# 플라즈마 디스플레이 패널의 어두운 영역에서의 계조 재현을 위한 실시간 영상처리기

論 文
51C-1-7

# Real time Image Processor for Reproduction of Gray Levels in Dark Areas on Plasma Display Panel (PDP)

李 昌 勳* · 朴 勝 虎** · 姜 晋 球*** · 金 椿 宇§
(Chang-Hoon Lee · Seung-Ho Park · Jin-Gu Kang · Choon-Woo Kim)

**Abstract** - Plasma Display Panel (PDP) is required to be both the determination of white point of each gray level and the inverse gamma correction since non-balanced RGB cell and linear property of PDP, respectively. However, these two methods cause degradation of grey level representation and undesirable false contour in the dark areas on PDP. In this paper, we implemented real time image processor of the proposed error diffusion algorithm and unsharp masking operation to protect the blurring image caused by the error diffusion. Experimental results showed drastic improvements of gray level representation and reduction of undesirable false contour.

**Key Words** : PDP, white point, inverse gamma correction, false contour, error diffusion, unsharp masking

## 1. 서 론

오늘날 영상 디스플레이 기기의 발달은 기존의 CRT 의 크고, 무거운 무게와 대비되는 얇고, 가벼우며, 대형화가 가능한 LCD(Liquid Crystal Display)와 PDP(Plasma Display Panel)로 변모해가고 있다. 그러나, LCD의 주된 단점은 최대 40인치인 화면 크기의 한계와 시야 각의 협소함이다. 반면에 PDP는 장점인 거대화면 크기(현재 63인치)의 용의, 총천연색 구현, 높은 휘도와 대비 특성, 넓은 시야각과 빠른 전환율 (rapid refresh rates)로 인하여 차세대 영상 디스플레이 시장을 빠르게 잠식하고 있다[1]. 따라서, 이와 같은 PDP의 급부상에 맞는 PDP의 화질 개선은 여러 각도에서 시도되어져 왔다. RGB의 화소의 배열(the pixel arrangement)방법[2]와 1개의 frame을 여러 영역으로 나누어 발광을 수행하는 여러 가지 subfield 방법[3]등이 그러한 것들이다. 따라서, 본 논문에서는 이와 같은 PDP의 화질 개선에 대한 주제에 발맞춰 어두운 영역에서의 계조 표현 성능 향상을 위한 방법 및 그의 구현 방법을 제안한다.

일반적으로 CRT에서는 사람의 시각특성에 맞도록 비 선형적인 휘도 특성을 갖는다[4]. 하지만, PDP에서의 휘도는 선형에 가깝다. 따라서, CRT의 휘도 특성과 같게 만들기 위해서는 디지털 입력의 수정(modification)이 필요하다[5]. 이와 같은 수정과정을 역 감마 보정이라 한다. 역 감마 보정시 PDP의 어두운 영역에서는 표현 가능한 계조수가 크게 감소하여 사람의 시각에 거슬리는 의사윤곽(false contour)이 나타나게 된다. 또한, PDP의 백색점 결정시 blue의 발광휘도 특성이 좋지 않기 때문에 상대적으로 특성이 좋은 red와 green의 사용범위를 줄이게 되고 이에 따라 표현 계조수의 감소가 발생한다.

이와 같은 PDP의 역 감마 보정과 백색점의 비대칭으로 인한 어두운 영역에서의 의사 윤곽을 감소시키기 위해 제안하는 방법에서는 먼저 각 채널 별로 목표로 하는 이상적인 역 감마 보정을 수행한다. 그리고 소수 값으로 된 역 감마 보정 결과를 정수 부분과 소수 부분으로 분리한 후, 소수 부분을 반올림 과정을 통해 0 또는 1의 정수 값으로 변환한다. 이때 반올림 과정에서 발생한 오차는 오차확산 방법을 적용하여 보상하였다. 그리하여, 일정영역에 대해 표시될 계조값이 평균적으로 이상적인 역 감마 보정값과 일치하게 하였다. 또한, 제안된 알고리즘은 실제 PDP에서의 실시간 적용을 위해 VHDL를 이용한 FPGA 디바이스 인 FLEX10K200EGC599-2 가 탑재된 보드를 직접 제작하여, 알고리즘에 대한 실험 및 검증을 하였다.

더불어 제안된 오차확산 방법은 저역 통과 필터(low pass filter)의 역할을 하기 때문에, 모서리(edge) 부분이 흐릿하게(blurring) 보이는 현상이 나타난다. 따라서, 영상에 대한 향상(enhancement)과정 역시 필요하게 되어, 본 논문에서는 이러한 영상 향상 과정 중에서 단계 조절이 가능한 unsharp masking[6] 연산을 사용하고, 이를 역시 하드웨어로 구현하였다.

2.1 에서는 제안한 오차확산 방법 설명하였으며, 2.2 에서는 제안한 알고리즘에 대한 하드웨어 상의 구현 방법이 자세히 설명되었다. 또한, 2.3에서는 소프트웨어 상의 알고리즘 검증과 하드웨어 상의 시뮬레이션 결과를 나타내었고, 실제 PDP에 표시된 영상 결과를 촬영하였다. 그리고, 마지막으로 논문의 결론을 맺고자 한다.

* 李 昌 勳 : 仁荷大學 情報通信大學院 碩士課程
** 朴 勝 虎 : 仁荷大學 電氣工學科 博士課程
*** 姜 晋 球 : 仁荷大學 電子·電氣工學科 助敎授 工博
§ 金 椿 宇 : 仁荷大學 情報通信工學科 副敎授 工博
接受日字 : 2001年 10月 16日
最終完了 : 2001年 11月 13日

## 2. 본 론

### 2.1. 제안한 오차확산 방법

#### 2.1.1. 역 감마 보정(Inverse Gamma Correction)

제안한 오차확산 방법을 적용하기에 앞서 가장 먼저 하여야 할 부분은, PDP에서 red 와 green 에 비해 blue의 발광 특성이 좋지 못하기 때문에, 먼저 백색점을 결정해야한다. 백색점이 결정된 후 역 감마 보정을 통하여 CRT의 휘도 특성과 거의 같게 만들어 주어야 한다. 그림1은 CRT와 PDP의 일반적인 휘도 특성을 나타낸 것으로, CRT의 휘도 특성은 인간의 시각 특성에 맞게 디지털 입력에 대해 비 선형적인 휘도 특성을 나타낸다. 그러나, PDP의 휘도 특성은 입력 값에 대해 선형적인 휘도 특성을 나타낸다.

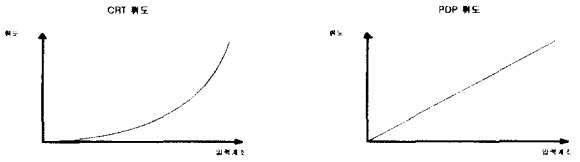


**그림 1** CRT 와 PDP의 특성휘도

**Fig. 1** The characteristic brightness of CRT and PDP

또한, PDP에서는 blue의 발광 휘도 특성이 좋지 않기 때문에 백색점을 위해 blue 계조는 256 계조를 모두 사용하고, red 와 green 계조는 blue의 그것보다는 작게 white point를 결정한다. 그림2는 일반적인 PDP의 불균형한 RGB의 발광 휘도 와 휘도의 선형성을 보정한 것이다.

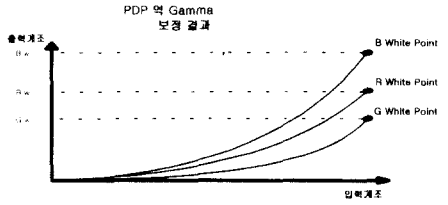


**그림 2** PDP 역 감마 보정결과

**Fig. 2** The reseult of inverse gamma correction on PDP

역 Gamma 보정을 위해, 각 채널에 대해 입력의 계조는 아래의 수식(1)에 따라 변환된다.

$$C_{gamma} = C_W \times (\frac{C}{255})^{\gamma} \qquad (1)$$

여기서 $C$ 는 입력 계조를 나타내고, $\gamma$ 는 PDP 특성에 맞는 감마값을 나타내며, $C_W$ 는 백색점에서의 계조값을 나타내고, $C_{gamma}$는 소수가 포함된 목표로 하는 이상적인 역 감마 보정값이 된다. 그러나, PDP는 정수값만을 표시 할 수 있으며, 이 값들은 나중에 가장 가까운 정수 값으로 바꾸어 주어야 한다.

#### 2.1.2. 제안한 오차확산 방법의 적용

표1 에서는 역 감마 보정의 한 예로써 B 채널부분을 보인 것이다. 첫 번째 열은 입력 계조를 나타내고, 두 번째 열인 Ideal Output은 실제로 수식(1)을 적용하여 결정된 값이며, 세 번째 열인 Actual Output은 Ideal Output에서 나온 값을 가장 가까운 정수 값으로 바꾸어 준 것이다. 그러나, 보는 바와 같이 0에서 9까지의 입력 계조에 대해서 Actual Output 모두 0 (zero)로 나타나고 있다. 이러한 의미는 0에서 9까지의 입력 계조를 구분할 수 없다는 것이 된다. 따라서, 이를 보상할 연산작용이 없다면, 어두운 영역에서 의사 윤곽이 나타난다.

**표 1** 역 감마 보정의 한 예($\gamma$=1.8 , BW = 255)

**Table 1** The example of inverse gamma correction

| Input | Ideal Output | Actual Output |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0.01188 | 0 |
| 2 | 0.04136 | 0 |
| 3 | 0.08582 | 0 |
| ... | ... | ... |
| 10 | 0.74949 | 1 |
| ... | ... | ... |
| 254 | 253.20282 | 253 |
| 255 | 255 | 255 |

어두운 영역에서 의사 윤곽을 감소시키기 위해서, 본 논문에서는 간략화된 오차확산 방법을 제안한다. 이 오차확산 방법으로 Actual Output 의 local mean 과 Ideal Output 사이의 오차를 현저히 줄일 수 있다. 덧붙여서, 사람의 시각특성은 일반적으로 local mean 값에 의해 영상을 인지하기 때문에, 실제 각각의 화소값 보다는 그 화소들에 대한 영역의 local mean 값이 Ideal Output 과 같다면, 사람의 눈으로는 거의 같게 보이게 된다.

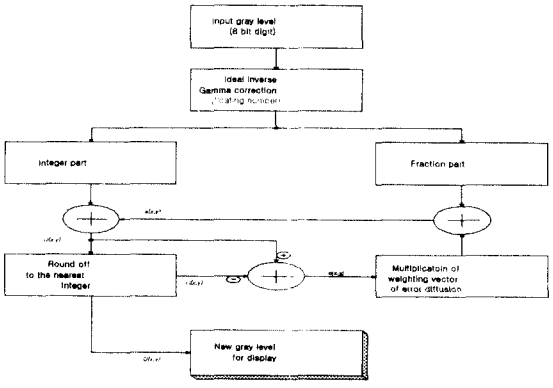


**그림 3** 제안한 오차확산 방법 흐름도

**Fig. 3** Flow chart of proposed error diffusion

그림3은 하나의 칼라 채널에 대한 오차확산 방법이 간략화 되어있고, 이 흐름도에서 Input Gray Level은 수식(1)에 의해 Ideal Inverse Gamma Correction의 계산을 수행하여 floating number로 바뀌게 된다. 그 후 정수부와 소수부의 두 부분으로 나누어진다. 정수부는 e(x,y) 와 더해진 후 가장 가까운 정수 값으로 출력된다. 또한 오차확산의 오차값은 반올림하기 전과 후의 값들의 뺄셈을 통해서 $a(x,y)$ 으로 출력되어, weighting factor 가 곱해진 후 각각의 4개의 위치로 전파되어간다. 따라서, 한 화소에서 오차를 받는다고, 역으로 생각하면 e(x,y)는 4개의 오차값과 그 위치의 소수부의 합이 된다.

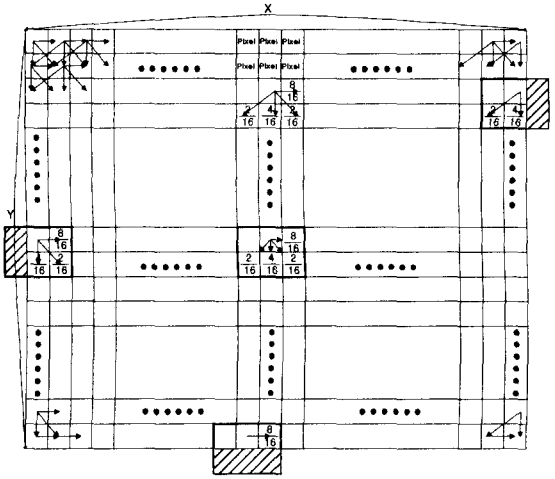


**그림 4** 1 frame 오차확산 연산 흐름도
**Fig. 4** Proposed error diffusion operation on 1 frame

그림4는 오차확산에 대한 전파를 나타낸 것으로, 바로 옆 화소, 다음 line의 아래 3개의 화소로 전파되어진다. 또한, 그림4의 frame의 4개의 측면을 보면, frame의 맨 왼쪽측면은 3방향으로 전파되고, 맨 오른쪽 측면은 2방향, 그리고, 가장 아래쪽은 1방향인 바로 옆 화소로만 전파된다. 이와 같은 과정은 수식(2)로 표현된다.

$$
\begin{aligned}
U(x,y) &= Inverse\_gamma\_LUT[input(x,y)] + e(x,y) \\
o(x,y) &= Round\_off[U(x,y)] \\
a(x,y) &= [U(x,y) - o(x,y)] \\
e(x+1,y) &= a(x,y)\times 8/16 \\
e(x-1,y+1) &= a(x,y)\times 2/16 \\
e(x,y+1) &= a(x,y)\times 4/16 \\
e(x+1,y+1) &= a(x,y)\times 2/16
\end{aligned}
\qquad (2)
$$

식(2)에서 표현된 $(x,y)$ 는 PDP의 1개의 frame을 2차원이라 가정하여 $x$ 축과 $y$축으로 나누어 각각의 화소의 위치를 표현하도록 한 것이다. 또한, $e(x+1,y)$ 는 바로 옆 화소로 전파하여, 그 다음에 오는 화소의 소수부 값과 더해지고, $e(x-1,y+1)$, $e(x,y+1)$, 와 $e(x+1,y+1)$는 다음 line에 error buffer 에 저장되어, 그 다음 line 의 그 위치에 오는 소수부 값과 더해지게 되는 것이다. 또한, 각 오차값에 대한 weighting value 인, 8/16, 2/16, 4/16, 2/16 는 Floyd and Steinberg 오차확산 알고리즘[7]을 수정한 것으로써, 하드웨어의 구현 시 곱셈을 없애고, shifter만으로 계수의 곱을 구현하기 위함이다. 따라서, 곱셈의 없애고, 단순한 shifter로 사용하여, 기존의 error diffusion 의 계산 량을 현저히 줄였다. 이것에 대한 실험 데이터는 2.3.1 절에 나타나 있다.

### 2.1.3. Edge enhancement 알고리즘 설명

계조 표현을 위해 제안한 오차확산 방법을 적용하면 영상이 전체적으로 blurring 되는 현상이 발생한다. 이를 보상하고, 영상의 화질을 개선하기 위해 경계선 부분을 강조하는 과정인 영상 향상과정을 수행하여야 한다. 영상 향상방법으로는 다양한 방법이 있으나, 본 논문에서는 향상정도를 조절할 수 있는 unsharp masking 방법을 이용하였다. Unsharp masking 방법은 처리하고자 하는 영상에서 낮은 주파수 성분을 뺀 영상을 적절하게 가중함으로써 sharpening 하는 방법이다. 이러한 과정을 수식으로 표현하면 아래 식(3)과 같게된다.

$$
\begin{aligned}
R_{out} &= s\times[R_{in} - k\times avg_{neighbor}(R)] \\
G_{out} &= s\times[G_{in} - k\times avg_{neighbor}(G)] \\
B_{out} &= s\times[B_{in} - k\times avg_{neighbor}(B)]
\end{aligned}
\qquad (3)
$$

여기서 $R_{in}$, $G_{in}$, $B_{in}$ 는 입력 계조값를 나타내고, s는 brightness scale factor 값을 나타내며, k는 향상의 정도를 결정하는 인자로써, $1-\frac{1}{s}$ 와 같다. $R_{out}$, $G_{out}$, $B_{out}$들은 출력 계조값을 나타낸다. 위 식에서 주위값의 평균값을 나타내는 $avg_{neighbor}()$는 영상의 저주파 성분을 의미한다. 따라서, 수식(3)에서 처리하고자 하는 영상에서 저주파 성분의 가중된 값을 빼면 고주파 성분이 강조되어 남아있게 된다. 이 값을 s 값에 의해 적절히 가중하면 sharpening된 결과 영상을 얻게 된다. 식에서 영상의 화질은 s 값과 $avg_{neighbor}()$의 계산시 사용하는 영역의 크기에 의해 영향을 받는다. s 값이 클수록 sharpening 정도는 증가하고, 작을수록 감소한다. 그리고, $avg_{neighbor}()$에서 고려한 영역 은 그림5와 같이 3×3 정도가 적당하였다.

| | | |
|---|---|---|
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |

**그림 5** $avg_{neighbor}()$ 계산시 고려한 영역
**Fig. 5** The masking area of $avg_{neighbor}()$

### 2.2. 실시간 Image processor 하드웨어 설계

그림6은 제안한 영상 처리기의 하드웨어 구현 시 전체 블럭도를 보여주고 있다. 간단하게 전체 블록의 신호들을 설명하면, 첫 번째로 경계선 부분의 흐릿하게 보이는 부분을 제거하기 위한 unsharp masking부분에서 R, G, B 각각 8bit 의 데이터와 한 line을 위한 sync 신호 horizontal enable(Henable)과, 한 frame을 위한 vertical enable (Venable)과, global clock(clk)이 있어서, clk의 한 주기에 1개의 data 씩 unsharp masking 블록으로 입력된다. Enhance되어진 output 데이터는 어두운 영역에서 의사 윤곽을 감소시키기 위한 오차확산 블록을 거쳐, 최종적으로 PDP에서 표시하게 된다. Unsharp masking블록이 전체 블록에서 먼저와야 하는 이유는 의사 윤곽을 제거한 후 sharpening을 하면, 역 감마 보정의 값들이 unsharp masking 연산으로 다시 바뀌어, 의사윤곽이 다시 발생하기 때문이다.

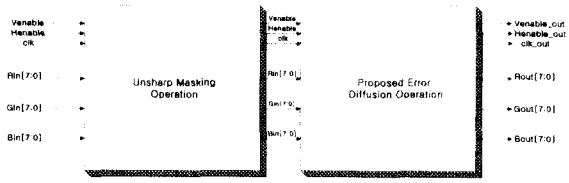


**그림 6** 제안한 실시간 영상처리기의 전체 블럭

**Fig. 6** The total block of the proposed real time image processor

#### 2.2.1. Unsharp Masking Module 설계

수식(3)은 하드웨어 설계에 맞게 수식(4)와 같이 바뀌게 된다. 변수들은 수식(3)에서 사용한 것 과 같게 된다.

$$\begin{aligned} R_{out} &= s \times R_{in} - (s-1) \times avg_{neighbor}(R) \\ G_{out} &= s \times G_{in} - (s-1) \times avg_{neighbor}(G) \\ B_{out} &= s \times B_{in} - (s-1) \times avg_{neighbor}(B) \end{aligned} \quad (4)$$

수식(4)는 k 라는 다른 변수로 치환하는 것 보다, s 라는 하나의 변수로 통일할 수 있는 이유 뿐 아니라, s 값 자체가 sharpening 이 되기 위해서는 1보다 큰 floating값 이여야 하므로, 나눗셈을 한번은 줄일 수 있다는 하드웨어 구현상의 장점이 있다. 그림7에 나타난 것은 unsharp masking 연산을 수행하기 위해, 한 화소씩 오른쪽으로 이동하면서 3×3 masking 연산을 수행하여 수식(4)의 출력을 구하고, 한 line의 masking 연산이 끝난 후 에는, 다음 line에서 역시 한 화소씩 오른쪽으로 이동하면서 3×3 operation을 수행하여 수식(4)의 출력을 얻어내는 것을 도식화 한 것이다. 그러나, 모든 line 에 대한 unsharp masking이 되기 위해서는 각 frame 에 대한 zero padding(빗 금친 부분)이 필수 적이다. 이러한 zero padding 이 없다면, masking operation 의 특성상 display 상의 가장 위쪽과 아래쪽 line 및, 양 측면에 대한 데이터를 잃어버리게 된다. 따라서, 그림7에 나타냈듯이 가장 첫 번째 line 에 대한 operation의 첫 번째 masking은 그림과 같이 4개의 데이터를 포함하며, 나머지는 빗 금친 zero로 묶여지게고, 그 다음부터는 6개의 data를 hold 하고, 위쪽 3개의 data 는 빗 금친 zero로 놓여지게 된다. 마지막 line의 operation 역시 첫 번째 line과 마찬가지로 이루어진다. 첫 line 과 마지막 line에 제외한 중간 부분의 양측면 masking operation 역시, 양 측면의 3개의 데이터씩은 zero로 padding 되고, 실제 데이터 6개를 가지고, 평균값을 낸다. Zero padding operation 은 2개의 line buffer 에서 2 line 의 데이터를 hold한 후, 그 다음부터 mean block으로 데이터를 넘기며, 양 측면의 데이터는 Henable 신호를 1개 shift 시킨 후, 원 Henable신호와 OR 연산을 하면 zero padding을 한 것처럼 Henable 신호가 각각 양 측면에 clk의 1주기의 만큼이 길어져서, 메모리 소비 없이 쉽게 zero padding 구현 할 수 있다. 이러한 unsharp masking 연산의 블록도는 그림8에 잘 나타나 있다.

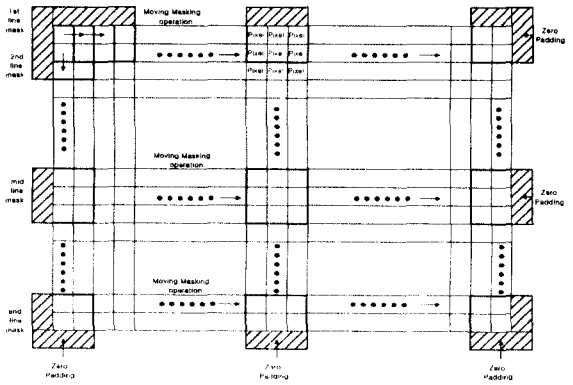


**그림 7** Unsharp Masking 연산

**Fig. 7** Unsharp Masking Operation

그림8 은 R, G, B block 중 R에 대한 module 이다. 그 외 G, B module들의 연산은 R과 같다. 또한, 그림8은 3개의 block으로 크게 나눌 수 있다. 첫 번째 블록은 memory controller와 2개의 FIFO 구조의 line buffer와 1개 register 이다. Unsharp masking 연산의 실시간 구현을 위해 3줄의

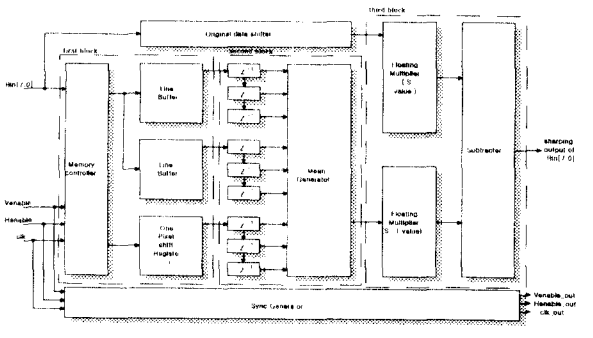


**그림 8** Unsharp Masking 연산 모듈

**Fig. 8** Unsharp masking operation module

line buffer 가 필요하나, 본 논문에서는 2개의 line buffer 와 1개의 register 을 가지고 구현하였다. 왜냐하면, 3×3 의 masking 연산을 하기 위해서 2줄의 line를 먼저 기억시키고, 마지막 line 은 실시간으로 들어오는 데이터를 다음 block에서 사용하면 되기 때문이다. 그리고, 그 다음 line의 데이터가 들어오면, 이전의 기억했던 두 줄 중에 첫줄의 line buffer의 데이터들을 지우고, 들어오는 데이터를 차례대로 기억시키고, 또 그 다음 line의 데이터가 들어오면 두 줄 중에 둘째 줄에 기억하고 있던 line buffer 의 데이터를 지운 후, 들어오는 데이터를 기억시키면 된다. 이렇게 첫 번째 line buffer와 두 번째 line buffer를 번갈아 가면서 저장하고, 지우면서, One pixel shift Register block에는 계속해서 들어오는 새로운 데이터를 받아서 보내면, 모든 line의 data가 masking operation을 위해 쓰일 수 있다. 이런 동작 제어는 memory controller에서 담당하게 되는데, 이 memory controller에 대한 state diagram은 그림9 에 나타나있고, 상태도의 천이는 Henable신호가 0에서 1로 바뀔 때 일어난다.

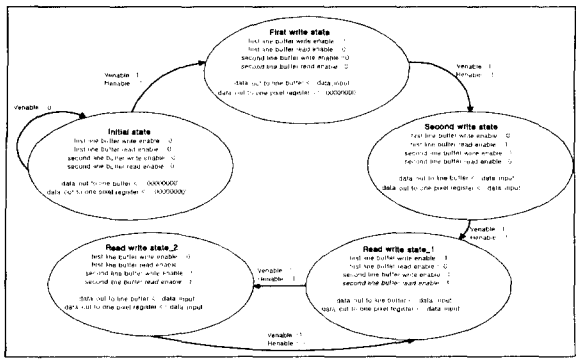


**그림 9** Unsharp masking의 메모리를 위한 상태도
**Fig. 9** State Diagram of memory controller for unsharp masking

그림9의 첫 번째 state는 frame과 frame 사이에는 모든 데이터가 zero로 setting된다는 것을 말하여 주는 것이며, 두 번째 state는 첫 번째 Henable 이 들어오면, 먼저 line buffer쪽으로 데이터 8bit 보내는 동시에, 첫 번째 line buffer에 write enable 신호를 generate 한다. 세 번째 state는 두 번째 Henable이 들어왔을 때, 두 번째 line buffer 에 대해 write enable 신호를 generate하여 데이터 저장시키고, 첫 번째 line buffer에 read enable 신호를 generate 하여, 첫 번째 line buffer쪽으로 보냈던 데이터를 출력한다. 또한, one pixel shift register쪽으로는 2번째 line buffer로 보냈던 데이터를 같이 보낸다. 왜냐하면, 처음의 zero padding 위해 첫 번째 masking line은 2줄의 데이터 와 1줄의 zero 데이터를 포함하고 있어야 하기 때문이다. 이렇게 하면 처음에 2 line 이 출력되고, 첫줄의 masking 연산이 수행되는 것이다. 네 번째 state는 세 번째 Henable 이 들어오면, 두개의 line buffer에 기억되어 있던 데이터들과, 현재 실시간으로 들어오는 데이터를 one pixel shifter register 쪽으로 보내 출력시켜 3줄의 데이터가 나가도록 하며, 현재 들어오는 데이터는 첫 번째 line에 기억시킨다. 이렇게 하면 두 번째 줄의 masking 연산이 수행되는 것이다. 마지막 state 인 다섯 번째 state 역시 두개의 line buffer에 기억되어 있던 데이터들과, 현재 들어오는 데이터를 one pixel shifter register 쪽으로 보내 출력시켜 3줄의 데이터가 나가도록 하며, 이렇게 현재 들어오는 데이터는 두 번째 line buffer 에 기억시킨다. 그 다음은 Henable이 바뀔 때마다 계속 해서 네 번째 state 와 다섯 번째 state 의 반복을 하여주면, 모든 PDP 상의 frame의 line 의 data를 unsharp masking 하는데 사용할 수 있다.

그림8의 두 번째 블록은 첫 번째 블록에서 3줄씩 나오는 data를 3×3 masking 연산을 하기 위해 line 마다 3개의 Z 1을 사용하여, mean generator block로 넘긴다. 출력되는 9개의 data 는 mean generator block에서 모두 더해지고, 평균값이 출력되어 수식(4)의 $avg_{neighbor}()$이 계산된다.

그림8의 세 번째 block 은 두 번째 블록으로부터 3×3 의 $avg_{neighbor}()$와 3×3의 정 중앙의 데이터를 함유하고 있는 Original data shifter와 수식(4)의 뺄셈을 수행한 후 최종 값을 출력한다. 여기에서 s 값은 1.125로 결정하여, 최종 실험 결과에 적용하였다.

### 2.2.2 제안하는 오차 확산 Module 설계

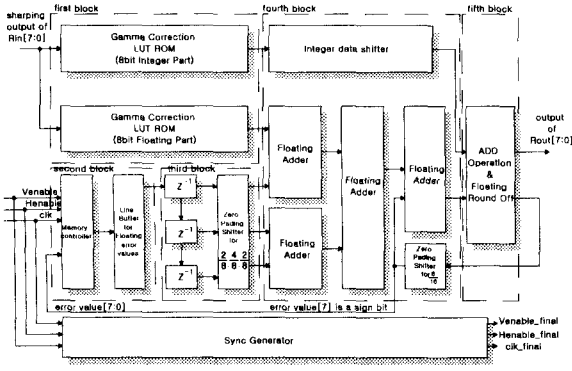


**그림 10** 제안한 오차확산 연산 모듈
**Fig. 10** The proposed error diffusion module

제안한 오차확산 알고리즘에 대한 흐름도는 그림3에 나타나 있으며, 그것에 대한 수식은 수식(2)에 잘 설명되어 있다. 그림10은 제안한 오차확산 알고리즘에 대한 하드웨어 module을 나타낸 것이다. 이 module은 크게 5개의 block으로 나누어져 있다. 그림10에 표시된 첫 번째 block은 input level(8bit)이 들어오면, gamma value 의 integer 값과 floating 값을 각각 출력하는 ROM table을 말한 것이다. 여기에서 integer는 8bit으로 0~255 까지 표현하도록 하였으며, floating 값 역시 8bit으로 소수점 두 자리까지 표현하도록 하였다. 두 번째 block 은 memory controller 와 line buffer 인데, 이것의 역할은 앞의 unsharp masking 에서와 같이 오차 확산을 수행 할 때, 아래쪽으로 전파되는 오차값을 저장하기 위해서 필요한 블록이다. 따라서, memory controller에서는 line buffer에 아래 line으로 전파되는 오차값들을 기억시킨 후, 다음 line의 화소에 오차값을 정확히 assign 되어 더해질 수 있도록 제어해 준다. 세 번째 block

은 오차 확산에서 아래쪽의 3부분의 오차를 전파 할 수 있도록 3개 delay Z-1을 주어, 한 개의 오차가 세 부분으로 나누어진 후, 다음 block에 $\frac{2}{8}, \frac{4}{8}, \frac{2}{8}$를 곱하는 block이다. 왜냐하면, 앞단인 zero padding shifter 에서 $\frac{8}{16}$를 하였기 때문에, 이들 계수들은 2배씩이 커지게 되었다. 또한, zero padding shifter란 이름은 수식(2) 상으로는 이들 계수들을 곱하여야 하나, 모든 floating point를 decimal로 표현한 후, 8 bit 중 최상위 bit(MSB)은 sign bit으로 하고, 7개 bit들을 floating 의 decimal bit으로 하면, -1.27 ~ +1.27 의 소수점 2자리까지 표현된다. 따라서, $\frac{8}{16}$ 의 계산은, MSB(sign bit)는 그대로 위치시킨 후, 7번째 bit에 zero를 padding 하고 0번째 bit를 버리면서, 7~1 번째 bit들을 LSB쪽을 1 bit shift 시켜 구현하였다. 이것은 곱셈을 단순한 shifter로 converting 하여 계산량을 줄이는 역할을 할 뿐 아니라, 마지막 block 의 delay를 줄이기 위해 사용되었다. $\frac{2}{8}, \frac{4}{8}, \frac{2}{8}$의 구현 역시 마찬가지로 각각 2bit, 1bit, 2bit를 shift 시켰다. 네 번째 block은 adder block으로 바로 위 line에서 전파된 3개 오차값과 옆 화소에서 전파된 오차값을 역 감마 보정 값의 integer 와 floating 값에 더하는 block 이다. 다섯 번째 block은 이렇게 더해진 모든 값을 반올림시켜, 가장 가까운 정수값으로 출력하여, PDP에서 표시 할 수 있게 한다. 여분의 block으로써, sync generator는 실제의 output를, 들어왔던 input과 같은 동기신호 형식으로 나갈 수 있게, Venable, Henable 와 clk를 다시 제 조정하여 맞추는 block 이다.

### 2.3. 실험 결과

#### 2.3.1. Software 상의 실험 결과 와 분석

표2는 실제 80×80의 단일 계조 영상을 만들어 각각의 계조값에 대해 나타나는 실제의 이상적인 역 감마값(①)과, PDP에 ①값을 보정 없이 그대로 적용했을 때의 출력값인 Actual output(②)과, 제안한 오차확산 방법을 적용한 후의 입력값에 대한 공간상의 output 값(③) 과, ①과 ③의 차이 값을 나타내는 오차값(① ③)을 나타냈다. 오차확산 방법을 을 적용한 경우 실험에 사용된 모든 영상들에서 평균값이 원 영상과 일치한다. 즉, 표2에 나타난 오차값은 1 frame 에 대한 4개의 측면 위치에 대해 오차확산을 하지 못하기 때문에 나타나는 오차값일 뿐이고, 실제로는 오차 확산 방법에 대한 오차는 0 %이다. 또한 하드웨어 구현 시 소수점 2자리까지의 역 감마값을 가지고 있기 때문에 어떤 영상에 대해서도 계산상의 공간적인 오차 값은 거의 무시할 수 있다.

그림11은 위의 표2에서 사용한 80×80의 영상에 대해 어두운 영역에 대한 출력 계조값을 그래프로 나타낸 것이다. 그림11에서 직사각형(■)표시는 이상적인 역 감마 보정 값이고, 원(●) 표시는 제안한 오차확산 방법을 적용하였을 때 값이며, 세모(▲)표시는 이 둘간의 차이를 나타내는 것이며, 엑스(×)표시는 역 감마 보정을 아무런 연산 없이 PDP에서 그대로 출력하여 의사윤곽이 발생되는 것을 보여주는 것이다. 그림11에서는 이렇게 제안하는 알고리즘과 이상적인 역 감마 보정사이의 오차값이 거의 제로로 나타나 있는 것을 말해주고 있고, 이는 제안한 알고리즘이 어두운 영역에서 의사윤곽을 감소시킨다는 결과를 보여주고 있다.

**표 2** 제안한 오차확산 방법을 적용한 영상의 입력결과

**Table 2** The result of example image applied the proposed error diffusion

| 입력 | 이상적인 역감마값① | Actual output② | 제안한 오차확산 방법 적용후③ | 오차값 (①-③) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0.011879 | 0 | 0.006944 | 0.00494 |
| 2 | 0.041364 | 0 | 0.036389 | 0.00498 |
| 3 | 0.085819 | 0 | 0.079722 | 0.00610 |
| ... | ... | ... | ... | ... |
| 10 | 0.749493 | 1 | 0.750278 | -0.00078 |
| 11 | 0.889764 | 1 | 0.894722 | -0.00496 |
| 12 | 1.040624 | 1 | 1.035556 | 0.00507 |
| ... | ... | ... | ... | ... |
| 20 | 2.609887 | 3 | 2.611944 | -0.00206 |
| 21 | 2.849460 | 3 | 2.854167 | -0.00471 |
| 22 | 3.098337 | 3 | 3.093611 | 0.00473 |
| ... | ... | ... | ... | ... |
| 254 | 253.2028 | 253 | 253.1989 | 0.00394 |
| 255 | 255 | 255 | 255 | 0 |

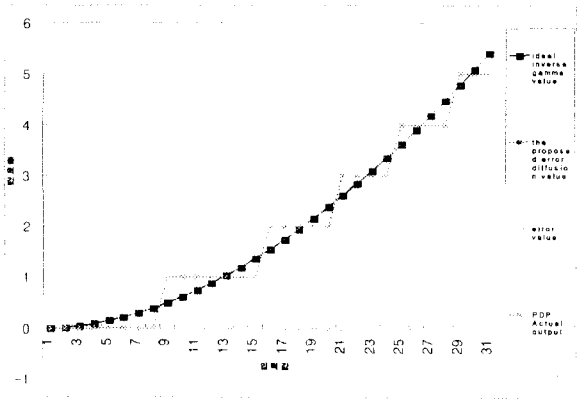

**그림 11** 어두운 영역에서 제안한 오차확산 결과와 이상적인 역 감마값 및 그것들의 차이

**Fig. 11** The result of the proposed error diffusion, ideal inverse gamma correction, and difference between them

#### 2.3.2. 하드웨어의 칩 specification & 보드 구현

제안한 오차확산 알고리즘을 적용한 image processor는 Altera 사의 FPGA (FLEX10K200EGC599-2)chip 과 line buffer로 사용되는 9개 NEC 40K buffer (unshap masking 에서 6개, 오차확산에서 3개)로 구성하여 실제 artwork 작업을 통해 보드로 직접 구현하였다. 표3은 이들에 대한 usage 와 specification을 나타내었다.

표 3 칩 구성요소와 사용량
**Table 3** Chip specification and Usage

| | FLEX10K20 0EGC599-2 | NEC 40K line buffer |
|---|---|---|
| Type | PGA(Pin Grid Array) | Plastic SOP(450mil) |
| Usage of Gates / The total Gates | 70,880 / 200,000 | |
| Usage of RAMbits / The total Gates | 82,944 / 98,034(option) | 8K / 40K per one |
| Usage of I/O pins / The total I/O pins | 210 / 470 | 24 |

또한, usage of RAM bits 에 대한 정보는 9개의 NEC 40K buffer를 사용하지 않고, FLEX10K200 의 Memory 에 구현했을 때에 사용된 RAM bits를 나타낸 것이고, 실제로는 ASIC 작업을 위해 FPGA의 RAM은 사용하지 않았다. 또한, 그림12는 이렇게 만들어진 보드에 대한 사진이다. 그림12에서 왼쪽 6개의 line buffer가 unsharp masking을 위해 사용된 것이고, 오른쪽 3개의 line buffer 가 오차확산을 위해 사용된 것이다. 또한, 위쪽이 input단과 아래쪽이 output 단으로 구성되어 있다.

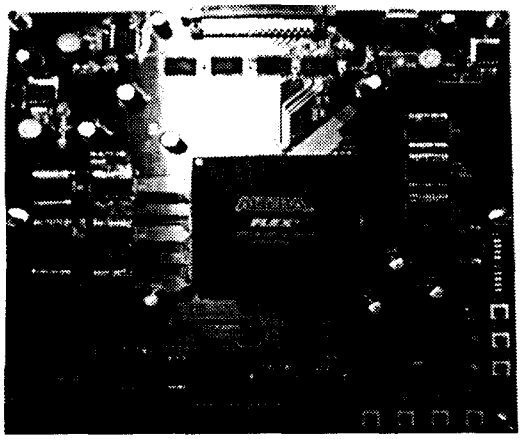

**그림 12** 실시간 영상처리기 기판
**Fig. 12** The real time image processor board

### 2.3.3. 하드웨어 상의 시뮬레이션 결과 및 측정결과

그림13은 top block 의 전체 시뮬레이션 파형이다. 실제 PDP 상에서는 DVS(frame enable)에 DEN(line data enable)이 480 개가 들어가고, clk은 33MHz 로 동작하여, 853개의 데이터가 한 DEN에 들어가게 되어 있지만, 시뮬레이션 상으로는 DEN는 2, 3 개를 넣고, 40개의 데이터를 한 DEN에 넣고, 33Mhz의 clk 상에서 정확히 assign 되는지를 확인하였다.

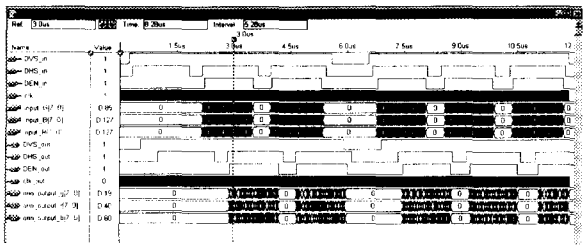

**그림 13** Maxplus2를 이용한 전체 모의실험 결과
**Fig. 13** The total simulation result using Maxplus2

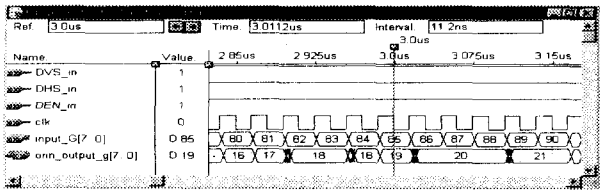


**그림 14** Maxplus2를 이용한 G channel의 모의 결과
**Fig. 14** The simulation result of G channel using Maxplus2

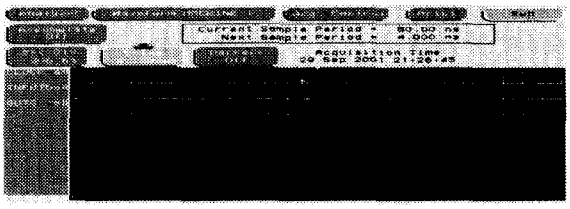

**그림 15** 논리분석기를 이용한 G channel 의 측정결과
**Fig. 15** The measurment result of G channel using Logic Analyzer

이것에 대한 확대 그림은 그림13에 있으며, 그림14는 RGB 중 G에 대한 의사윤곽이 심하기 때문에, 이것에 대한 확대 그림을 추가하였고, 그림15는 실제 보드를 연결한 후, 논리 분석기(HP1663A)를 사용하여 이 G channel을 측정한 결과이다. 보는 봐와 같이 시뮬레이션과 실제 측정 데이터가 잘 일치함을 알 수 있다.

### 2.3.4. 실제 PDP에서의 영상표시 특성

그림16과 그림17 는 실제 PDP에서 회색조 램프 영상에 대해 감마값 1.8 을 적용하여 표시된 결과를, 디지털 카메라로 촬영한 결과이다. 그림16에서는 의사윤곽 현상이 현저히 드러나는 반면, 그림17에서의 제안한 오차확산 알고리즘을 적용한 결과에서는 이러한 의사윤곽현상이 현저해 줄어드는 것을 볼 수 있다. 이와 같은 결과는 2.3.1절에서 설명한 소프트웨어상의 결과와 일치하는 것이다.

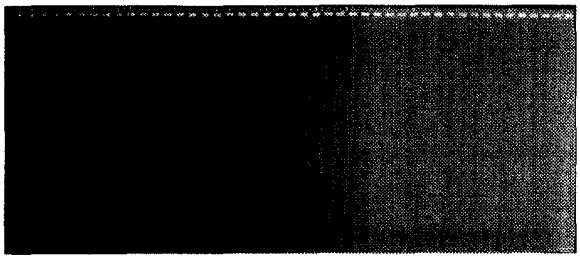

**그림 16** 어떠한 연산작용을 수행하지 않았을 때
**Fig. 16** Without additional processing

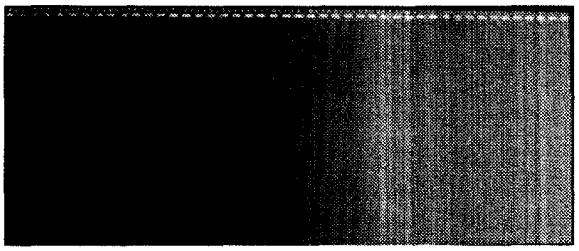

**그림 17** 제안한 Error diffusion 방법을 적용하였을 때
**Fig. 17** With the proposed error diffusion operation

## 3. 결 론

본 논문에서는 PDP에서 어두운 영역의 계조 재현 능력을 향상시키기 위해, 오차 확산 방법을 제안하였고 제안된 알고리즘을 실제로 FPGA를 이용한 실시간 image processor로서 구현하였다. 각 블럭에 해당하는 계산량을 최소로 줄이는 실시간 처리 위해, 역감마 보정 과정은 간단하게 구현할 수 있는 LUT ROM 으로 구현하였으며, 옆 화소로의 오차 확산은 1개의 feedback 부분으로, 아래 라인에 위치한 화소들에 대한 오차확산은 1 line 의 외부 RAM으로 구현하여 실시간 처리가 가능하도록 하였다. 이러한 오차확산 방법은 floating연산을 통해 오차확산이 이루어지기 때문에 더욱 정확한 계조 표현이 가능하도록 하였다. 빠른 시간에 결과를 볼 수 있도록 VHDL를 이용한 FPGA 보드를 직접 제작하여, 하드웨어 상의 성능을 측정하고 평가하였다.

**감사의 글**

본 논문은 (주)오리온 전기의 지원으로 수행된 PDP 화질 개선을 위한 과제 결과의 일부분입니다. (주)오리온 전기의 과제 지원에 감사하며 실험에 조언을 주신 (주)오리온 전기의 김상규, 김민철님께 감사 드립니다. 또한, CAD tool과 관련하여 IDEC의 부분적 지원에 감사 드립니다.

## 참 고 문 헌

[1] : L. F. Webber, "Plasma displays", IEEEInternational Plasma Science Conference, pp. 140, June 1996.

[2] : R. A. Salmon, "Progress in plasma display panels for HDTV",IEE International BroadcastingConvention Conference Publication, no.428, pp. 506-511, eptember 1996.

[3] : T. Shinoda, "Color Plasma Displays Bringing A Large Area Flat Panel Display World", Lasers and Electro Optics, Pacific Rim Conference, pp. 2-3, July 1997.

[4] : Poynton, "A Technical Introduction to Digital Video", John Wiley & Sons, pp.92-114, 1996.

[5] : K. Nunomura, "Advance of Picture Quality in Color AC Plasma Display Panels", Display Research Conference, Conference Record of the 1997 International pp. 499-502, October 1997.

[6] : R. C. Gonzalez and R. E. Woods, "Digital Image Processing", Addison Wesley, pp.196-197,1992.

[7] : Randy Crane, "A Simplified Approach to image processing", PrenticeHall, pp.166-168,1997.

[8] : 김춘우, 박승호, "플라즈마 디스플레이 패널(PDP)에서의 어두운 영역 표현을 위한 오차 확산방법", 대한민국특허 출원 제2000-52200호, 2000. 9.

# 저 자 소 개


**이 창 훈 (李 昌 勳)**

1973년 11월 6일생. 2000년 인하대 전자재료공학과 졸업. 2000년~현재 동 대학 정보통신대학원 석사과정

Tel : 032) 860 8618
Fax : 032) 875-5882
E-mail : c2001010@inhavision.inha.ac.kr

**강 진 구 (姜 晋 球)**

1961년 2월 27일생. 1983년 서울대 졸업. 1996년 North Carolina 주립대학 졸업(공박). 1997년~현재 인하대 전자.전기컴퓨터 공학부 조교수

Tel : 032) 860-7763
Fax : 032) 875-5882
E-mail : jkang@inha.ac.kr

**박 승 호 (朴 勝 虎)**

1974년 1월 25일생. 1996년 인하대 전기공학과 졸업. 1998년 동 대학원 전기공학과 졸업(석사). 1998년~현재 동 대학원 전기 공학과 박사 과정

Tel : 032-860-7401
Fax : 032-863-5822
E-mail : wintiger@hotmail.com

**김 춘 우 (金 椿 宇)**

1961년 2월 14일생. 1983년 서울대 제어계측공학과 졸업. 1989년 Purdue 대학교 전기공학과(공박). 1994년~현재 인하대학교 정보통신 공학부 부교수

Tel : 032-860-7401
Fax : 032-863-5822
E-mail : cwkim@inha.ac.kr