록이며, 행렬연산 블록에서는 식(6)을 연산하는 블록이다. 마지막으로 반올림 연산블록에서는 RGB 출력신호를 고려하여 12비트의 행렬연산블록 출력을 8비트로 반올림하는 블록이다.

a. 전처리 블록

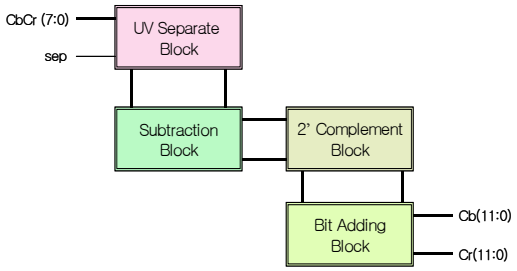

그림 11. 전처리 블록도
Fig. 11. Block Diagram of Pre-processing

전처리 블록은 그림 11과 같이 입력 신호인 Y, U, V 신호에 대하여 분리를 한다. Philips사의 7110과 같은 One Chip Fronted에서 Y신호와 UV신호가 출력된다. UV신호를 분리하여 행렬에 계산되어지는 계수가 소수점 3자리까지 연산을 하기 위하여 12비트의 입력을 가지므로, Y, U, V 신호를 12비트로 만들어 준다. 12비트로 만들어진 U, V 신호에 YUV를 RGB로 만들기 위한 행렬식을 위해 128을 감산하여 최종 UV의 신호를 출력한다.

b. 행렬연산 블록

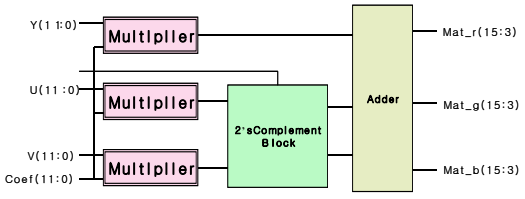

그림 12. 행렬 연산 블록도
Fig. 12. Block Diagram of Matrix Operation

YUV 형식에서 RGB 형식으로의 변환에는 덧셈기 및 곱셈기가 각각 4개씩이 필요하다. 그러나 곱셈기의 회로 양을 고려할 때 곱셈기의 수를 줄일 필요가 있다. 본 논문에서는 그림 12와 같이 곱셈기의 수를 2개로 줄여 회로를 완성하였다. 식(6)에서 V에 곱해지는 계수는 1.371과 -0.698이므로 1.371V를 Vr(R을 구하는데 사용하는 V신호), -0698V를 Vg(G를 구하는데 사용하는 V신호)라 명하고, 마찬가지로 -0.366U는 Ug(G를 구하기 위한 U신호), 1.732U는 Ub(B를 구하기 위한 U신호)라 하면, 1.731, -0.698, -0.336, 1.732는 각각 Vr, Vg, Ug, Ub를 구하기 위한 계수가 된다.

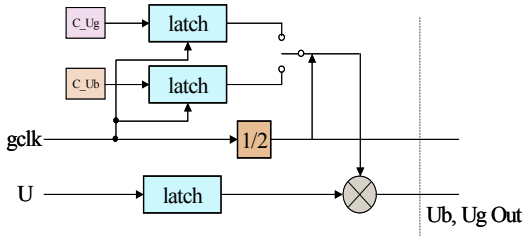

그림 13. 클럭 분주회로를 이용한 곱셈기 2의 예
Fig 13. Example of multiplier 2 that use Clock division

곱셈기 1에서는 V를 구하기 위한 신호인 Vr, Vg에 대한 곱셈 값을 곱셈기 2에서는 U를 구하기 위한 신호인 Ug, Ub에 대한 곱셈 값을 연산하며, 분주 된 클록을 이용하여 구현하였다. 그림 13에서는 곱셈기 2에 대한 예를 블록도를 이용하여 보여주고 있다.

행렬 연산 블록에서는 곱셈기의 수를 줄이기 위하여 U신호와 V신호에 대한 곱셈기를 설계한 후 클럭을 분주하여 계수를 곱하는 방식으로 설계하였다.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.371 \\ 1 & -0.336 & -0.698 \\ 1 & 1.372 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U - 128 \\ V - 128 \end{bmatrix} \quad (7)$$

행렬 연산의 경우 YUV 신호를 RGB로 변환하기 위한 기본적인 행렬식은 식(7)과 같다.

행렬 연산에서 곱셈기의 수를 줄이기 위하여 Y신호에 대한 경우는 같은 계수를 곱셈하기 때문에, 따로 분리하여 연산을 수행하도록 설계하였다. 계수 12Bit와 Y, U, V 신호 12Bit를 곱하면 24Bit가 필요하지만 연산된 결과에 영향을 주지 않는 범위에 소수점 3자리를 고려하여 결과는 13Bit가 된다. 소수점 3자리까지 연산하며, 음수의 표현은 2의 보수를 이용하여 표현하였다.

c. 반올림 연산 블록

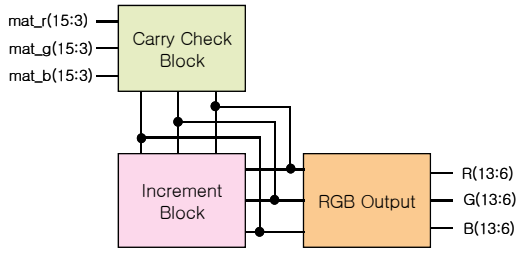

그림 14. 반올림 연산 블록도
Fig. 14. Block Diagram of Round-off

행렬 연산을 수행한 결과는 RGB의 형식으로는 적합하지 않다. 일반적인 RGB는 정수부분만을 포함하고 있다. 원하는 결과 값을 얻기 위하여 그림 14와 같이 행렬 연산을 통하여 얻어진 결과에 반올림을 하여 출력을 할 수 있도록 설계하였다. 행렬식을 통하여 얻은 결과는 13비트에서 소수점 부분을 반올림 하여 최종 출력은 8비트만이 나온다.

3.2 시뮬레이션 결과

가. 컴퓨터 시뮬레이션 결과

시뮬레이션은 컴퓨터 모의실험으로 기존의 선형 보간 알고리즘인 ELA 알고리즘과 본 논문에서 구현한 ADI 알고리즘을 비교하였다. 비교방법으로는 객관적인 판단 기준인 PSNR을 선택하였다. PSNR의 계산식은 식(8), 식(9)로 계산되었다.

$$PSNR = 10\log_{10}\frac{(255)^2}{MSE}\ [dB] \quad (8)$$

$$MSE = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}|I(i,j) - I'(i,j)|^2 \quad (9)$$

여기서 MSE는 Mean Square Error를 나타내며 I(i, j)는 원래의 화소값을, I'(i, j)는 처리된 화소값을 나타낸다.

실험에는 Lena, Couple, Bridge, Peppers 등의 표준 정지 이미지를 가지고 실험을 하였다. 표 1은 ADI 알고리즘은 vt = 25, T = 5일 경우의 결과이다.

표 1. PSNR에 대한 이미지 비교
Table 1. The Comparison of Image by PSNR

	Lena	Bridge	Couple	Peppers
ELA	35.9789	26.6718	30.7932	34.0391
ADI	36.4889	26.8694	31.0343	33.8041

그림 15는 준 동화인 세일즈맨의 30frame에 대한 ELA 알고리즘과 ADI 알고리즘의 PSNR을 비교한 것으로 약 0.4dB 정도가 개선되었다.

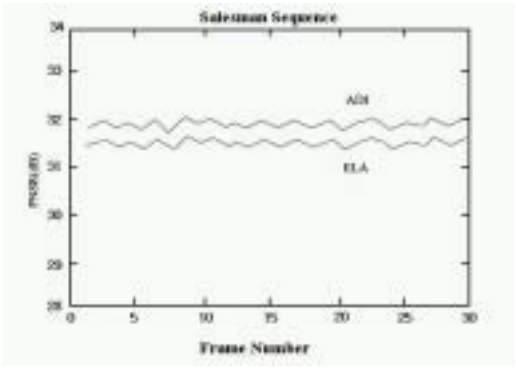

그림 15. 세일즈맨 30frame의 PSNR 비교
Fig. 15. The comparison of Salesman 30frame by PSNR

표 2는 Color Space Converter에 대한 컴퓨터 시뮬레이션에 의해 계산된 값과 회로의 시뮬레이션 결과 값을 나타낸 것이다. 시뮬레이션 결과 거의 오차가 없이 Color Space Converter에서 신호 변환이 이루어짐을 확인할 수 있었다.

표 2. YUV에서 RGB의 변환 결과 비교
Table 2. Compare with Result of YUV to RGB Conversion

	Computer Simulation			Hardware Design		
	R	G	B	R	G	B
White	235	235	235	235	235	235
Black	16	16	16	16	16	16
Red	236	12	15	236	12	15
Green	15	243	15	15	242	15
Blue	16	12	237	16	12	237
Yellow	234	234	14	234	238	14
Cyan	16	239	236	16	239	236
Magenta	235	7	235	235	7	235

나. FPGA 시뮬레이션 결과

시뮬레이션 결과 ADI 알고리즘이 기존의 ELA 알고리즘 보다 약 0.5dB 정도 PSNR이 개선되었으며, 적응 주사선 보간 알고리즘은 이미지가 전체적으로 복잡함과 평탄한 특성이 같이 있는 이미지에서 성능이 좋게 나타났다.

그림 16은 설계한 전체 회로의 시뮬레이션 결과 값을 보여 주고 있으며 입력된 복합영상신호를 De-interlacing 처리 후 RGB 신호로 변환한 뒤의 출력값을 나타내고 있다.

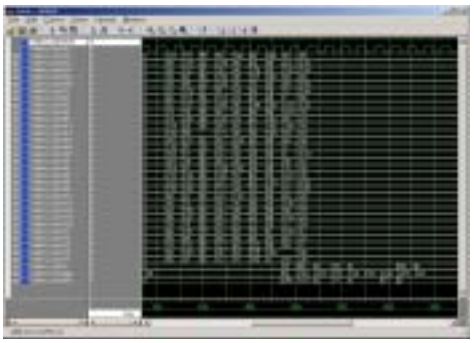

그림 16. 전체 회로의 시뮬레이션 결과
Fig. 16. Simulation Result of Adaptive De-Interlacing Block

다. FPGA Floor Planning 결과

그림 17은 VHDL을 이용하여 설계한 Color Space Converter의 Floor Planning을 나타내고, 그림 18은 전체 회로를 VHDL로 설계한 후 FPGA 타겟 보드에

구현한 그림이다.

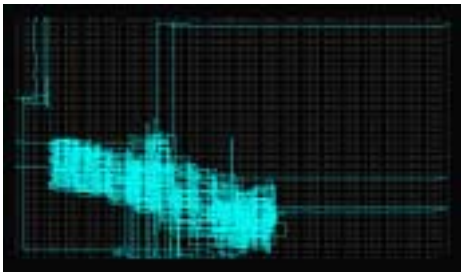

그림 17. Color Space Converter의
Floor Planning
Fig. 17. Floor Planning of Color Space
Converter

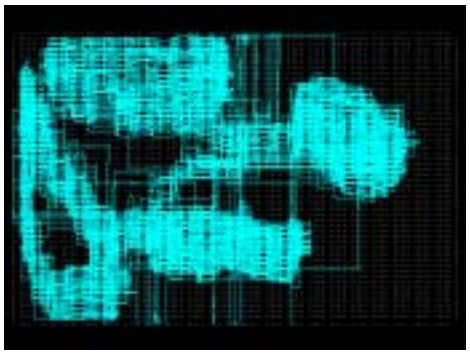

그림 18. 설계한 전체 회로의
Floor Planning
Fig. 18. Floor Planning of Designed Total Circuit

Ⅳ. Layout 설계

설계한 칩은 크게 ADI 블록과 Color Space Converter 블록으로 구성되어 있으며, 각 블록들은 기본 게이트를 설계한 후 Cadence의 instance 기능을 이용하여 레이아웃을 설계하였다.

4.1 ADI 블록 Layout

가. 가/감산기의 Layout

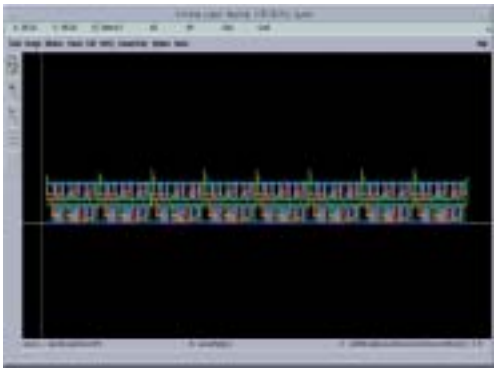

그림 19. 8Bit 가산기의 Layout
Fig. 19. Layout of 8Bit Adder

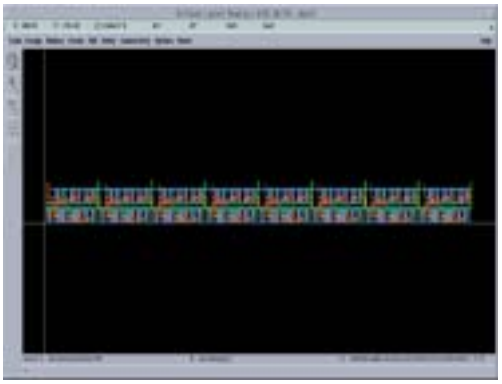

그림 20. 8bit 감산기의 Layout
Fig. 20. Layout of 8Bit subtracter

그림 19 및 그림 20은 설계한 가/감산 블록의 Layout을 보여주고 있다.

가산기 및 감산기의 Layout은 1bit 가산기 및 감산기를 설계한 후 병렬로 8개를 나열하여 Ripple Carry 방식으로 설계하였으며 입력되는 각 휘도 신호들을 불러들여 각각의 기본적인 연산을 수행하도록 하였다.

나. 절대값 계산기 Layout

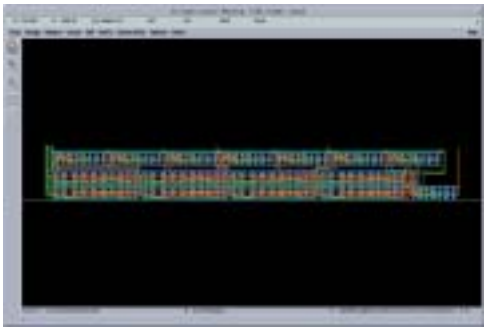

그림 21. 절대값 계산기의 Layout
Fig. 21. Layout of Absolute value computer

그림 21은 절대값 계산 블록의 Layout을 보여 주고 있다. 절대값 계산 블록은 8bit의 신호를 처리하기 위하여 D Flip-Flop 과 2 to 1 MUX 각각 8개를 병렬로 연결하여 구현하였으며, 가/감산기의 타이밍을 맞추기 위하여 클럭으로 제어하였다.

다. 정렬제어 블록의 Layout

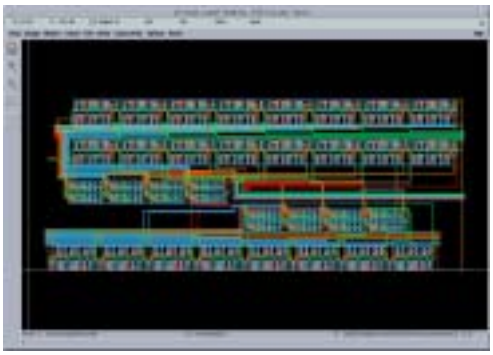

그림 22. 정렬제어 블록의 Layout
Fig. 22. Layout of Arrangement Control Block

그림 22은 정렬제어 블록의 Layout을 보여 주고 있다. 정렬제어 블록은 d, e, f, g 값을 입력으로 받아 최소값 및 최대값을 차례로 정렬하는 블록으로 상하 평균값의 차로 경계값을 판별하기 위한 전처리 블록이다.

라. ADI 블록의 전체 Layout

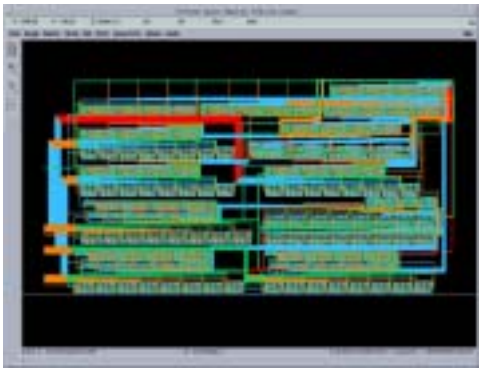

그림 23. ADI 알고리즘의 Layout
Fig. 23. Layout of ADI Algorithm

그림 23은 설계한 ADI 알고리즘의 Layout을 나타낸다. ELA 알고리즘 블록에서 미리 계산을 해놓은 각 보간 방법의 값들을 윤곽선 방향 결정 블록에서 판단되어진 신호에 따라 후단에 놓여진 선택기에서 알맞은 보간 값을 출력하게 된다.

4.2 Color Space Converter의 Layout

가. 전처리 블록의 Layout

그림 24는 전처리 블록의 Layout을 나타낸다. 전처리 블록은 음수 및 실수의 연산을 위해 Sign Bit에 1Bit 지수 Bit에는 3Bit를 첨부하는 블록으로 8Bit의 입력을 받아서 12Bit를 출력하는 블록이다.

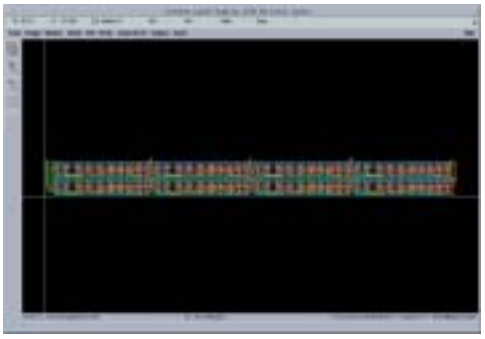

그림 24. 전처리 블록의 Layout
Fig. 24. Layout of Pre-processing Block

나. 행렬 연산 블록의 Layout

그림 25. 행렬 연산블록의 Layout
Fig. 25. Layout of Matrix Operation Block

그림 25는 행렬 연산 블록의 Layout 이다. 행렬 연산 블록의 출력은 연산된 결과에 영향을 주지 않는 범위에 소수점 3자리를 고려하여 결과는 13Bit가 되며 음수의 표현은 2의 보수를 이용하여 표현하였다.

다. Color Space Converter의 전체 Layout

그림 26은 Color Space Converter의 전체 Layout을 보여주고 있다.

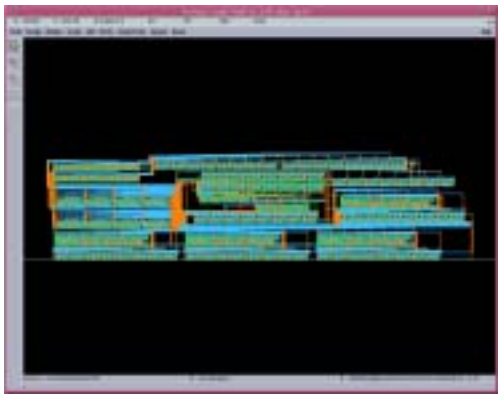

그림 26. Color Space Converter의 전체 Layout
Fig. 26. Layout of Designed Color Space Converter Block

라. 설계한 회로의 전체 Layout

ADI 알고리즘 블록 및 Color Space Converter 블록을 연결한 최종 Layout을 그림 27에서 보여주고 있다. ADI 알고리즘 블록에서 보간된 후의 YUV신호를 Color Space Converter에서 받아 들여 RGB 신호로 변환하여 출력한다.

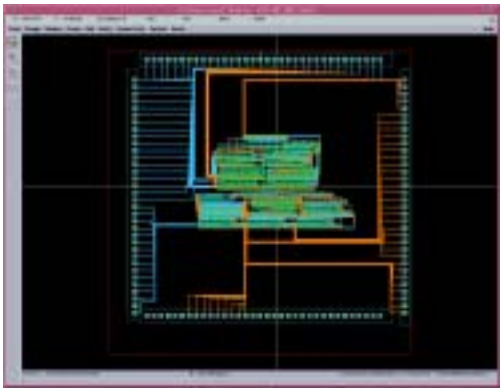

그림 27. 설계한 전체 블록의 Layout
Fig. 27 Layout of Designed Total Block

그림 28은 전체 Layout의 Design Rule Check 후의 출력화면을 Cadence Tool의 주 화면인 CDS 창에서 보여주고 있는 것으로 전체 레이아웃이 Design Rule에 맞게 설계되었음을 확인할 수 있다.

그림 28. CDS 창의 DRC report 출력
Fig. 28 Display of DRC report

스케메틱과 Layout을 비교하여 검증하는 LVS 검증에서도 merged와 rewired 없이 netlist가 match됨을 확인할 수 있었으며, 설계된 Layout의 총 게이트 수는 7,860개가 사용되었다.

Ⅴ. 결론

본 논문에서는 선형 보간 방식 중 ELA 알고리즘의 수평 및 수직 윤곽선을 개선한 ADI 알고리즘을 이용하여 De-interlacing 한 후의 YUV신호를 Color Space Converter에서 RGB 신호로 변환함으로서 Interlace YUV신호를 입력받아 바로 컴퓨터 모니터 및 CRT에 직접 디스플레이 할 수 있도록 하는 칩을 설계하였다.

설계 및 검증은 하드웨어 설계 언어인 VHDL을 이용하여 설계 XILINX FPGA Express를 통해 구현하고 검증한 뒤 현대 0.6um CMOS 공정을 사용하여 Layout을 설계하였다. 설계한 Layout은 IDEC의 MPW를 통하여 제작하게 된다. 설계한 Layout의 총 게이트 수는 7860개이며 FPGA 시뮬레이션 결과와 Layout의 시뮬레이션 결과를 비교한 결과 입력과 출력시간이 각각 49ms, 43ms로서 Layout의 시뮬레이션 결과가 7ms정도 빠름을 알 수 있었다. ADI 알고리즘 블록은 기존의 ELA 블록에 비해 그 구성이 30% 증가하였으며, 제작된 칩은 PC관련 멀티미디어 부분이나 영상 디스플레이 관련 분야에 응용 될 수 있을 것이다.

참 고 문 헌

[1] Chung J. Kuo, Ching Liao, and Ching C. Lin, "Adaptive Interpolation Technique for Scanning Rate Conversion", IEEE Trans. Circuits and Systems. Video Technology, vol. 6, no 3, pp. 317- 321, June. 1996.

[2] R. Simonetti, S. Carrato, G. Ramponi, A. Polo Filisan, "Deinterlacing of HDTV Images for Multimedia Applications", International Workshop on HDTV '92 Proceedings, Vol. 2, pp. 95-108, NOV, 18-20, 1992.

[3] Myeong-Hwan Lee, Jeong-Hoon Kim, Jeong-Sang Lee, Kyeong-Keol Ryu, and Dong-Il Song, "A New Algorithm For Interlaced To Progressive Scan Conversion Based On Directional Correlations And Its IC Design", IEEE Trans. Consumer Electronics, Vol. 2, No. 2, MAY 1994.

[4] Yeong-Taeg Kim, "Deinterlacing Algorithm Based on Sparse Wide Vector Correlations", SPIE Vol. 2727.

[5] Benjamin Gordon, "A Low-Power Multiplierless YUV to RGB Converter Based on Human Vision Perception," 1994, IEEE.

[6] Keith Jack, "Video Demystified", Harris, pp.15-34, pp.116-127, 2001

저 자 소 개

李 治 雨(學生會員)

2000. 2 : 호서대학교 정보통신공학과 졸업.
2002. 2 : 호서대학교 정보통신공학과 공학 석사.
2002.3 - 현재 : 넥스지 텔레콤 soc 연구소 연구원
<관심 분야> 신호처리, ASIC 설계

朴 相 俸(正會員)
1985. 2 : 광운대학교 전자재료공학과 졸업.
1987. 2 : 고려대학교 전자공학과 공학석사.
1992. 2 : 고려대학교 전자공학과 공학박사.
1994. 3 - 1999. 2 : 삼성반도체 ASIC 설계 팀. 2000. 3 ~ 현재 (주)옴니미디어 ASIC Team 2000. 7 ~ 현재 : @Lab Digital Design Team. 1999. 3 ~ 현재 : 세명대학교 정보통신학과 조교수. <관심 분야> Digital TV, Embedded Memory Test, Serial ATA 등

陣 鉉 埈 (正會員)
1984. 2 : 고려대학교 전자공학과 졸업
1986. 2 : 고려대학교 전자공학과 공학석사
1986. 3 - 1991. 2 : 삼성전자 시스템개발실
1998. 2 : 미국 리하이 대학교 전산학 박사
1998년 - 현재 : 호서대학교 전기정보통신공학부 조교수

<관심분야> 시스템프로그램, 멀티미디어 정보처리

朴 魯 京 (正會員)

1984. 2 : 고려대학교 전자공학과 졸업
1986. 2 : 고려대학교 전자공학과 공학석사
1990. 2 : 고려대학교 전자공학과 공학박사
1984. 3 - 1987. 2 삼성반도체 통신(주) 반도체 연구소 연구원
1999. 3 - 2000. 2 미국 오레곤 주립 대학교 ECE 연구교수
1997. 3 - 1999. 2 호서대학교 공업기술연구소장
2000. 12 - 현재 IDEC 호서대 WG 책임교수
1988. 4 -현재 : 호서대학교 정보통신공학전공 정교수
<관심 분야> 회로 및 시스템 설계(DAB, HDTV, Chip Design, Telematics 등