'93 한국자동제어학술회의논문집 (1993.10.20~22)

마스크/논리 연산에 효율적인 H/W 구조를 갖는 영상 데이터 처리장치

° 이상현, 김진헌, 박귀태
고려대학교 전기공학과

An Image Data Processing Unit of Efficient H/W Structure for Mask/Logic Operations

° Sang Hyun Lee, Jin Heon Kim, Gwi Tae Park
Dept. of Electrical Engineering, Korea University

Abstract

This paper introduces a PC-based image data processing unit that is composed of preprocessor board and main processor board; The preprocessor contains Inmos A110 processor and efficient H/W architecture for fast mask/logic operations at the speed of video signal rate. It is controlled by the main processor which communicates with the host PC. The main processor board contains TI TMS320C31 digital signal processor, and can access the frame memory of the processor for extra S/W tasks.

We test 3×3, 5×5 masks and logic operations on 386/486/DSP and compare the result with that of the proposed unit. The result shows ours are extremely faster than conventional CPU based approach, that is, over several hundred times faster than even DSP.

I. 서론

영상 입력기로부터 입력된 영상을 문자 인식이나 패턴 인식등의 영상처리에 이용하고자 할 때 대부분의 경우 전처리의 단계가 가장 먼저 필요하다. 전처리란, 원영상을 비젼 시스템이 처리하기에 유리한 형태의 새로운 영상으로 변환시키기 위해 수행하는 일련의 처리를 말한다.[1]

영상 입력기로부터 취득된 영상 데이타는 보통 256단계의 계조치로 이루어지는 2차원의 행렬로 표현될 수 있는데, 전처리의 단계를 거치지 않고 이 256계조치의 데이타를 그대로 사용하는 경우도 있지만 이런 경우 저장 용량도 커야 하며 처리 속도도 느려져 하드웨어(hardware) 비용의 증가를 초래한다. 이를 방지하기 위해 대부분의 영상처리에서 전처리의 과정을 가지며, 또한 이 256 계조치로 이루어진 행렬을, 그 영상에 존재하는 물체나 무늬의 외곽선을 나타내는 2단계의 수치(2치)로 이루어진 행렬로 변환해 주는 등의 윤곽선 검출이 전처리 과정에서 큰 비중을 차지한다.[1][2]

영상의 이치화는 전역 트레숄딩(global thresholding)에 의한 방법이 일반적이지만 보다 정교한 처리를 위해서는 처리하고자 하는 화소(pixel) 주변의 계조치 값을 참조하여 이진화를 수행하는 마스크 및 논리 연산이 필요하다. 마스크 연산은 그림 1과 같은 값을 갖는 배열을, 원영상의 모든 화소와 식 (1)과 같은 방법으로 연산해 새로운 영상을 만들어 내는 것이다. 식 (1)에서 f(x,y)는 원영상의 (x,y)좌표상의 화소의 계조치이며, F는 마스크 연산에 의한 결과 영상이다.[2]

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

그림 1. 전처리를 위한 마스크

$$\begin{aligned} F(x,y) = {} & x_1 \cdot f(x-1,y-1) + x_2 \cdot f(x,y-1) + x_3 \cdot f(x+1,y-1) \\ & + x_4 \cdot f(x-1,y) + x_5 \cdot f(x,y) + x_6 \cdot f(x+1,y) \\ & + x_7 \cdot f(x-1,y+1) + x_8 \cdot f(x,y+1) + x_9 \cdot f(x+1,y+1) \end{aligned} \quad (1)$$

이러한 마스크 연산은 마스크의 크기를 크게 할수록 더욱 정교한 처리가 가능하다. 그러나 마스크 연산을 전화면에 대하여 수행할 경우, 마스크의 크기가 증가함에 따라 처리해야할 데이타의 양은 급격히 증가하게 된다.

-1	-1	-1
2	2	2
-1	-1	-1

(a) 가로선 추출 마스크

-1	2	-1
-1	2	-1
-1	2	-1

(b) 세로선 추출 마스크

그림 2. 선 추출용 마스크

예를들어 그림 2와 같이 가로선 또는 세로선을 추출하는 3×3크기의 마스크를 512×512(가로×세로) 화소의 한 영상에 대해 마이크로 프로세서(micro processor)로 수행할 경우 프로세서는 프레임 메모리(frame memory)를 최소 $512\times512\times9 \fallingdotseq 2.36\times10^6$번 억세스 해야 한다. 특히 5×5크기의 마스크 연산을 수행할 경우 총 억세스 횟수는 $512\times512\times25 \fallingdotseq 6.55\times10^6$번이 되어 3×3마스크 연산횟수의 2배 이상이 된다. 여기에 산술연산 시간과 프로세서의 코드 펫치(code fetch)시간을 합하면 처리시간은 더욱 증가한다.

본 논문에서는 이러한 전처리 과정 중 마스크 연산과 논리 연산을 프로세서의 관여 없이 일련의 하드웨어동작으로 구현해 마이크로 프로세서 특유의 지연요소인 인스트럭션 펫치(instruction fetch) 시간과 오퍼랜드(operand) 펫치시간, 프레임 메모리 억세스 시간등이 모두 제거된 효과적인 영상 데이타 처리장치를 제안한다. 본 논문에서 기술하게 될 처리장치는 크게 전처리기(preprocessor)와 주처리기(main processor)의 두 부분으로 나뉘는데, 전처리기는 Inmos사의 IMSA110[3] 이미지 프로세서와 논리연산 회로를 이용하여 실시간으로 마스크 연산과 논리 연산을 하드웨어적인 동작으로 수행하며, 주처리기는 영상 처리에 필요한 소프트웨어적인 연산과 전처리기의 하드웨어 동작을 제어하는 역할을 담당한다. 주처리기의 프로세서로는 TI사의 TMS320C31[4] DSP(digital signal processor)를 사용하였다. 본 논문에서는 이러한 영상 데이타 처리장치의 구조에 대해 서술하고 이 처리장치로 영상을 전처리 할 때의 수행시간과 CPU를 사용한 S/W적인 방식을 사용했을 때의 수행 속도에 대한 비교 실험을 통해 본 논문에서 제안한 처리기의 성능을 분석하였다.

II. 영상데이타 처리장치의 구조 및 동작

본 논문에서 제안된 영상 데이타 처리장치는 그림 3과 같이 PC의 ISA(industrial standard architecture) 버스(bus)상에 설치된 두장의 보드(board)로 구성된다. 주처리기는 ISA버스를 통해 PC로부터 제어를 받거나 필요한 데이타를 전달받으며 전처리기를 제어한다. 전처리기는 본 논문에서 주로 다룰 마스크 및 논리연산을 수행하는 장치로 주처리기와 전처리기 사이에는 ISA버스와는 독립적인 자체의 비젼 버스(vision bus)로 연결되어 다량의 데이타를 빠른 속도로 전송할 수 있다. 전처리기의 동작제어도 이 비젼 버스를 통해 주처리기가 담당한다. 전처리기는 자체의 프레임 메모리상의 데이타에 대해 연산을 실행할 수도 있지만, 외부 데이타 취득장치를 통해 들어오는 영상 데이타를 실시간으로 처리할 수도 있다.

1. 전처리기

전처리기는 그림 4와 같이 두개의 프레임 메모리와

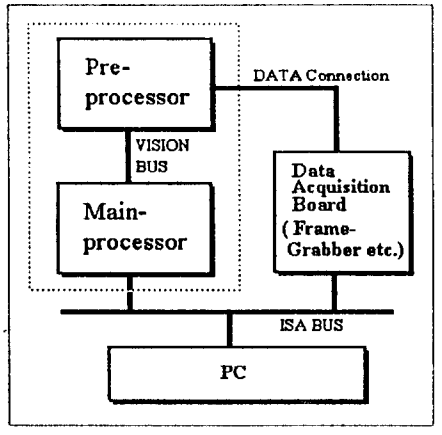

그림 3. 영상데이타 처리장치의 구조

마스크/논리 연산자및 주변 회로로 구성되어 있다. 처리될 영상과 처리된 영상은 각각 전처리기 내부의 0번 프레임 메모리와 1번 프레임 메모리에 존재하게 되며, 이들 메모리의 내용은 주처리기가 비젼버스를 통해 억세스 할 수 있다. 영상 데이타는 제어신호 발생부의 명령을 받아 0번 프레임에서 마스크 연산부로 전달되며, 마스크 연산부를 거친 데이타는 다시 논리 연산부와 1번 프레임으로 전달된다. 마스크 연산부로부터의 데이타를 1번 프레임으로 전달할지의 여부는 마스크 연산된 각 화소 데이타의 논리연산 결과에 따라 결정된다.

이미 취득된 영상에 대해 처리 작업을 수행할 때는 그림 4에서처럼 처리될 원 영상이 0번 프레임에 존재하고 이 영상이 처리되어 1번 프레임에 담기게 되지만 영상 취득기를 이용해 외부로 부터 영상 데이타를 입력할 수도 있다. 이 경우 0번 프레임으로는 영상취득기로부터 입력된 원 영상이 입력되고, 1번 프레임으로는 마스크 연산 한번의 전처리 단계를 거친 영상이 입력되게 된다. 이렇게 하면 영상 취득중에도 전처리가 동시에 일어나기 때문에 한번의 마스크 연산시간을 절약할 수 있다. 이러한 처리작업은 1/30초 내에 이루어 지므로 RS-170 신호를 실시간으로 처리할 수 있다.

그림 5는 제작된 전처리기 보드의 실물사진이다.

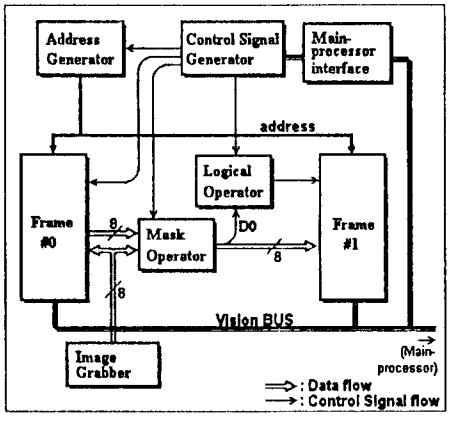

그림 4. 전처리기의 구조

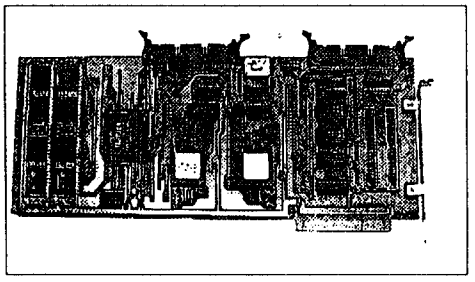

그림 5. 전처리기 보드의 실물영상

전술한 전처리기의 각 부별 동작을 기술하면 다음과 같다.

어드레스 발생부(address generator)는 프레임 메모리들에 순차적인 어드레스를 제공하는 기능을 담당한다. 즉, 어드레스 발생부로부터 어드레스를 공급받은 0번 프레임과 1번 프레임은 각각 서로 같은 어드레스에 의해 동작하지만 0번 프레임은 메모리 읽기 동작에 의해 영상 데이타를 순차적으로 마스크 연산부로 넘겨주게 되며, 1번 프레임은 메모리 쓰기 동작에 의해 논리 연산부로부터 넘겨지는 데이타를 순차적으로 저장한다. 메모리 쓰기 동작과 읽기 동작을 수행하는 신호는 제어신호 발생부로부터 전달되며, 발생시킬 어드레스의 범위는 주처리기 인터페이스(interface)부를 통해 주처리기에 의해 제어된다.

제어신호 발생부는 전처리기 내부의 모든 부분들의 동기클럭신호 및 제어신호를 제공한다. 여기서 발생된 클럭은 어드레스 발생부 내의 카운터 회로의 입력펄스와 마스크 연산부의 A110 프로세서의 동기클럭등으로 사용된다. 또한 클럭 펄스폭을 변화시켜 프레임 메모리의 $\overline{WE}$(Write Enable; 메모리 쓰기) 신호와 $\overline{OE}$(Output Enable; 메모리 읽기)신호등 제어 신호를 프레임 메모리에 제공한다. 제어신호 발생부에서 만들어지는 클럭은 프레임 메모리의 속도 문제와 논리 회로 소자들의 지연시간(delay)때문에 8MHz의 구형파로 한다. 따라서 프레임 메모리의 한 화소가 처리되는데 걸리는 시간은 동기신호의 주기인 125nS가 된다. (만약 메모리 소자와 논리 소자들의 속도가 충분히 빠르다면 A110프로세서의 최대 속도인 20MHz의 속도로도 구동할 수 있다. 이렇게 하면 한 화소를 50nS에 처리할 수 있다.)

마스크 연산부는 프레임 메모리로부터 데이타들을 받아 실시간으로 마스크 연산을 해 그 결과를 논리 연산부로 넘겨준다. 마스크 연산은 A110 마스크 전용 프로세서를 사용한다. 이 프로세서는 내부에 길이가 1120개인 3행의 쉬프트 레지스터(shift register)群과 21개의 가산기, 21개의 곱셈기등을 가지고 있어, 순차적으로 흘러 나오는 영상 데이타에 최대 3×7크기의 마스크 연산을 실시간으로 처리할 수 있다. 본 영상 데이타 처리장치에서의 마스크 연산부는 이 프로세서를 2중으로 배열해 최대 6×7크기의 마스크 연산을 수행 할 수 있도록 했다. 또한 이 프로세서는 출력단에 LUT(Look Up Table)를 가지고 있어 문턱치에 의한 이치화(thresholding)가 가능하다.

논리 연산부는 마스크 연산부에서 연산한 결과가 0또는 1의 논리값으로 2치화 된 경우 마스크연산 결과값 8비트(bit)중 LSB인 D0를 사용하여 현재 1번 프레임에 존재하는 영상 데이타와 논리연산을 한다. 논리 연산은 연산 식에서 한 오퍼랜드(operand) 값만 알아도 AND/OR연산이 가능한 경우가 존재함을 이용한 것이다. 그림 6에서 보듯이 AND연산을 할 경우, 마스크 연산부에서 출력된 값이 0이면, 목적지(1번 프레임)에 존재하는 값에 관계없이 연산 결과는 0이 되어 목적지에 0이라는 값이 등록 되어져야 한다. 반대로 마스크 연산부의 출력값이 1인 경우 연산결과는 전적으로 목적지의 현재값에 의존한다. 따라서 목적지에 연산 결과를 등록할 필요가 없어진다. OR연산의 경우 AND연산의 경우에서 0과 1을 바꿔서 생각하면 된다. 그림 7의 논리도에서 보는것과 같이 목적지에의 등록 여부는 마스크 연산 출력값(D0)과 *OR/AND*신호를 조합해 목적지(1번 프레임)메모리로 들어가는 $\overline{WE}$신호를 제어함으로써 가능하다. LE(*Logical Enable*)신호는 논리연산의 수행여부를 결정한다. 데이타가 유효해지기 시작하면 이와 동시에 $\overline{WE}$신호를 제어해야 하므로 이 부분은 매우 빠른 속도의 논리소자를 요구한다.

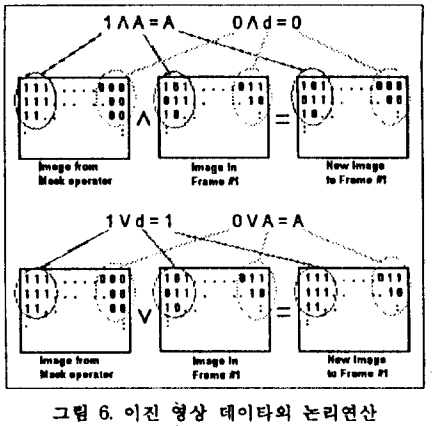

그림 6. 이진 영상 데이타의 논리연산

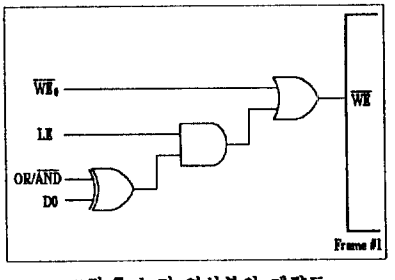

그림 7. 논리 연산부의 개략도

주처리기 인터페이스(interface)부는 전처리기의 모든 하드웨어적인 동작을 주처리기가 제어할 수 있도록 주처리기의 명령을 전처리기에 전달한다. 전처리기가 하드웨어적인 연산을 수행하지 않는 동안에는 전처리기 내

부의 프레임 메모리들은 주처리기의 메모리영역중 일부로 동작할 수 있게 하므로써 마스크/논리 연산 이외의 영상처리 작업을 주처리기가 담당하도록 하였다.

2. 주처리기

주처리기는 전처리기로 처리하기 힘든 복잡한 연산이나 인식등의 고급 알고리즘을 수행하고 전처리기의 동작을 제어하는 기능을 한다. 그림 8은 제작된 주처리기 보드의 사진이다.

주처리기에는 부동소수(floating point)연산에 강력한 능력을 가진 TMS320C31-33 DSP와 0 wait state로 동작하는 SRAM으로 구성된 프로그램용 메모리, 그리고 다량의 1~2 wait state로 동작하는 대용량의 DRAM으로 구성된 데이타 메모리가 접속되어 있다. 그림 9에서 볼 수 있듯이 주처리기의 동작은 ISA 버스를 통해 PC로부터 제어되며, 비젼버스를 통해 전처리기 내부의 프레임 메모리를 억세스 하거나 전처리기 각 부의 동작을 제어할 수 있다.

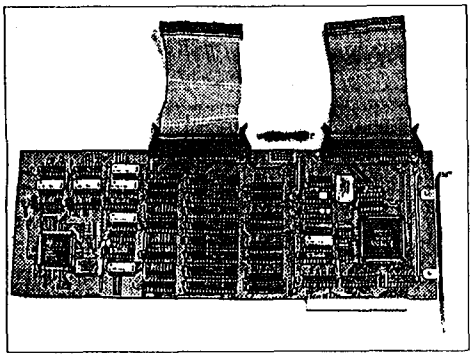

그림 8. 주처리기 보드

주처리기를 동작 시키는 애플리케이션(application) 프로그램등은 PC에서 작성되어 그 실행 코드(execution code)가 ISA 버스를 통해 DSP로 전달된다. 애플리케이션 프로그램이 주처리기에서 동작되는 동안에는 PC와 DSP간의 정보교환에 대비하여 서버(server)프로그램이 항상 PC에서 가동되고 있다. 이 서버 프로그램은 주처리기가 콘솔(console) 입출력, 화일 입출력, DOS기능 호출, ISA 버스상으로의 I/O동작등을 요구할 때 이 기능들을 수행해 준다. 이밖에 전처리기의 제어, 메모리 관리등을 위해 많은 함수들을 C언어로 만들어진 라이브러리(library)로 준비하였다. 이 함수들중 기능별로 몇가지 사례를 보이면 표 1과 같다. 이 라이브러리들을 이용하면 PC상에서 본 처리장치를 쉽게 제어할 수 있다.

III. 성능실험

본 절에서는 제안된 영상데이타 처리장치의 성능을 분석하기 위해 영상처리의 기본적인 몇가지 연산을 IBM호환 386/486 PC와 TMS320C31 DSP에서 수행했을 때와 비교해 본다.

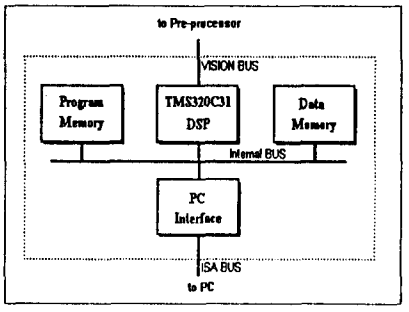

그림 9. 주처리기의 구조

표 1. 주처리기용 소프트웨어 지원 라이브러리의 예

기능	함수 예
전처리기 내외 각 부분의 동작제어	ResetA110() : A110 리셋 SetupA110() : A110의 모든 레지스터를 셋업 LoadMask() : 마스크 연산의 파라미터 셋업 LoadLUT() : LUT를 설정 SetSign() : Signed/Unsigned data결정 SetImSize() : 마스크 연산할 영상의 크기설정 SetupBoard() : 전처리기의 동작모드 설정 StatusPLSI() : 전처리기의 동작상태 SetArea() : 전처리시킬 메모리 공간 결정 CountUP() : 전처리 시작 SetLogical() : 논리 연산부의 동작설정 이외 다수
주처리기 내부의 데이타 메모리 관리	SetFrame() : 프레임 메모리의 H/W정보 설정 FormatFrame() : 각 프레임 정보 설정 AllocFrame() : 프레임 할당 FreeFrame() : 할당된 프레임 해제 ReadImage() : 화일로부터 영상 데이타 로드 SaveImage() : 영상 데이타를 디스크로 저장 ReadPixel() : 특정한 프레임의 화소를 읽음 WritePixel() : 특정한 프레임에 화소를 씀 MoveMem() : 메모리 내용을 이동(DMA이용) MemFill() : 메모리 내용을 채움
PC와의 메시지 전달 (자체 프로토콜)	SendAByte() : PC에 1byte를 보냄 GetAByte() : PC로부터 1byte를 받음 SendAWord() : PC에 1word(4byte)를 보냄 GetAWord() : PC로부터 1word를 받음 SendMessage() : PC에 연속된 자료를 보냄 GetMessage() : PC로부터 연속된 자료를 받음
File입출력, Console입출력 등 관리	printf(), scanf(), fread()등 PC에서의 환경과 동일한 함수명 다수

1. 3×3 마스크 연산

그림 10은 물체의 윤곽선을 강조하는 마스크이다.

먼저 마이크로 프로세서를 탑재한 시스템에서 512×480크기의 영상에 그림 10의 마스크 연산을 수행할 경

우 예상되는 연산 시간을 대략적으로 고찰해 본다.

마스크의 각 계수가 메모리에 저장되어 있는것을 가정할 때, 한 화소를 마스크 연산하기 위해서는 마스크의 3×3개의 각 계수값과, 처리할 화소 주위의 3×3 윈도우(window)에 해당하는 데이타를 메모리에서 읽어내어야 하며, 읽어낸 데이타를 이용해 9번의 곱셈과 8번의 덧셈을 한 후 새로운 프레임 메모리에 써야 한다. 512×480 크기의 영상을 처리하기 위해서는 이 작업을 총 512×480 = 245760번을 반복해야 한다. 이 횟수만큼을 반복 동작하는 중에는 위에서 언급한 마스크의 각 계수와 화소데이타를 읽고 쓰기위한 메모리 억세스 시간과 덧셈, 곱셈 시간만이 포함되는것은 아니다. 이 외에도 인스트럭션 펫치시간 및 오퍼랜드 펫치시간, 프로그램이 수행되기 위해 실행되는 몇가지 부가적인 코드를 실행하는 시간, 프로그램 수행도중 발생하는 인터럽트 처리시간등이 실 처리 시간에 포함될 것이다. 상황에 따라 차이는 있지만 메모리를 읽고 쓰거나 덧셈, 곱셈 연산 시간이 386의 경우 각각 수100nS또는 1000nS이상의 값을 가질 수 있으므로 386으로 동작시켰을 경우 수 초의 시간이 소요될 것으로 예상할 수 있다.[5]

본 실험에서는 "C"언어로 작성된 프로그램을 컴파일해서 수행시켜 본다. 실험환경은 표 2에서 나타낸 것과 같다. (실험 결과의 영상은 본 논문에서의 관심대상이 아니므로 생략한다.)

$\frac{1}{5}$

0	-1	0
-1	5	-1
0	-1	0

그림 10. 윤곽선을 강조하는 마스크

표 2. 마이크로 프로세서에 의한 전처리 실험환경

CPU	i386DX-40	i486DX2-66	TMS320C31-33
Benchmark Speed tested by "Norton SysInfo 7.0" (×4.77MHz XT)	41.3	132	-
Main Memory	16MB	16MB	4.5MB
Cache Memory	64KB	128KB	64Byte On-chip
Compiler	Borland C++ 3.1	Borland C++ 3.1	TMS320C3x/4x ANSI C 4.40
Compile Option	Speed Optimize	Speed Optimize	none
Instruction Set	80386	80386	TMS320C30

같은 마스크를 본 논문에서 제안한 영상데이타 처리장치로 연산하는 경우에는 마스크 프로세서인 A110은 내부에 길이가 1120개인 3행의 쉬프트 레지스터가 있어 입력되는 화소의 데이타가 3행분이 이곳에 저장되어 있기 때문에 3×3의 마스크를 처리할 때는 각 화소의 데이타를 순차적으로 한번씩만 A110으로 공급하면 된다. 따라서 한 화소를 처리하는데 걸리는 시간을 T라 할 때 512×480개의 화소를 처리하는 총 시간은 512×480×T가 된다. 그러나 A110 내부회로에 의한 지연 36주기와 전처리기 회로상의 지연 1주기를 합하면 총 연산시간 T_{total}은 식 (2)와 같이 나타낼 수 있다.

$$T_{total} = (512\times480+37)\times T = 245797T \qquad (2)$$

제어신호 발생부의 기준 클럭이 8MHz이므로 한 화소를 처리하는데 걸리는 시간 T = 1/8MHz = 125 nS이다. 따라서 총 예상시간은 $245797\times125\times10^{-9}$ ≒ 0.03072초가 된다.

표 3은 제안된 영상 처리장치와 386, 486, 그리고 본 논문에서 제작된 TMS320C31 주처리기에서 각각 같은 "C" 소스(source)프로그램으로 동작시킨 경우 실제 수행시간을 비교한 것이다. 여기서는 처리하고자 하는 영상을 프레임 메모리로 저장하거나 하드웨어 동작의 셋업(setup), 프로그램 수행 준비과정, 연산결과 영상의 출력등의 시간은 제외하고 순수한 연산 시간만을 측정하였다. 제안된 영상 처리장치의 경우 제어신호 발생기의 속도가 전처리기 수행시간에 그대로 반영되므로 예상 시간과 실제 수행시간은 같은 값을 가질 수 밖에 없다. 이 처리 시간은 측정하기에는 매우 짧은 시간이므로 같은 연산을 1000번을 수행시켜 총 시간을 수행 횟수로 나눈결과 거의 같은 값을 얻을 수 있었다. 그러나 이 경우 주처리기에서의 부가적인 지연 요소가 첨가되었으므로 정확한 연산 시간으로 볼 수 없다. 따라서 제안된 영상 처리장치에서의 수행시간은 식 (2)와 같은 계산식에서 나온 값으로 했다. 이후의 모든 실험에서도 같은 조건으로 실험을 행하였다.

표 3. 3×3 마스크 연산의 결과 (단위:초)

CPU	i386	i486	TMS31	전처리기
처리시간	10.03	3.27	1.16	0.03072

2. 5×5 마스크 연산

5×5 크기의 마스크 연산은 마스크의 크기만 다를뿐 계산 방식은 3×3크기의 마스크 연산과 같다.

마이크로 프로세서가 5×5크기의 마스크 연산을 할 경우 메모리로 부터 25개의 마스크 계수와 25개의 화소 데이타를 읽어와, 25번의 곱셈, 24번의 덧셈 연산을 수행한 후 그 결과를 메모리로 써야 한다. 이 과정을 512×480 영상의 총 화소수인 245760번 반복해야 한다. 이 횟수동안 3×3마스크 연산의 경우와 마찬가지로 여러가지 부가적인 시간지연이 포함된다. 한 화소를 마스크 연산하는데 수행되는 절차만 보더라도 3×3 마스크 연산에 비해 두배 이상의 시간인 수 초 내지 수십초의 시간이 걸릴 것으로 예상할 수 있다.

5×5마스크 연산을 본 영상데이타 처리장치로 수행할 경우 두개의 A110을 연속으로 사용해야 하므로 A110에서의 총 지연시간 72주기와 회로상 지연 1주기를 합해 73주기이다. 따라서 총 연산 시간 T_{total}은 식 (3)과 같이 되어 약 0.03073초가 된다.

$$T_{total} = (512\times480+73)T = 245833T \quad (3)$$

이 값은 3×3마스크 연산일 때와 별 차이가 없다. 그 이유는, A110 마스크 프로세서로 입력되는 영상 데이타는 칩 내부의 쉬프트 레지스터에 줄단위로 저장되므로 처리할 영상의 각 화소데이타를 한번씩만 마스크 연산부로 공급해 주면 되기 때문에, 영상 데이타를 억세스하는 횟수는 마스크의 크기에 영향을 받지 않기 때문이다.

표 4는 5×5마스크 연산을 실제 수행해본 결과를 비교한 것이다.

표 4. 5×5 마스크 연산의 결과 (단위:초)

CPU	i386	i486	TMS31	전처리기
처리시간	32.77	11.24	5.99	0.03073

3. 논리연산

두개의 2치화된 영상을 서로 논리연산할 경우 마이크로 프로세서를 사용한 소프트웨어적 기법은 두 메모리에서 각각 화소값을 읽어내어 논리 연산을 한 후 다시 메모리로 써 넣는 작업을 반복하게 된다. 논리연산을 일회 수행하기위해 읽기 두번 쓰기 한번의 메모리 억세스와 한번의 논리연산이 필요하므로 512×480화소의 영상 데이타를 모두 처리했을 경우 마스크 연산의 경우에 비해 매우 짧은 시간이 소요될 것으로 예측할 수 있다.

본 영상데이타 처리장치로 논리 연산을 시행하는 경우 마스크 연산부는 아무 동작도 않고 입력데이타를 그대로 통과 시키고 논리연산 회로는 통과되어 들어온 데이타를 목적지 프레임(frame #1)에 등록할 것인지 하지 말아야 할 것인지를 결정해 목적지 프레임을 동작시킨다. 이 모든 절차가 하드웨어적으로 동시에 일어나므로 총 512×480개의 데이타를 처리하는데 걸리는 시간 T_{total}은 식 (4)와 같이되어 약 0.03072초가 걸린다. 이는 식 (2)에서의 값과 같다. 그 이유는 마스크 연산부에서 아무 작업도 않고 입력되는 데이타를 통과시키는 데도 A110에 의한 지연 36주기를 포함해야 하기 때문이다. 표 5는 두 방법으로 논리연산한 결과 수행시간을 비교한 것이다.

$$T_{total} = (512\times480+37)\times T = 245797T \quad (4)$$

표 5. 논리 연산의 결과 (단위:초)

CPU	i386	i486	TMS31	전처리기
처리시간	2.84	1.02	0.66	0.03072

4. 마스크/논리연산

서론에서의 그림 2와 같이 어떤 화면의 가로방향의 윤곽선을 찾는 마스크와 세로방향의 윤곽선을 찾는 마스크가 있다고 할 때 완전한 윤곽선을 찾기 위해 밟아야 할 단계는 먼저 가로방향추출 마스크연산을 해 이진화(thresholding)한 후 새로운 데이타로서 저장하고, 다시 원 영상을 세로방향추출 마스크 연산을 해 이진화하고 이전에 저장했던 가로방향 윤곽선 추출 결과 영상에 논리 OR를 해 주는 절차가 필요하다.

이 과정을 CPU가 수행할 경우에는 두번의 마스크 연산과 한번의 논리연산, 그리고 두번의 영상 이진화가 수행되어야 한다. 문턱치에 의한 이진화는 고급언어로 구현할 경우 비교 판단문이나 LUT역할을 하는 배열을 사용할 수 있다. 두 방법 모두 많은 수의 프로그램 코드를 사용하기 때문에, 이를 수행하는데 걸리는 시간은 한 영상 데이타를 논리연산 하는데 걸리는 시간보다도 길 수 있다. 이전 실험에서의 마스크 연산 시간과 논리연산을 토대로 하고, 두번의 이진화 시간을 고려하면 총 연산시간은 386의 경우 수십초가 될 것으로 예측할 수 있다.

같은 연산을 본 영상 데이타 처리장치로 수행할 경우 논리 연산은 마스크 연산의 수행시 함께 수행되며, 영상 이진화 역시 A110의 출력단에서 LUT에 의해 실시간으로 마스크 연산과 함께 동작되므로 총 수행 시간은 두번의 마스크 연산을 수행할 때와 같다. 이를 식으로 표현하면 식 (5)와 같이 되어 약 0.06145초가 된다.

$$T_{total} = 2\times(512\times480+37)\times T = 491594T \quad (5)$$

표 6은 두개의 마스크연산과 논리 연산을 조합했을 때의 실제 실험시 소요 시간을 비교한 것이다.

표 6. 마스크/논리 연산의 결과 (단위:초)

CPU	i386	i486	TMS31	전처리기
처리시간	27.83	9.81	4.45	0.06145

IV. 결론

본 논문에서는 전처리 단계에서 많은 시간을 소요하는 마스크 연산과 논리적인 연산을 하드웨어적인 방법으로 구현해, 마이크로 프로세서에 의한 소프트웨어적인 방법에 비해 획기적으로 빠른 속도를 구현할 수 있는 영상 데이타 처리장치를 제안하고, 일반적으로 전처리에 많이 사용되는 연산들을 적용해 보아 i386, i486, TMS320C31 등의 마이크로 프로세서를 탑재한 시스템에서의 결과와 비교해 보았다.

본 논문에서 제안된 영상 데이타 처리장치의 특징을 나열하면 다음과 같다.

① 512×480 크기의 영상 하나를 마스크 연산하는데 걸리는 시간이 약 1/30초로 실시간 전처리가 가능하다.

② A110내부의 쉬프트 레지스터가 화소 데이타를 3행분을 가지고 있기 때문에 화소 데이타는 항상 1번씩만 A110으로 공급되므로 마스크의 크기가 커져도 수행 시간에는 큰 변동이 없다.

③ 마스크연산과 논리 연산이 연속된 회로에서 같은 동기시간에 수행되므로 마스크 연산과 논리 연산을 혼합해 수행시켜도 한번의 마스크 연산 시간과 같은 시간에 수행된다.

④ 영상 취득기를 이용해 외부로 부터 영상을 입력하는 경우 영상 데이타를 마스크 연산부를 통해서 입력할

수 있다. 따라서 한단계의 전처리를 거친 영상과 원 영상을 동시에 얻을 수 있으며 RS-170 영상 신호를 취득과 동시에 실시간으로 전처리 할 수 있다.

⑤ 자체의 비젼 버스를 사용하기 때문에 데이타 전송이 빠르며 주처리보드에서의 프레임 메모리 관리가 용이하다.

⑥ PC를 개발 환경으로 하기 때문에 애플리케이션 개발이 용이하고, 개발 비용이 적게든다.

⑦ A110내부의 LUT를 이용하면 실시간으로 전역적인 트레숄딩이 가능하다.

⑧ 부동 소숫점 연산에 강력한 기능을 갖는 DSP인 TMS320C31로 구성된 주처리기는 인식등의 고급 알고리즘을 빠른 속도로 구현할 수 있다.

참고문헌

[1] 김태균, 최형진, 화상처리 기초, 正益社, 1990

[2] Rafael C. Gonzalez, Paul Wintz, *Digital Image Processing*, Addison-Wesley, 1987

[3] *Image Processing*, (c) INMOS, 1989

[4] *TMS320C3x User's Guide*, (c) TI, 1991

[5] *80386 Hardware Reference Manual*, (c) Intel, 1987

* 본 연구는 상공자원부 공업 기반 기술 과제의 지원을 받아 시행되었습니다.