論文2000-37SD-5-9

On Screen Display용 자막처리 ASIC 설계

(Design of Caption-processing ASIC for On Screen Display)

鄭 根 泳 *, 禹 鍾 植 *, 朴 鍾 仁 *, 朴 柱 成 *, 朴 鍾 錫 **

(Geun-Young Jeong, Jong-Sik Woo, Jong-In Park, Ju-Sung Park, and Jong-Seok Park)

요 약

본 논문은 가요반주기의 OSD(On Screen Display)에 필요한 영상·자막처리 ASIC의 설계에 관한 내용을 기술한다. 기존의 자막처리는 범용 DSP를 이용하여 소프트웨어적으로 처리되었으나, 본 논문에서는 하드웨어 비용을 절감할 수 있는 ASIC을 설계하였다. 설계된 자막처리 ASIC의 주요기능은 외부로부터 명령코드와 함께 영상 및 자막 데이터를 받아 여러 영상효과를 가하여 화면으로 출력하는 것이다. 전체적인 설계는 Compass tool에서 schematic으로 설계되었고 부분적으로 VHDL로 코딩하였다. 설계된 ASIC은 로직 시뮬레이션을 통하여 일차적으로 검증한 후, FPGA를 이용하여 실제 시스템에 응용하여 최종 점검을 하였다. 칩은 0.8㎛ CMOS 공정을 활용하여 제작하였으며, 제작된 칩은 가요 반주기에서 원하는 기능을 수행하는 것을 확인하였다.

Abstract

This paper describes design and implementation of caption-processing ASIC(Application Specific Integrated Circuits) for OSD(On Screen Display) of karaoke system. The OSD of conventional karaoke system was implemented by a general purpose DSP, however this paper suggest a design to save hardware resources. The ASIC receives commands and data of graphic and caption from host processor, and then modifies the data to have various graphic effects. The design has been done by schematic and VHDL coding. The design was verified by logic simulation and FPGA emulation on the real system. The chip was fabricated with 0.8㎛ CMOS SOG, and worked properly at the karaoke system.

* 正會員, 釜山大學校 電子工學科

(Dept. of Electronic Engineering, Pusan National University)

** 正會員, (株)보이소半導體

(VOISO Semiconductor Co., Ltd)

※ 본 연구는 산업자원부 산업기반기술개발 사업의 연구과제를 수행하는 (주)보이소반도체의 위탁연구비에 의해 수행되었음.

接受日字:2000年2月23日, 수정완료일:2000年4月25日

Ⅰ. 서 론

반도체 및 통신 기술의 급격한 발전으로 이제는 멀티미디어 시대에 이르고 있고 음성은 기본이고 대부분의 시스템에 시각적인 정보를 필요로 한다. 이러한 영상정보의 이해를 시각적으로 보다 높여주는 것 중의 하나가 OSD(On Screen Display)장치이다. OSD장치는 자막이나 영상을 바탕이 되는 본 영상 위에 중첩하여 출력하는 것으로 단순히 자막이나 영상을 원형 그대로 출력하기도 하고 적당한 영상효과를 더해 출력하기도

한다. OSD의 활용 분야는 텔레비전, 비디오, 모니터, 비디오 카메라 등 다양한 영상 기기에 활용되고 있다.

기존의 OSD장치에서 영상 및 자막처리는 주로 범용 DSP(Digital Signal Processor)에 의해 이루어지고 있다. 주어진 영상 데이터를 처리하고 저장하고 출력하는 기능은 DSP 명령어의 조합으로 이루어진 프로그램에 의해 수행된다. 즉 외부에서 어떤 기능의 동작을 요구하면 그 기능에 해당되는 프로그램이 수행되면서 영상 데이터를 처리하게 된다. 그러므로 하나의 기능을 구현하기 위해 수 개 또는 수십 개이상의 명령어가 필요하다. 또한 이런 기능들의 동작은 영상 데이터의 화소 수만큼 반복되기도 하므로 그 효율성이 더욱 떨어지기도 한다. 여러 기능들이 동시에 실행될 때 프로그램의 복잡도는 높아지고 수행속도도 떨어진다. 연산과정 측면에서도 복잡하고 현란한 영상효과가 필요치 않는 OSD 장치에서 영상처리의 연산은 비교적 단순하고 반복적이므로 복잡한 연산을 고속으로 처리가 가능한 고가 DSP는 낭비라고 할 수 있다.[1] 따라서 가요반주기와 같이 비교적 간단한 영상효과가 요구되는 장치를 위한 OSD ASIC이 필요하다.

하드웨어를 구성함에 있어서 프로그램드(programmed) 방식은 하드와이어드(hardwired) 방식에 비하여 융통성은 많으나 하드웨어 자원의 낭비가 많고 속도가 느리다는 문제점이 있다.[2-4] 본 논문에서 하드와이어드 방식을 채택하여 설계된 ASIC은 영상 및 자막처리에 필요한 기능별로 명령어를 할당하여 한번의 명령어 입력으로 하나의 기능이 완전히 실행 가능하게 하였고, 배경영상의 비디오신호와 동기를 맞추는 장치는 다른 기능들의 수행에 영향을 받지 않도록 독립적으로 구현하였다.[5] 또한 시각적인 효과는 뛰어나면서도 가감산의 연산만으로 구성된 알고리즘으로 다양한 영상효과를 낼 수 있게 하였다. 범용 DSP를 사용하는 기존 시스템에서 필요한 FIFO 메모리와 단순한 디지털회로들을 칩 내부에 내장시켜 시스템의 보드를 간편하게 하였다. 그리고 실제 주된 응용 시스템은 가요반주기임을 고려하여 영상보다는 노래가사의 시각적인 효과를 잘 살릴 수 있는 방향으로 설계를 하였다. 즉 자막데이터는 자음·모음을 단위로 분리해서 받을 수 있게 하여 한 글자가 구성될 때 각 자음·모음의 경계가 서로 닿아 영상 데이터의 손실을 유발할 수 있는 가능성을 제거할 수 있는 기능을 기본 기능 외에 추가하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 OSD장치에서 영상처리 방법, Ⅲ장에서는 영상처리 효과, Ⅳ장에서는 매크로 명령어 기능, Ⅴ장에서는 구조 및 설계, Ⅵ장에서는 설계 검증 및 테스트, Ⅶ장에서는 결론의 순서로 논하겠다.

Ⅱ. OSD장치에서 영상처리 방법

보통의 OSD장치는 주어진 영상이나 자막을 그대로 바탕이 되는 화면 위에 중첩시켜주는 역할을 하는 것이고 본 논문에서 설계된 자막처리 ASIC은 OSD기능에 영상이나 자막에 시각적인 효과를 가할 수 있는 기능이 추가되었다. OSD장치의 주된 구성요소는 그래픽 메모리, 메모리 컨트롤러, 그래픽 프로세서, 디스플레이 프로세서, 디스플레이 인터페이스로 구성된다. 그래픽 메모리는 영상 데이터를 저장하는 역할을 하고 두 영역으로 나누어서 한 쪽은 영상처리를 위한 임시저장공간으로 다른 한 쪽은 디스플레이 출력용으로 사용한다. 메모리 컨트롤러는 내부 프로세서들이 메모리를 사용할 수 있게 메모리에 제어신호를 발생시킨다. 그래픽 프로세서는 영상 데이터의 화면출력 이전에 화소의 위치나 색의 변화를 일으키는 연산을 맡아 영상처리를 한다. 이러한 영상처리를 위해서는 항상 임시적인 저장 메모리가 필요하다. 그래픽의 경우 대용량이므로 내부에 메모리를 포함시키기가 힘들므로 외부에 장착하게 된다. 이 때문에 외부의 메모리를 제어하기 위한 메모리 컨트롤러가 필요하다. 이렇게 하면 영상 데이터를 발생시키는 구조는 거의 완성되나 외부 화면으로 출력하기 위한 프로세서가 필요하다. 이러한 역할은 디스플레이 프로세서가 맡는다. 이는 그래픽 프로세서의 동작과는 독립적으로 그래픽 메모리에 있는 영상을 디스플레이 인터페이스 부로 출력시키는 것이다. 디스플레이 프로세서는 영상처리의 일부분도 담당하는데 그래픽 메모리로부터 영상 데이터를 읽어내어 디스플레이 인터페이스로 넘겨주는 과정에서 영상화소를 처리하거나 위치를 옮기는 기능을 할 수 있다. 디스플레이 인터페이스에서는 메모리(RAM)와 , DAC(Digital Analog Converter)로 구성되어 메모리의 디지털 출력을 바로 아날로그 신호로 바꿔주는 RAMDAC장치에 의해 그래픽 메모리의 영상 데이터가 아날로그 신호로 출력되어 비디오 신호가 된다.[6]

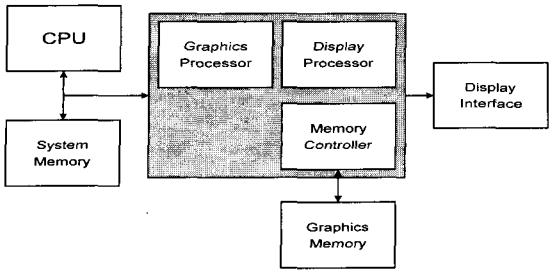

그림 1. 일반적인 영상처리 구조도
Fig. 1. Structure of a general graphics processing.

데이터가 처리되는 과정을 살펴보면 그림 1과 같이 외부 CPU에 의해 영상처리 명령어와 데이터를 먼저 내부 그래픽 프로세서로 보내면 영상 데이터는 메모리 컨트롤러를 통하여 그래픽 메모리에 일단 저장된다. 그 후에 입력되는 다른 명령어의 수행으로 그래픽 메모리 내의 영상 데이터는 영상처리과정을 거쳐 영상효과를 일으킨다. 이와 동시에 디스플레이 프로세서에 의해 영상효과가 가해진 영상은 디스플레이 인터페이스를 거쳐 최종적으로 화면으로 출력된다.[7]

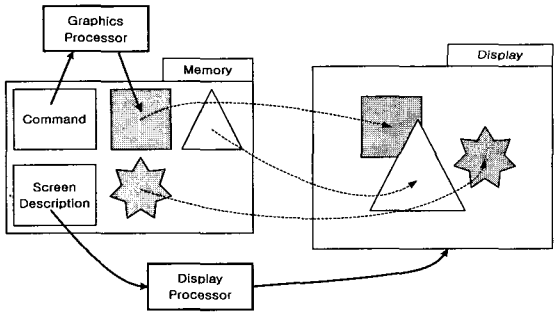

그림 2. 그래픽 프로세서와 디스플레이 프로세서의 기능
Fig. 2. Function of graphic processor and display processor.

그림 2는 그래픽 프로세서와 디스플레이 프로세서의 역할을 나타낸 것으로 그래픽 프로세서는 입력된 명령어 순서에 따라 영상 데이터를 처리하여 메모리에 저장한다. 그러면 디스플레이 프로세서가 화면제어 명령에 따라 일정 메모리영역을 화면영역으로 옮기게 된다. 이 때에 디스플레이 프로세서에 의하여 특정위치로 이동하거나 크기의 비를 조절하여 확대나 축소가 가능하고 어느 영역이 앞으로 나오는가와 또는 가려지는 가를 결정할 수 있다.[7]

일반적인 영상이 아니고 자막일 경우에는 먼저 자막 데이터를 영상으로 만들어 주어야한다. 그림 3에 나타낸 자막처리 과정을 살펴보면 전체 시스템 제어 프로세서가 먼저 font ROM으로부터 자막 데이터를 자음·모음 단위로 읽어서 font decoder로 보낸다. Font decoder는 자막 데이터를 해석하여 자모의 위치 및 자모형태를 구성한다. 그 다음 그래픽 프로세서는 구성된 자모가 실제 화면에 출력될 수 있도록 화소 단위로 색을 주어 영상 메모리의 해석된 위치에 저장시킨다. '가' 글자를 예로 들면 'ㄱ' 자음이 앞의 과정을 통해 영상 메모리에 저장되고 'ㅏ' 모음을 'ㄱ' 자음이 차지한 영역의 영상 데이터를 피해 적절한 위치에 저장시킨다. 그 후 영상 메모리에 저장된 자막은 일반적인 영상으로 취급된다.

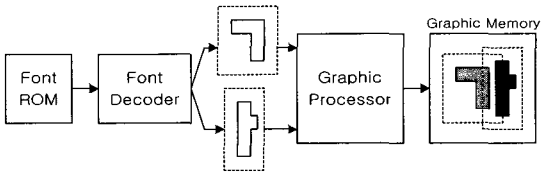

그림 3. 자막처리 방법
Fig. 3. Method of font processing.

범용 DSP를 이용한 기존의 자막처리의 구조는 그림 4에 나타내었다. 이 구조는 DSP와 ROM, RAM, FIFO, 비디오 제어회로로 구성된다. 전체적인 동작은 ROM에 저장된 프로그램에 의해 DSP와 RAM을 통해 수행된다. 먼저 FIFO를 통해 수행할 명령이 인가되면 프로그

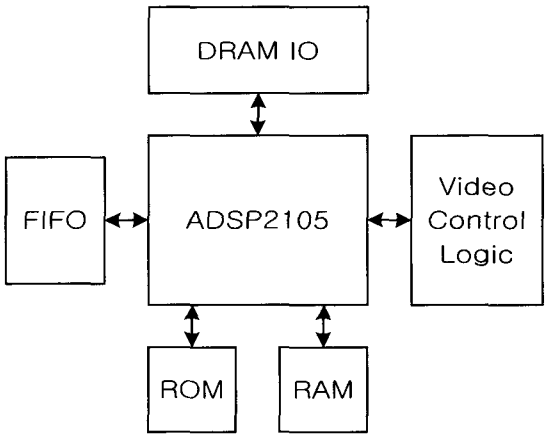

그림 4. 범용 DSP를 이용한 자막처리 구조도
Fig. 4. Architecture of font processing with DSP.

램중 한 루틴(routine)이 수행되면서 DRAM과 비디오 신호를 제어하게 된다. 본 논문에서는 이러한 소프트웨어적인 구조를 하드와이어된 하나의 ASIC으로 구현하였다. 각 독립요소에 대응되는 칩들은 하나의 칩으로 구현되므로 시스템 면적은 이전의 4분의 1로 줄일 수 있고 복잡도와 비용 또한 이에 상응하여 줄게 된다. 단 영상처리 효과의 기능 및 종류는 하드웨어적으로 고정되어 있으므로 차후에 수정 및 추가는 힘들게 된다.

Ⅲ. 영상처리 효과

설계된 자막처리 ASIC은 원 영상의 시각적인 효과를 높이기 위해 다양한 영상효과 기능을 갖고 있다. 자막처리 ASIC에서 연산기능은 가감산만 가능하고 또한 저장공간이 한정되어 있으므로 모든 효과는 알고리즘이 간단하고 가감산만으로 실행이 가능하도록 해야한다. 이렇게 고안된 영상처리 효과는 크게 스크롤(scroll), 확대/축소, 펼침, 꼬리끌기 네 종류이다. 회전 효과는 디스플레이 프로세서에 의한 것이고 나머지 효과는 그래픽 프로세서를 이용하여 구현한다.

1. 스크롤 효과

스크롤 효과는 영상을 화면상에 평면으로 스크롤시키는 것으로 다음 그림 5~7과 같이 수평방향 스크롤, 수직방향 스크롤, 그리고 화면영역 밖에서 안으로 움직이는 효과(Move-on)이다. 스크롤 효과의 경우는 나머지 효과와 달리 실제 그래픽 메모리영역에서 데이터의 이동이나 복사는 없다. 그래픽 메모리에서 화면으로 출력되는 과정에서 디스플레이 프로세서가 기준 포인터를 이동하여 효과를 일으킨다. 또한 스크롤 효과는 화면전체 스크롤과 블록 스크롤이 있다. 말 그대로 화면전체 스크롤은 전체 화면영역이 동시에 스크롤이 되는 것이고 블록 스크롤은 특정영역의 내부만 스크롤하는 것이다. 두 방식 모두 세로방향과 가로방향 스크롤이 있고 가로와 세로 두 방향으로 동시에 효과를 줄 수 있다. 그림 8은 왼쪽 방향의 블록 스크롤이 되는 경우를 나타내고 있다. 블록 스크롤이 되는 영역의 화소 수는 가로 3에 세로 3이며 총 4 프레임만에 원 영상으로 돌아온다.

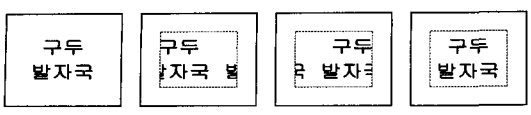

그림 5. 수평 방향 스크롤 효과의 프레임 흐름
Fig. 5. Frame flow of horizontal Scroll effect.

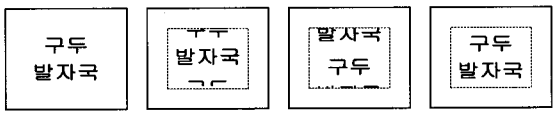

그림 6. 수직 방향 스크롤 효과의 프레임 흐름
Fig. 6. Frame flow of vertical Scroll effect.

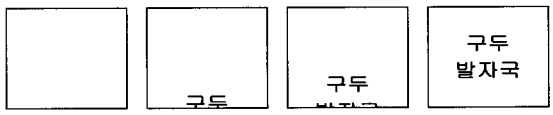

그림 7. Move-on 효과의 프레임 흐름
Fig. 7. Frame flow of Move-on effect.

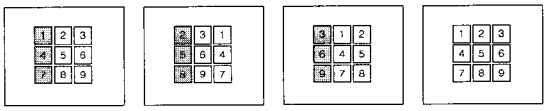

그림 8. 스크롤 효과의 프레임별 화소 이동
Fig. 8. Change of pixels in Scroll effect.

2. 확대/축소 효과

확대/축소 효과는 그림 9에서 나타낸 바와 같이 영상의 원 크기를 확대 또는 축소하여 나타내는 효과로 배율의 범위는 0.2~2 배 사이이다. 배율을 점진적으로 조절하면서 연속적으로 확대나 축소효과를 사용하면 그림 10과 같이 zoom-in이나 zoom-out같은 효과를 얻을 수 있다.[8] 그림 10은 1.5 배 확대효과를 적용한 경우이다.

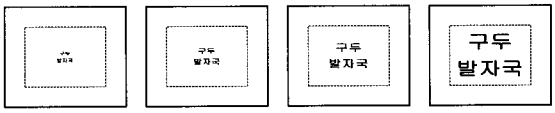

그림 9. 확대 효과의 프레임 흐름
Fig. 9. Frame flow of Zoom effect.

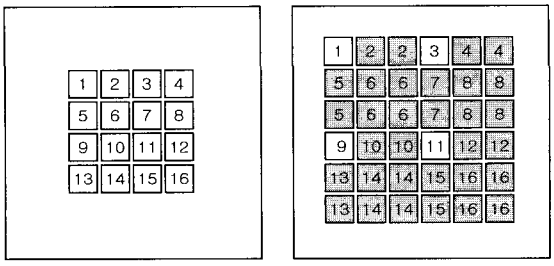

그림 10. 본 크기의 영상과 1.5 배로 확대된 영상
Fig 10. Original pixel image and Enlarged pixel image with ratio 1.5.

3. 펼침 효과

펼침 효과는 그림 11에서 나타낸 바와 같이 수평선상에서 반으로 나눠진 영상이 아래위로 미끄러지듯이 펼쳐지는 효과로 확대와 달리 실제 크기가 변하지는 않는다. 펼침 효과의 경우 중앙선을 기점으로 한 라인씩 위 아래로 밀려나오는 동작으로 그림 12에서 보면 처음엔 최 상단의 라인인 1, 2, 3번 화소와 최 하단의 라인인 10, 11, 12번 화소가 그 다음 프레임에서는 안쪽의 라인들인 4, 5, 6번 화소와 7, 8, 9번 화소가 나와서 원 영상이 복원되면 펼침 효과의 수행은 종료된다.

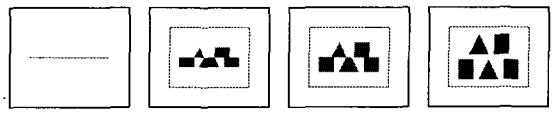

그림 11. 펼침 효과의 프레임 흐름
Fig. 11. Frame flow of Unroll effect.

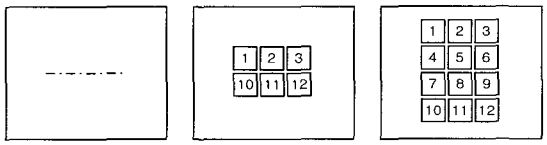

그림 12. 펼침 효과의 프레임별 화소 이동
Fig. 12. Change of pixels in Unroll effect.

4. 꼬리끌기 효과

꼬리끌기 효과는 그림 13에서 나타낸 바와 같이 영상의 전단부터 영상의 단면이 화면 오른쪽 끝까지 반복되며 마치 꼬리를 끌듯이 점차 영상이 완성되어 간다. 꼬리끌기의 과정은 그림 14와 같이 열 단위로 데이터가 복사되면서 그 복사 시점이 한 칸씩 이동하게 된다. 처음엔 1, 4, 7번 화소가 그 다음엔 2, 5, 8번 화소가 마지막으로 3, 6, 9번 화소가 복사된다. 블럭에서 한 열의 복사가 화면의 끝에 다다르면 그 시작점이 오른쪽으로 이동하게 되고 그 시작점 또한 영상의 끝에 이르면 꼬리끌기의 수행이 완료되게 된다.

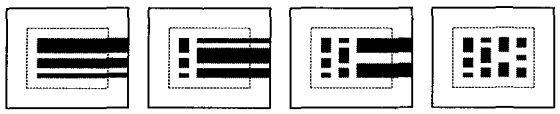

그림 13. 꼬리 끌기 효과의 프레임 흐름
Fig. 13. Frame flow of Tail effect.

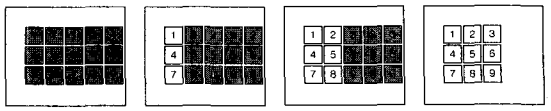

그림 14. 꼬리끌기 효과의 프레임별 화소 변화
Fig. 14. Change of pixels in Tail effect.

Ⅳ. 매크로 명령어 기능

Font-down, Change-color, Screen-clear, Color-palette-set, Image-forward, Effect는 매크로 기능을 하는 명령어들로 그 기능은 다음과 같다.

1. Font-down 기능

Font-down은 화면에 출력될 영상 및 자막 데이터를 외부로부터 받아 그래픽 메모리에 저장하는 기능이다. 원래 영상 데이터를 그대로 저장하는 기능 외에 가로나 세로로 두 배 확대해서 저장하는 기능이 있다. 영상 데이터는 영상의 최소 단위인 화소들의 집합으로 이루어진다. 여기서 각각의 화소가 가진 데이터는 전체 영상에 사용되는 색 중에 하나를 나타낸다. 그러므로 전체 영상에 동시표현이 가능한 색 수에 의해 화소의 비트 수가 정해진다. 여기서는 동시표현 색 수를 256색으로 한정하여 화소를 8비트로 나타내었다. 화소는 8비트 단위이고 사용된 그래픽 메모리의 데이터 입출력단위는 16비트이므로 한 번의 메모리 액세스(access)로 2개의 화소를 읽을 수 있다. 자막일 경우 2비트만으로 색 표현 수를 한정하여 16비트로 8개의 화소를 표현할 수 있다. 이 때는 미리 입력된 font color와 shadow color의 값에 의해 2비트인 값이 8비트인 정상 화소 값으로 변환되어 영상 메모리에 입력되게 된다. 따라서 자막데이터의 입력속도를 4배로 높일 수 있으므로 외부의 느린 데이터 입력으로부터 빠르게 자막데이터를 영상 메모리에 저장할 수 있다.

2. Change-color 기능

Change-color는 자막의 색깔을 이동하며 연속적으로 바꾸는 기능이다. 가요반주기에서 반주에 따라 가사의 색을 바꿀 때 주로 쓰인다. 이 명령의 수행은 그림 15에 나타낸 바와 같이 초기설정(set)과 실행(exe)으로 나뉜다. 먼저 설정 명령에서 초기 위치와 세로 크기, 현재 자막의 그림자 색과 바뀌질 색상 값이 설정된다. 설정이 되면 다음 설정 명령이 있을 때까지 실행 명령에 의해 한 열씩 색상이 변한다. 실행 명령은 가요반주에

맞추어 적절한 타이밍에 자막의 색을 바꿔주어야 하므로 다른 명령의 수행에 영향을 받지 않고 독립적으로 명령을 입력받을 수 있어야 한다. 따라서 실행 명령은 명령어 코드로 입력되지 않고 별도의 시리얼 포트(serial port)를 통해서 펄스 입력으로 인지한다. 연속적인 동작을 그림 15에서 보면, 처음 초기설정 후에 세 번의 실행 펄스가 입력되고 각 펄스마다 1, 2, 3열의 자막 색이 바뀌게 된다. 두 번째 초기설정 후에 두 번의 실행 펄스가 입력되어 4, 5열의 자막 색을 바꾼다. 실행 펄스는 자막에서 한 열의 자막 색을 변화시킨 후 자동으로 다음 열로 포인터를 옮기므로 연속된 펄스만으로 자막의 색이 왼쪽에서 오른쪽으로 옮겨가며 한 열씩 변하게 된다. 또한 이 펄스가 입력되면 화면출력 상태가 아닌 이상 모든 명령수행을 정지하고 Change-color 실행 명령을 수행한다.

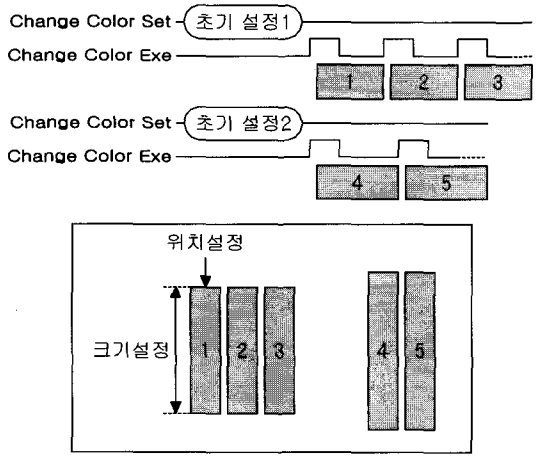

그림 15. Change-Color 기능 실행 과정
Fig. 15. Detail of Change color execution.

3. Effect 기능

Effect는 일단 Font-down이 수행되어 그래픽 메모리의 임시저장 영역에 저장되어 있는 영상 데이터를 디스플레이 영역으로 옮기면서 효과를 가해 주는 기능을 한다. 다만 스크롤의 경우 화면출력 과정 중에 일어나므로 실제의 데이터 이동은 없다. 내장된 영상효과는 스크롤, 펼침, 확대/축소, 꼬리끌기가 있다. 각각의 영상효과는 스크롤 효과와 동시수행이 가능하도록 설계하였다.

4. 기타 명령어 기능

Screen-clear는 대상이 되는 화면영역의 영상 및 자막을 지우는 기능이다. 여기서 지운다는 의미는 영상 메모리의 화소 값을 미리 약속된 데이터로 덮어쓴다(overwrite)는 것이다. 화면출력을 할 때 지워진 부분은 배경화면에 오버레이(overlay)되어 나타나지 않고 배경화면이 그대로 출력된다.[6,9] 이 기능에는 지울 영역의 위치, 크기정보와 덮어쓸 데이터가 따른다.

Color-palette-set은 RAMDAC장치의 color palette를 설정하는 기능이다. Color palette는 전체 영상에 사용되는 모든 색상들을 화소의 값에 대응되게 정해 놓는 것으로 영상이 바뀌거나 영상효과가 주어질 때 그 값들이 바뀌게 된다.[9] 설정 순서는 먼저 mask 값을 설정하고 뒤따를 RGB(Red Green Blue)색상 값의 개수와 palette상의 설정 시작 주소를 받는다. Palette의 주소는 256개이고 각각 6비트인 RGB로 구성되므로 데이터의 최대 개수는 256×3개이다.

Image-forward 기능은 그래픽 메모리의 임시저장 영역에 저장된 영상을 영상효과 없이 바로 디스플레이 영역으로 출력하는 명령으로 외부에서 느리게 전송되는 영상을 저장 완료 후 빠르게 화면으로 출력할 수 있다.[5]

명령어 기능 외의 중요 사양을 표 1에 나타내었다. 비디오 방식은 NTSC와 PAL겸용으로 NTSC의 화면을 확대하여 PAL방식으로 출력한다.

표 1. 중요 설계 사양
Table 1. Important specification of design.

항 목	시스템 사양
Clock 주파수	21.48 ㎒
비디오 방식	NTSC / PAL 겸용
비디오 영역	NTSC: 512×480 PAL: 512×560
비디오 메모리	4 MBit DRAM : 16 Bit×512×512
주 입력포트	8 Bit
FIFO 깊이	8 Bit × 128
Color Palette	256 Color R/G/B 각각 2 Bit
기능	OSD, 자막 및 영상처리
영상효과	스크롤, 펼침, 확대/축소, 꼬리끌기

V. 구조 및 논리 설계

자막처리 ASIC의 구조는 II절에서 소개한 OSD 영상처리장치의 구조를 바탕으로 하고 있다. 그림 16은

설계된 자막처리 ASIC의 구조를 나타내고 있다. OSD의 그래픽 프로세서 기능은 제어부에서 맡고 있고, 디스플레이 프로세서 기능은 제어부와 DRAM 주소 발생기에서 주로 수행하며, 메모리 인터페이스 기능은 DRAM 주소 발생기와 DRAM 데이터 입출력 장치가 맡고 있다.

외부 호스트 프로세서로부터 제공되는 데이터는 여러 동작들을 실행할 명령어와 각종 수치, 자막 및 영상 데이터를 포함하고 있다. FIFO에 잠시 저장된 이 값들 중 명령어는 디코더를 통해 해석되어 제어장치로 보내어 지고 데이터는 레지스터에 저장되거나 외부의 메모리로 바로 보내어 진다. 해석된 코드에 의해 제어장치의 해당 명령회로를 동작시킨다. 이 때 각종 제어신호는 화면에 대응되는 DRAM에 영상 및 자막 값들을 입출력시키거나 여러 효과를 일으킨다. RAMDAC에는 초기 RGB palette값들이 입력되게 된다. 디스플레이 제어회로는 이런 과정 중에도 항상 외부 비디오 신호와 동기를 맞추어 DRAM의 내용을 RAMDAC을 통해 화면으로 출력시킨다.[5]

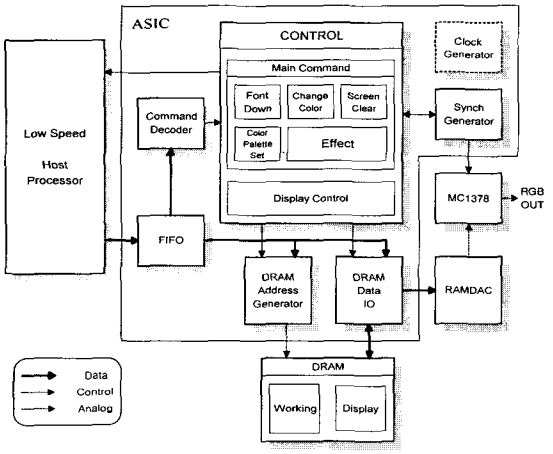

그림 16. 자막처리 ASIC의 구조도
Fig. 16. Architecture of Caption-processing ASIC.

1. 입력부

FIFO는 영상·자막처리기에 비해 느리고 불규칙적으로 동작하는 프로세서로부터 데이터를 일시 저장하여 공급하는 버퍼역할을 하는데 총 128단이고 8비트의 입출력을 갖는다. 외부는 8비트이고 내부는 16비트 체계이므로 FIFO 내부의 입력데이터 처리회로가 항상 한 쌍의 8비트 데이터를 16비트로 모아서 출력시킨다. 따라서 실제로는 8비트로 입력을 받아 16비트로 출력하게 되어있다. 또한 FIFO는 회로와는 독립된 외부용으로 하나를 더 내장하고있다.

명령어 디코더 회로는 입력된 데이터 중에서 명령어를 해석하여 그 명령어에 해당되는 매크로 명령을 수행하는 제어회로를 동작시킨다. 명령어 디코더에 입력되는 명령어의 구조를 그림 17에서 살펴보면 크게 상위 8비트와 하위 8비트로 나뉜다. 상위 8비트는 주 명령어이고 하위 8비트는 자막효과 명령어에서 효과의 종류를 구분하고 자막 데이터를 받아 저장할 때 확대 여부를 결정한다. 명령어는 각 비트가 바로 제어 신호로 쓰일 수 있게 비트별로 할당했다.

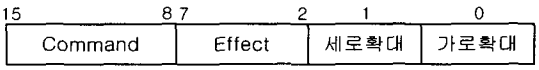

그림 17. 명령어의 구조
Fig. 17. Structure of instruction.

2. 출력·연산부

DRAM 주소발생 회로는 영상의 위치, 크기, 배율 등의 값들을 저장하는 레지스터와 카운터, ALU로 구성되어 있다. DRAM은 상시 화면으로 출력되는 디스플레이 영역과 임시 저장 영역으로 나뉜다. 이 각각의 영역은 다시 홀수 필드와 짝수 필드로 나뉜다.[9] DRAM은 수시로 refresh를 해줘야 하는데 이를 피하기 위해 빠른 주기로 주소 값이 증가하도록 DRAM의 row address를 배치하여 화면출력과 동시에 refresh가 가능하게 하였다. DRAM의 데이터 입출력은 16비트 단위이고 한 화소는 8비트 단위이므로 한 주소 값은 화면의 두 화소를 동시에 지칭한다. 따라서 그림 18과 같이 상위 8비

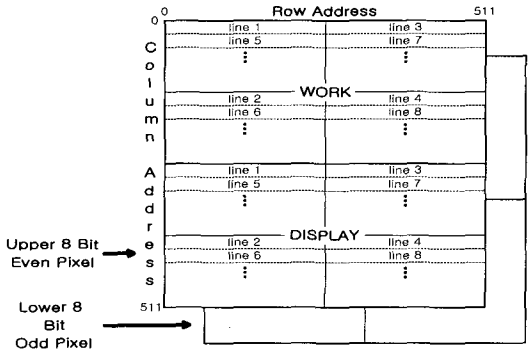

그림 18. DRAM 영역 분할
Fig. 18. Partition of DRAM region.

트는 짝수 번째 화소로 하위 8비트는 홀수 번째 화소로 나누었다. 동시 표현 가능 색상은 256색이고 3원색인 red, blue, green은 각각 6비트의 단계로 총 218가지의 색상을 표현할 수 있다.

DRAM 데이터 입출력 회로는 영상 데이터의 저장된 값, 조작된 값, 바로 입력된 값 등 여러 데이터들 중에서 어느 하나를 선택하거나 또는 강제로 마스킹(masking)하여 입출력시키는 역할을 한다. 또한 영상 데이터를 DRAM에서 바로 읽어 RAMDAC에 보내는 역할도 한다.

3. 제어부

제어회로는 명령어 기능별로 독립된 모듈을 갖고 있고 각 모듈의 공통 신호는 논리 OR되어 출력된다. 그림 19는 매크로 명령을 수행하는 순차회로(state machine)들과 각 순차회로의 출력이 공통신호로 더해져서 프로세싱 유닛(processing unit)의 제어신호로 입력되는 구조를 보여주고 있다. Font-down 기능을 한 예로 하여 제어회로로 입력되는 명령어와 파라미터의 구성을 살펴보자. 그림 20은 Font-down 기능의 명령어 구성을 16비트를 한 단위로 순차적으로 나타낸 것이다. 먼저 주 명령어와 함께 확대여부가 입력되고 그 다음으로 2비트를 8비트로 디코딩하기 위한 font 및 shadow color가 입력된다. 그 뒤에 블럭위치와 블럭크기가 가로와 세로 각 8비트로 0~255사이의 값으로 입력된다. 실제 화면의 화소 번호는 0~511사이의 값이므로 한 주소 값은 두 화소를 동시에 지칭하게 된다. 명령어와 3개의 16비트 파라미터 값 뒤에 영상 자막 데이터가 영상의 화소 수만큼 이어진다.

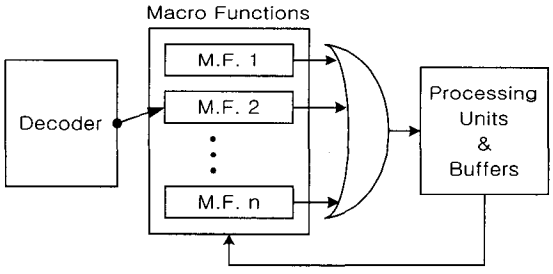

그림 19. 매크로 명령의 수행 구조
Fig. 19. Architecture of macro functions.

설계된 제어회로에서 Font-down 기능의 상태 천이를 살펴보면 그림 21과 같다. 이는 그림 20의 Font-down 명령어 구성과 일치한다. 첫 상태에서는 Font-down 명령을 받고 다른 명령어의 실행을 막기 위해 하나의 명령이 실행 중임을 나타내는 플래그(flag)를 켠다. 그 다음 상태에서부터 실제 동작으로 font color, position, size인 세 개의 16비트 정수 값을 받고 자막이나 영상을 DRAM에 쓰게 된다. 이 때에 영상의 확대 여부를 판단하여 보통, 세로 두 배, 두 배 확대인 세 갈래로 나뉜다. 쓰기 전에는 항상 읽기 과정이 있는데 먼저 쓸 영역에 다른 자막 데이터가 있는지를 확인하고 있다면 쓰기를 중단하고 그 다음 영역으로 넘어간다. 이는 자막이 자소 단위로 쓰여지므로 서로 겹쳐지는 부분이 있기 때문에 미리 쓰여진 데이터의 유

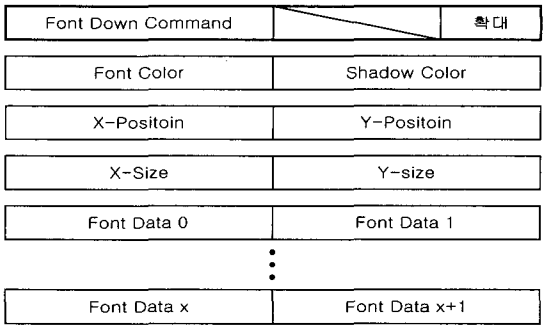

그림 20. Font-down 기능의 명령어 구성 예
Fig. 20. Example of instruction combination for Font down function.

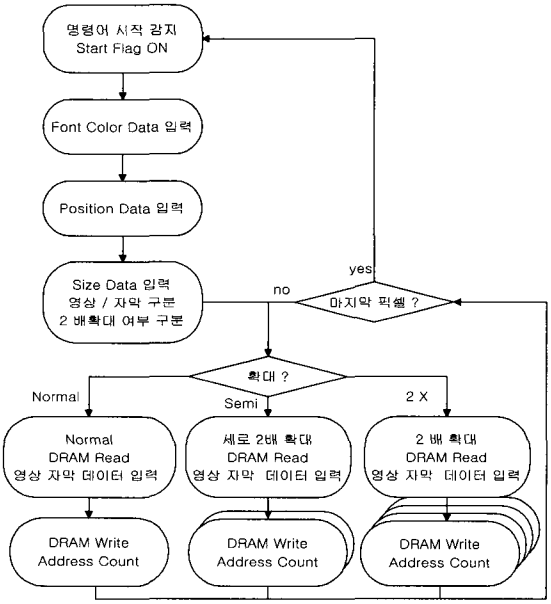

그림 21. Font-down 기능의 순서도
Fig. 21. State diagram of Font down function.

실을 방지하기 위함이다. 또한 이 읽기 과정에서 DRAM에 쓸 데이터를 받아서 대기하고 있게 된다. 보통은 쓰기 과정에서 이 대기 데이터가 한 번 쓰이지만 세로 확대의 경우는 두 번이고 두 배 확대의 경우는 네 번이 쓰인다. 읽기와 쓰기 과정은 마지막 데이터에 이르기까지 계속 반복되고 마지막 쓰기를 끝으로 다시 초기화된다.

VI. 설계 검증 및 테스트

1. 시뮬레이션을 통한 검증

시뮬레이션을 위하여 대부분의 데이터와 명령어는 FIFO로 입력시키고 기타 동기신호와 제어 신호는 내부 동작에 맞추어 주었다. DRAM의 경우는 양방향으로 데이터가 입출력되는데 DRAM의 입력은 설계된 ASIC에서 출력되어 문제가 없지만 DRAM을 읽을 때 그 데이터 값은 타이밍에 맞게 만들어 주어야 한다. 또한 내부의 동작은 외부에서 입력되는 배경화면의 비디오 신호에 항상 동기되어야 하므로 명령어 및 신호의 입력 시점과 수행 시점이 중요시되었다. 시뮬레이션은 Compass tool에서 best, typical, worst 세 단계로 delay factor를 주어 수행하였다.

설정된 기능의 검증은 25㎒로 동작하여 80000주기가 소요되었다. 회로 동작상의 한 주기인 33㎱에 10분의 1 정도의 시간에 원하는 동작확인을 위해 몇 개의 테스트 핀(test pin)을 할당하여 강제로 카운터를 동작시키거나 내부 신호 값을 변화시켜 가속(acceleration) 시뮬레이션을 수행하였다.

2. FPGA를 이용한 실시간 검증

소프트웨어적인 시뮬레이션으로는 시뮬레이션 시간이 너무 많이 소요되기 때문에 실제 상황의 극히 일부분만을 확인할 수밖에 없다. 특히 본 논문에서 다루는 영상처리 같은 경우는 한 화면의 변화, 즉 한 프레임의 영상은 33㎳ 단위이므로 한 프레임의 상황을 확인하기도 힘이 든다. 실제 시각적인 변화를 확인하려면 수초의 시간이 걸리므로 FPGA를 통한 검증이 필요하다. 이러한 과정을 통하여 설계된 ASIC을 실장해서 주변회로와의 동작을 검증할 수 있고 설계자체의 신뢰성도 높일 수 있다. 본 ASIC의 설계검증에 사용된 PLD 칩은 Altera사의 FLEX10K503으로 10만 게이트급의 용량을 가지고 있다. 실제 제작될 ASIC회로와는 비교해 지연시간이 길어 고속의 동작은 무리가 있지만 회로에 약간의 수정을 통하여 실제 동작을 확인할 수 있었다. FPGA를 이용하여 설계를 검증하는 관경을 그림 22에 나타냈다.

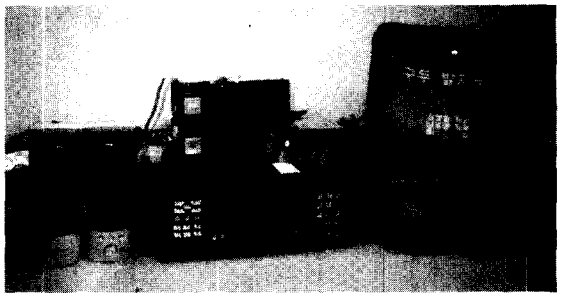

그림 22. 구현된 FPGA 보드를 통한 회로 검증
Fig. 22. Verification of design with implement- ed FPGA board.

3. 칩 제작 및 응용

FPGA를 이용하여 설계된 ASIC이 실제 시스템에서 정상적으로 동작하는 것을 검증한 후 0.8㎛ CMOS SOG로 칩을 제작하였다. 칩은 56,000 게이트로 구성되었으며, 최대 동작 주파수는 25㎒로 실제 동작 주파수 21.48㎒를 능가하였다. 소모전력은 200㎽이며 사용된 패키지는 160MQFP이다. 제작된 칩이 사용되는 보드의 구성은 그림 23과 같다. 전체 시스템의 제어 및 데이터 이동은 8051 프로세서가 맡고 있고 프로그램 수행을 위한 시스템 메모리와 폰트정보를 저장하고 있는 마스크 롬이 8051 포트에 연결되어 있다. ASIC은 8051로부터 명령 및 데이터를 받아 처리해서 영상정보를 RAMDAC으로 출력한다. 동시에 MC1378에서는 외부 입력된 바탕영상에 RAMDAC에서 출력되는 내부 합성 영상을 중첩하여 출력한다. 실제로 칩이 응용된 보드는

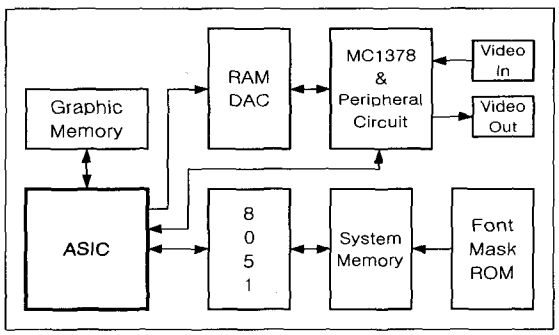

그림 23. ASIC 응용 보드의 구조도
Fig. 23. Structure of ASIC application board.

그림 24와 같다. 칩이 장착된 보드를 가요반주기 시스템에 연결하여 당초에 요구하는 기능이 정상적으로 수행되는 것을 확인하였다. 실제 시스템에 응용하여 얻은 중요한 몇 가지 기능에 대한 모니터 화면을 그림 25에 나타내었다.

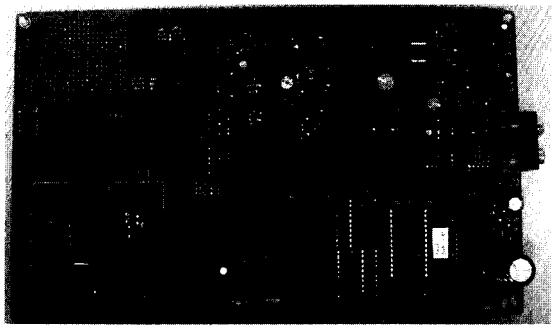

그림 24. 제작된 칩이 장착된 응용 보드
Fig. 24. Application board of fabricated chip.

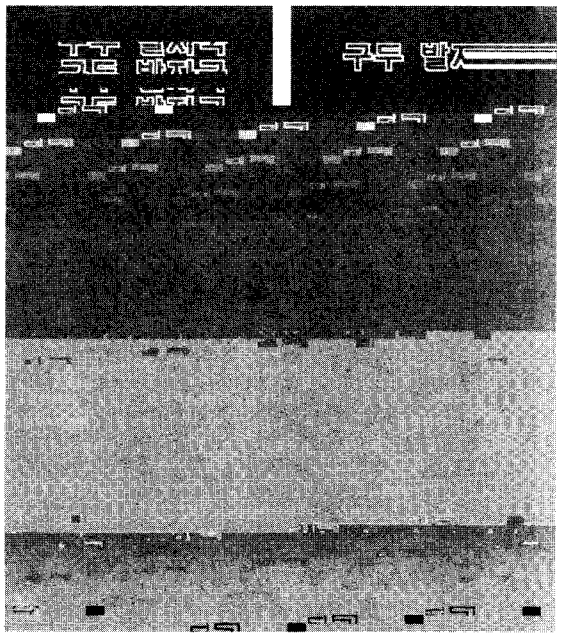

그림 25. 다양한 기능의 동작 화면
Fig. 25. Screen pictures of various operations.

Ⅶ. 결 론

본 연구에서는 OSD를 위한 영상 자막처리 ASIC을 설계·검증하였다. 자막처리 ASIC의 주요기능은 외부로부터 명령코드와 함께 영상 및 자막 데이터를 받아 여러 영상효과를 가하여 화면으로 출력하는 것으로 영상효과에는 확대, 축소, 스크롤, 펼침, 꼬리끌기 등의 다양한 기능이 있다. 설계과정에서는 먼저 회로의 사양을 정하고 기능별로 알고리즘을 고안한 뒤 각 기능을 중심으로 제어회로를 설계하고 필요한 기능블럭을 설계하였다. 이후 각각의 회로는 구조적으로 연결되었고 그 과정에서 동시에 시뮬레이션이 이루어졌다. 완성된 전체회로는 통합 시뮬레이션 검증을 거쳐 실시간 검증을 위해 FPGA로 구현하여 칩 설계의 신뢰도를 높였다. 칩 제작은 0.8㎛ CMOS 공정으로 제작되었고 최종회로의 논리 게이트 수는 56,000개 정도이며, 칩 면적은 31.2×31.2㎟이고, 동작속도는 25㎒로 제작되었다. 기존의 소프트웨어에 의한 영상 및 자막 효과를 하드웨어적으로 설계하여 동작속도를 높일 수 있었고 범용 DSP와 FIFO 및 주변 논리회로를 설계회로 내부에 포함시켜 시스템의 부품 수와 PCB 면적을 줄일 수 있었다. 또한 하드웨어적인 영상처리 구조와 주변 회로와 인터페이스 방법은 이후의 차기 설계에 한 모델이 될 수 있다.

향후 연구에서는 하드웨어적으로 고정된 효과를 소프트웨어적으로 작성과 수정이 가능하도록 함과 동시에 2차원적인 영상효과를 3차원으로 확대해 보다 다양하고 미려한 영상효과를 구현함을 목표로 하고 있다.

참 고 문 헌

[1] 공태호, "다기능 영상처리 시스템의 하드웨어 구현", 전자공학회논문지, 제24권, 2호, 315-323쪽, 1987년 2월

[2] John P. Hayes, Computer architecture and organization, Mcgraw-Hill, pp. 251-270, 1988.

[3] W. Stallings, Computer organization and architecture, New York, pp. 60-62, 1990.

[4] V. Carl Hamacher, Computer organization, Mcgraw-Hill, pp. 121-126, 1996.

[5] Anam Semicon. Co., VISA 96 User's manual, Anam Semicon. Co., 1996.

[6] 백철, "다분할 디스플레이 및 다상 오버레이 기능 설계에 관한 연구", 대한전자공학회 추계확술대회 논문집, 제20권, 2호, 827-830쪽, 1997년 11월

[7] G. Shires, "A New VLSI Graphics Coprocessor-The Intel 82786", IEEE CG&A, pp. 49-55, October 1986.
[8] 권오준, "G4 팩스용 실시간 영상 확대 축소 모듈의 개발", 포항공과대학교 전자공학과 석사학위 논문, 20-22쪽, 1995년 2월
[9] B. Grob, Basic television and Video systems, Mcgraw-Hill, pp. 128-130, 1984.

저 자 소 개

鄭 根 泳(正會員)
1974년 12월 11일생. 1997년 2월 : 부산대학교 전자공학과(공학사). 1999년 2월 : 부산대학교 전자공학과(공학석사). 1999년 3월~현재 : 부산대학교 전자공학과 대학원 박사과정. 주연구분야 : 마이크로 프로세서 설계, DSP 설계, ASIC 구현

禹 鍾 植(正會員) 第 35卷 C編 第 11號 參照

朴 鍾 仁(正會員)
1958년 5월 10일생. 1982년 8월 : 경북대학교 전자공학과(공학사). 1999년 8월 : 부산대학교 전자공학과(공학석사). 1999년 9월~현재 : 부산대학교 전자공학과 대학원 박사과정. 1982년 10월~1994년 3월 : (주)금성사(현 LG전자) 선임연구원. 1994년 3월~현재 : (주) 금영 이사. 주연구분야 : 시스템 개발 및 설계, ASIC 설계

朴 鍾 錫(正會員)
1973년 5월 26일생. 1998년 2월 : 부산대학교 전자공학과(공학사). 2000년 3월~현재 : 부산대학교 전자공학과대학원 석사과정. 1998년 3월~현재 : (주)보이소 반도체 연구원. 주연구분야 : 영상처리 구현, 마이크로 프로세서 설계

朴 柱 成(正會員) 第 36卷 S編 第 5號 參照
부산대학교 전자공학과 부교수, 컴퓨터 및 정보통신연구소 겸임연구원