論文2002-39SD-11-4

영상 평활화를 위한 화소-병렬 영상처리 시스템에 관한 연구

(A Study on the Pixel-Parallel Image Processing System for Image Smoothing)

金 玄 起 *, 李 天 熙 **

(Hyun Gi Kim and Cheon Hee Yi)

요 약

본 논문에서는 포맷 변환기를 사용하여 여러 가지 영상처리 필터링을 구현하였다. 이러한 설계 기법은 집적회로를 이용한 대규모 화소처리배열을 근거로 하여 실현하였다. 집적구조의 두가지 형태는 연산병렬프로세서와 병렬 프로세스 DRAM(또는 SRAM) 셀로 분류할 수 있다. 1비트 논리의 설계 피치는 집적 구조에서의 고밀도 PE를 배열하기 위한 메모리 셀 피치와 동일하다. 이러한 포맷 변환기 설계는 효율적인 제어 경로 수행을 능력을 가지고 있으며 하드웨어를 복잡하게 할 필요 없이 고급 기술로 사용 될 수 있다. 배열 명령어의 순차는 프로세스가 시작되기 전에 호스트 컴퓨터에 의해 생성이 되며 명령은 유니트 제어기에 저장이 된다. 호스트 컴퓨터는 프로세싱이 시작된 후에 저장된 명령어위치에서 시작하여 화소-병렬 동작을 처리하게 된다. 실험 결과 1)단순한 평활화는 더 높은 공간의 주파수를 억제하면서 잡음을 감소시킬 뿐 아니라 에지를 흐리게 할 수 있으며, 2) 평활화와 분할 과정은 날카로운 에지를 보존하면서 잡음을 감소시키고, 3) 평활화와 분할과 같은 메디안 필터링기법은 영상 잡음을 줄이기 위해 적용될 수 있고 날카로운 에지는 유지하면서 스파이크 성분을 제거하고 화소 값에서 단조로운 변화를 유지 할 수 있었다.

Abstract

In this paper we implemented various image processing filtering using the format converter. This design method is based on realized the large processor-per-pixel array by integrated circuit technology. These two types of integrated structure are can be classify associative parallel processor and parallel process DRAM(or SRAM) cell. Layout pitch of one-bit-wide logic is identical memory cell pitch to array high density PEs in integrate structure. This format converter design has control path implementation efficiently, and can be utilize the high technology without complicated controller hardware. Sequence of array instruction are generated by host computer before process start, and instructions are saved on unit controller. Host computer is executed the pixel-parallel operation starting at saved instructions after processing start. As a result, we obtained three result that 1)simple smoothing suppresses higher spatial frequencies, reducing noise but also blurring edges, 2) a smoothing and segmentation process reduces noise while preserving sharp edges, and 3) median filtering, like smoothing and segmentation, may be applied to reduce image noise. Median filtering eliminates spikes while maintaining sharp edges and preserving monotonic variations in pixel values.

Key Words : 병렬처리, 포맷 변환기, 영상처리

* 正會員, 極東情報大學 電子通信科
(Keukdong College., Dept of Electronic Telecommunication)

** 正會員, 淸州大學校 電子工學科
(Chong-ju Univ., Dept of Electronic Engineering)

接受日字:2001年8月22日, 수정완료일:2002年10月23日

Ⅰ. 서 론

현대의 VLSI 기술은 고밀도로 배열된 PE(Processing Elements)에서 메모리와 프로세서들로 집적화되어 있으며 PE 배열들은 화소-병렬 영상 처리 시스템을 위한 기본 형태가 된다. 각각의 PE는 하나의 영상을 갖는 한 화소(pixel)를 저장하고 처리한다.[1] <그림 1>은 화소-병렬 영상 처리 시스템의 구성도를 보여주고 있는데 이것은 한 개의 PE 배열, 컨트롤러, 주 컴퓨터와 두 개의 포맷 변환기로 구성되어 있다. 카메라로부터 출력되는 아날로그 신호는 ADC(Analog-to-Digital Converter)를 거쳐 디지털 신호로 변환되며 이때 PE에서 처리하기 위한 신호로 포맷이 되어 PE 배열에 옮겨진다. 주 컴퓨터로부터 나온 명령들은 컨트롤러를 경유해서 모든 PE에 전달되며 처리된 데이터들은 다음의 처리를 위해 재 포맷 된다.

PE 배열에 필요한 두 개의 집적 회로 구조는 내용을 어드레스 가능 메모리 셀에 사용하는 연산 병렬 프로세서와 종래의 DRAM 셀들을 사용하는 미세하게-결정(結晶)된 병렬 구조가 있다. 이 구조에서 각각의 PE에 필요한 논리회로는 동일한 칩의 DRAM과 결합됨으로서 행 디코더에 필요한 것을 제거해준다. 이런 식으로 고밀도와 소형의 메모리 셀 크기를 가진 구조를 구현할 수 있다.

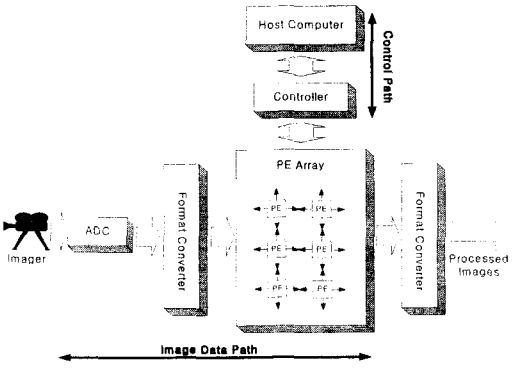


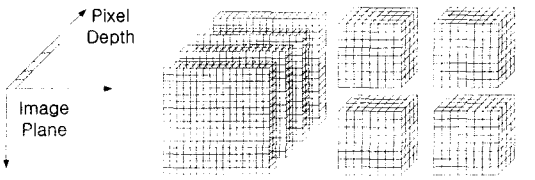
그림 1. PE 배열을 사용한 영상 처리 시스템
Fig. 1. Image processing system using PE array.

이 시스템은 주 컴퓨터로부터 나오는 명령을 구현할 수 있으나 실시간 영상 데이터가 시스템에 들어가고 나오는 경로는 결여되어 있다. 만일 ADC로부터 나오는 영상 데이터 출력이 화소를 기초로 한 PE 배열에 직접 전송되는 경우 단지 하나의 PE만이 전송이 가능할 것이다. 따라서 좀 더 효율적인 PE 배열을 이용하기 위해서는 데이터가 병렬 배열로 전송될 수 있도록 영상 데이터의 포맷변환이 필요하다. 본 논문에서는 <그림 1>에서 나타낸 것과 같이 비디오 카메라등의 아날로그 영상을 디지털 영상으로 바꾸어 PE 배열에서 처리하는 입·출력 포맷 변환장치를 설계하여 여러 가지 필터링을 통하여 영상을 처리하는 과정을 제시하였다. 시스템 환경은 SUN UNIX OS 상에서 VHDL을 사용하여 Synopsis 툴을 이용하여 합성하였고, Xilnix FPGA 칩으로 시뮬레이션 하였다.

Ⅱ. 화소-병렬 영상처리 시스템

1 데이터 입·출력 발생기

디지털화된 영상을 표현하는 한 가지 방법중의 하나는 3차원적인 블록으로 나타내는 것이다. 블록의 높이와 넓이는 영상의 크기를 나타내며 깊이(안쪽 길이)는 각각의 화소에 있는 비트의 수를 나타낸다. <그림 2(a)>는 화소당 네 개의 비트를 갖는 영상을 보여주고 있다. 일반적인 그래픽 응용에서 비디오 메모리들은 <그림 2(b)>에서 볼 수 있는 것처럼 영상 블록을 비트-평면(bit-plane)으로 나누는데 이용된다. 여기에서 하나의 비디오 메모리칩은 그 영상의 한 비트 평면을 저장한다. ADC는 연속적인 화소 형태로 연이어서 영상 데이터를 출력한다.



(a)Dimension(b) Planar partition (c) Spatial partition
그림 2. 3차원적인 영상 재현
Fig. 2. Image representation of 3 dimension.

그리고 배열은 실제로 몇 개의 칩으로 이루어져 있으며 각각의 칩은 영상의 한 부분을 처리한다. 따라서 <그림 2(c)>에서 볼 수 있는 것과 같이 영상의 공간적 분할이 있게 된다. 정보를 빠르게 전송하기 위해서는 모든 칩에 데이터를 동시에 보내 주어야 한다. 데이터

경로 구조의 목적은 PE 배열에 대한 실시간 데이터 전송을 가능하게 하는 것이기 때문에 들어오는 영상과 배열에 의해 요구되는 영상 포맷간의 적합성이 필수적이다. 따라서 포맷 변환기들은 효율적인 데이터 전송과 더불어 효율적인 배열 상태를 고려하여 설계하였다.

2 포맷 변환기 설계

MDA(Multi-Dimensional Access) 메모리 저장 형식은 화소 비트들이 메모리 버퍼에 저장되기 전에 재배열하고 처리된 데이터가 PE 배열에 전송되기 전에 재배열할 수 있는 데이터 혼합기(Data Shuffler)들의 구현을 필요로 한다. 각각의 메모리 칩에 대하여 여러 가지 다른 메모리 어드레스 레지스터(MARs)를 필요로 한다. <그림 3>은 데이터를 PE 배열에 보낼 수 있는 포맷 변환기의 모듈을 보여주고 있다. 디지털화된 영상 데이터를 입력으로 하고 배열에 대한 전송을 위해서 적절한 단위로 묶여져 포맷된 데이터를 생성한다. 이 포맷 변환기를 위해 혼합기와 MAR 두 가지를 설계하여야 한다. 또한 다른 포맷 변환기에서 처리된 데이터를 디스플레이에 보내는데 이용되는 역행 처리과정을 이해하여야 한다.

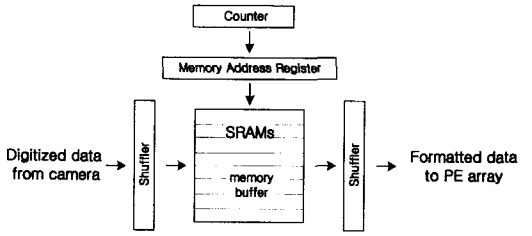


그림 3. 포맷 변환기의 기능적 모듈
Fig. 3. Functional modules of format converter.

<그림 4>의 저장 패턴은 데이터는 메모리 칩에 순서적으로 저장되지 않는다. 각각의 칩에 대해서는 각기 다른 연속성이 필요하다. 영상 데이터가 화소로 연달아서 도달되는 것처럼 단지 칩 0만이 위치 0(location 0)으로부터 위치 7(location 7)에 까지 그 메모리 위치들의 연속적인 어드레스를 유지한다. 칩 1의 경우 메모리 위치 1이 우선 어드레스되고 위치 0이 그 뒤를 따르며 그리고 연속적으로 위치 3과 위치 2가 뒤따른다. 즉 해당 어드레스는 '점프'하며 두 개의 메모리 위치마다 순서가 바뀐다. 칩 2의 경우 바뀐 두 개의 연속적인 메모리 위치로 이루어진 두 세트의 순서가 된다

이러한 '점프' 알고리즘은 계속하여 화소 0에서 화소 7에 있는 비트들 어드레스의 역행 순서로 칩 7에 저장한다. 바로 이런 어드레스 구성을 수행할 수 있는 한가지 간단한 방식은 카운터의 활용이다. 특히 카운터 출력들은 각각의 MAR(Memory Address registers)에 따라 바뀐다. 그런 구현은 <그림 5>에 나타내었다.

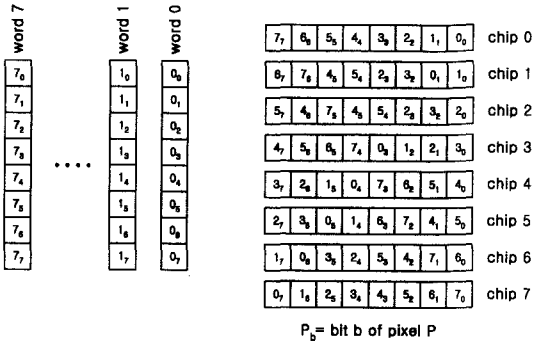


그림 4. 8×8 MDA 메모리 저장 패턴
Fig. 4. Storage pattern for an 8×8 MDA memory.

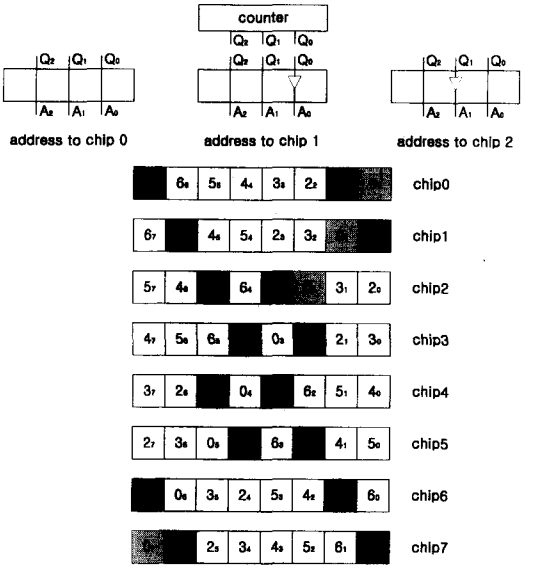


그림 5. 데이터 저장을 위한 MARs
Fig. 5. Functional description of the MAR for storing data.

메모리 버퍼로부터 나온 데이터를 처리하는 8×1 영상의 경우에 동일한 비트-평면에 속하는 모든 비트들은 동일한 메모리 위치로부터 처리된다. 특히 비트-평면 0에 있는 모든 비트들은 위치 0에 있고 비트-평면 1은 위치 1에 있으며 나머지도 동일하다. 마찬가지로

단지 하나의 어드레스만이 256×256 화소에 필요한 메모리 버퍼로부터 나온 데이터를 처리하는데 필요하다. 그러므로 각각의 영상블록을 배열에 전송할 때 처음 16개의 열에서 모든 열들은 두 번째 세트의 상응하는 열보다 앞서야 하므로 이것은 어드레스 순서에서 '점프'를 필요로 한다. 비트의 재배열은 <그림 6>과 같다.

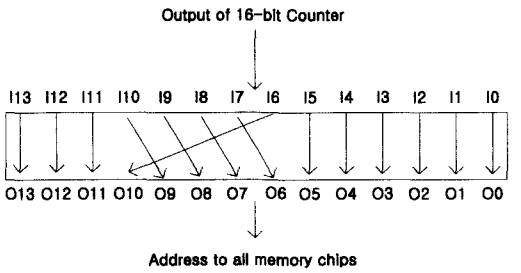


그림 6. 데이터를 액세스하는데 필요한 MAR의 기능 설명

Fig. 6. Functional description of the MAR for accessing data.

3 혼합기(Shuffler)

영상 데이터를 메모리 버퍼에 저장하기 위해서, 화소 비트와 메모리 어드레스 두 가지 모두가 혼합될(shuffled) 필요가 있다. 블록 0에 있는 각 화소의 8개 비트들은 상응하는 8개의 메모리 칩 속에 저장된다. 그러나 블록 1에 있는 화소들의 경우 모든 두개의 비트들마다 서로 스위칭 된다. 예를 들어 비트 0은 칩 1에 저장되고 비트 1은 칩 0에 저장된다. 블록 2의 경우 모든 두 개의 2-비트 그룹들마다 서로 스위칭 된다. 그런 스위칭 패턴은 계속되는데 이것은 블록 7의 경우에 그 순서가 칩 7에 저장된 비트 0은 칩 0에 저장된 비트 7이 반전되도록 하기 위함이다.

총 8개의 다른 혼합 패턴은 데이터를 메모리 버퍼에 저장하기 위해 필요하다. 그러나 기능적인 측면에서 그것들은 <그림 7>에서 볼 수 있는 것처럼 단지 4개의 다른 스위칭 블록들의 적절한 결합을 통해서 생성될 수 있다. 스위칭 블록S_0은 데이터의 순서를 변화시키지 않은 채로 둔다. S_1은 모든 2개의 비트마다 스위칭 한다. S_2는 모든 2개의 2-비트 그룹들마다 스위칭 한다. 그리고, S_4는 두 개의 4-비트 그룹들마다 스위칭 한다. 바람직한 패턴은 멀티플렉서로의 입력을 통해 선택된다. 비록 저장 패턴이 비트-평면의 처리를 쉽게 허용한다고 할지라도 비트-평면 내의 비트들이 다시 순서 있게 정렬되어야 한다는 것이다. 그러므로 메모리 출력에서는 데이터를 PE 배열에 보내기 위해 또 다른 혼합기가 필요하다. 저장 패턴에 대한 관점은 두 번째 혼합기가 설명된 것과 똑같은 방식으로 구현될 수 있다는 것을 보여주는 것이다. 기능적인 측면에서, 혼합기들 두 가지 모두는 동일하다.

4 메모리 버퍼

메모리 버퍼 32개의 개별 메모리 칩을 필요로 하는데 그 각각은 1비트 버스를 갖는다. 1 이상의 비트로 된 버스들을 갖는 SRAMs들이 이용된다면 필요한 칩들의 수와 그 시스템 비용은 크게 절감될 수 있다. 8개의 16K×4 SRAM 칩들을 사용하면 메모리들은 <그림 8>에서 볼 수 있는 것처럼 그룹화 될 수 있다. 카메라가 하나의 화소를 출력할 때 모든 8 비트들은 8개의

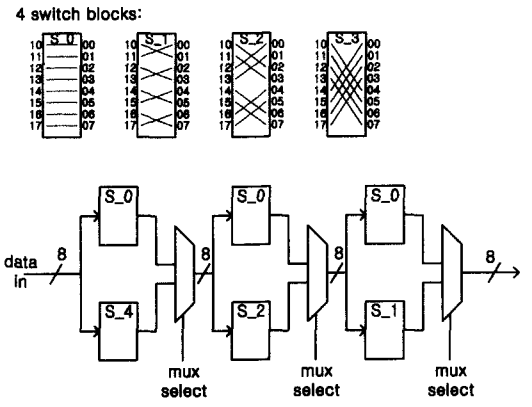


그림 7. 혼합기의 기능 설명

Fig. 7. Functional description of a shuffler.

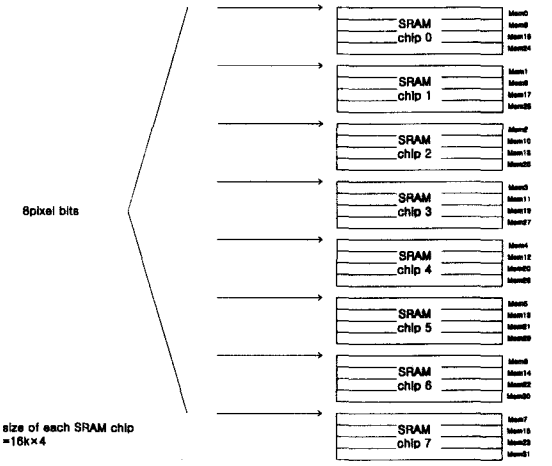


그림 8. 메모리 버퍼 구현

Fig. 8. Memory buffer implementation.

물리적으로 분리된 칩들 속에 저장될 수 있다. 그러므로 포맷 변환기는 데이터를 메모리 버퍼에 '매스크 쓰기(mask writing)'할 수 있는 능력을 부여해서 설계 되어야 하며 각각의 화소 클럭 기간 중에 포맷 변환기는 읽기 싸이클을 구현하고 한 단어에서 단지 한 비트만을 수정하여 그 결과를 동일한 메모리 위치에 기록할 필요가 있다. 이런 식으로 영상 블록 8에서 15는 메모리 8에서 15에 합쳐져서 저장될 수 있다. 그 동안 영상 블록 0에서 영상블럭 7에 있는 정보를 계속 보관하게 되는데 이것들은 메모리 0에서 메모리 7에 저장된다.

Ⅲ. 포맷 변환기 하드웨어 구현

데이터 패스 구조를 하드웨어로 구현하기 위해 <그림 9>의 하드웨어 기능 블록도에 있는 네 가지의 기능 블록들로 구성되어 있다. 이것들은 한 개의 영상 I/O 단위, 두 개의 포맷 변환기, 그리고 한 개의 PE 배열 인터페이스이다. 영상 I/O 단위는 카메라, 화면표시장치, 그리고 VMEbus가 갖춰진 인터페이스로 구성되어 있다. 각각의 포맷 변환기에서는 메모리 버퍼를 구현하기 위해 8개의 SRAM들이 사용되고 혼합기들과 MAR들을 구현하는데는 FPGA가 사용된다. 각기 다른 구동기들과 수신기들이 PE 배열 인터페이스에 사용되며 데이터는 twisted-pair ribbon cables를 경유해서 PE 배열에 전송되기도 하고 PE 배열로부터 전송되기도 한다. 포맷 변환기, 특히 FPGA들은 실시간 영상 처리의 실현을 가능하게 한다.[8]

<그림 10>은 입력 포맷 변환기용 FPGA 내에 있는 주 기능 블록들을 보여주고 있다. 디지털화된 영상 데이터는 액티브 비디오 주기 동안 왼 쪽에 있는 혼합기 1에 의해 재배열된다. 이 포맷 변환기는 '매스크 쓰기' 동작을 수행한다. 그러므로 영상 화소가 혼합기 1에 의해서 혼합되는 동안 8개의 단어들이 SRAM들로부터 회수되어서 매스크 기록 논리 블록 속에 입력된다. 이런 식으로 혼합된 영상 데이터는 그것들의 적절한 비트 위치에 있는 메모리 버퍼로 저장된다. 그런 후 수직 블랭킹 기간 중에 데이터는 메모리 버퍼로부터 읽어들여 혼합기 2에 의해서 다시 재배열되고 그 결과들은 PE 배열로 보내어 진다. FSM들은 모든 제어 신호들, 특히 비디오 디지타이저를 초기화하기 위한 신호, 데이터를 SRAM들에 저장하기 위한 신호, 그리고 데이터를 PE 배열에 전송하는데 필요한 신호들을 담당한다.

출력 포맷 변환기는 <그림 11>과 같이 입력 포맷 변환기와 유사한 방법으로 구현된다. 그 메모리 버퍼를 구현하는데는 8개의 SRAM 칩이 사용된다. 혼합기는 둘 다 기능적으로 FPGA 1에 있는 것들과 일치한다. FPGA들에서의 주요한 두 가지 차이는 MAR 블록들 및 선택 로직 블럭과 대비한 매스크 논리 블록이다. 각각의 영상 블록의 경우에 PE 배열은 데이터 패스 보드로부터 데이터의 첫째 열을 수신하지만 데이터의 마지막 열을 데이터 패스 보드로 전송한다. 선택 블록은 본질적으로 매스크 논리 블록의 역기능을 수행한다. 데이터를 메모리 버퍼 2로부터 화면으로 송신하거나 또는 호스트 컴퓨터로 전송할 때 32비트의 데이터가 SRAM들로부터 판독되지만, 단지 8 비트만이 선택되고 혼합된다.

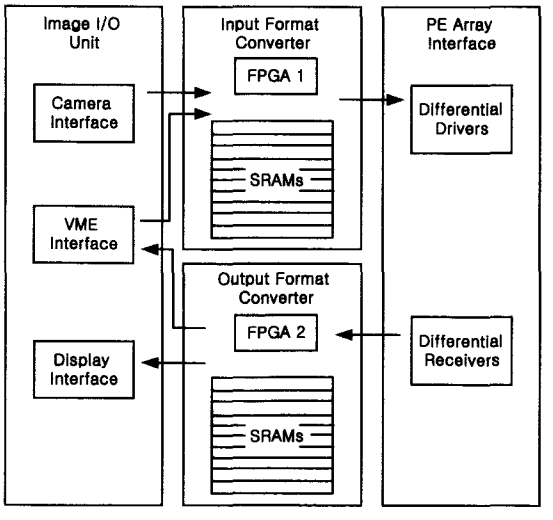


그림 9. 하드웨어 기능 블록도
Fig. 9. Hardware functional block diagram.

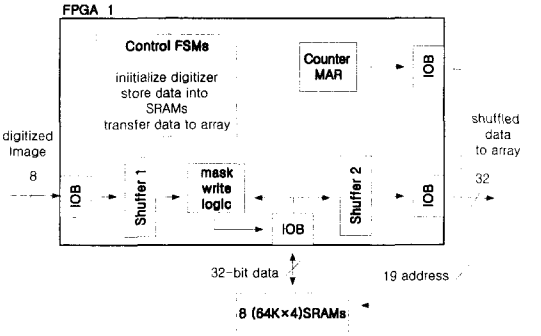


그림 10. 입력 포맷 변환기의 FPGA 기능 블록도
Fig. 10. FPGA functional block diagram for input format converter.

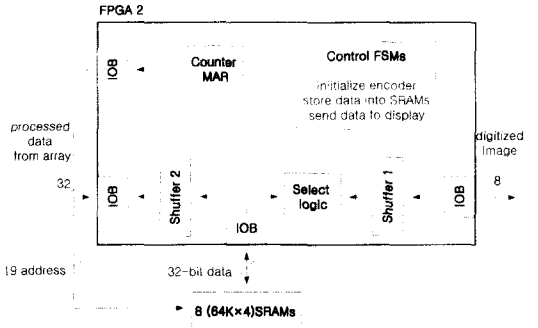


그림 11. 출력 포맷 변환기의 FPGA 기능 블록도
Fig. 11. FPGA functional block diagram for output format converter.

Ⅳ. 실험 및 고찰

1. 포맷 변환기 타이밍 도형

카메라와 입력 포맷 변환기 사이에 존재하는 타이밍 관계, 입력 포맷 변환기와 PE 배열 사이에 존재하는 타이밍 관계, PE 배열과 출력 포맷 변환기사이에 존재하는 타이밍 관계, 그리고 출력 포맷 변환기와 화면 사이에서 존재하는 타이밍 관계가 있다. <그림 12>는 A/D 변환기가 출력한 데이터를 입력 포맷 변환기에 있는 SRAM 속에 저장하는 데 필요한 타이밍도이다. 입력 포맷 변환기는 한 화소의 영상을 메모리 버퍼의 적당한 위치에 저장하기 위해 160ns를 갖는다. FPGA의 매스크 쓰기 논리는 혼합기 1이 재배열한 영상 데이터를 받아서 그것들을 SRAM으로부터 생긴 데이터와 결합시킨다.

그림 12. SRAM에 데이터 저장을 위한 타이밍도
Fig. 12. Timing for storing data into SRAM.

<그림 13>은 PE 배열로의 전송을 위해 입력 포맷 변환기의 SRAM으로부터 생긴 데이터를 처리하는데 필요한 타이밍도이다. SRAM에 연결된 어드레스는 FPGA가 생성한다. SRAM의 데이터 읽기는 FPGA에 있는 레지스터들의 셋업 타임을 충족시켜줄 필요가 있다. 그러므로 래치된 데이터는 혼합기 2가 재배열하며 나중에 FPGA 출력 핀에 2개의 클럭 주기에서 나타난다. FPGA가 생성한 BUSY 신호는 첫 번째 데이터 그룹이 PE 배열에 전송되며 마지막 데이터 그룹에 전송된다. START 신호는 출력 포맷 변환기에서 FPGA로의 입력으로 이용된다. Low로부터 High로의 변화시 PE 배열로부터 처리된 데이터의 첫 번째 그룹이 PXCK의 상승 에지의 데이터 패스 보드에 래치(latch)된다.

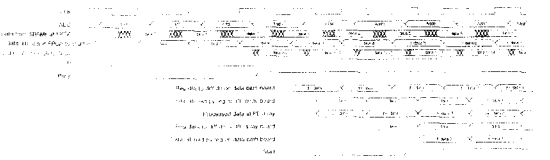
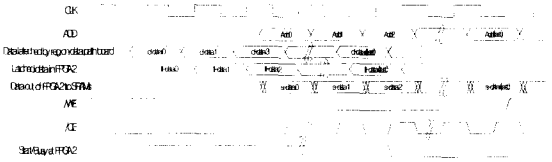
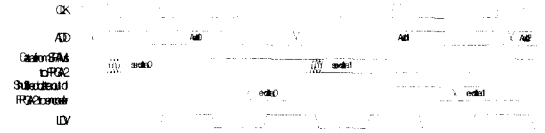


그림 13. SRAM의 데이터 처리를 위한 타이밍도
Fig. 13. Timing for data access into SRAM.

PE 배열로부터 처리된 데이터를 출력 포맷 변환기에 저장시키는데 필요한 타이밍과 데이터를 화면에 보내는데 필요한 타이밍은 간단하다. <그림 14(a)>와 같이 PE 배열로부터 생긴 데이터를 출력 포맷 변환기의 FPGA에 래치되고 혼합된 후 적정 메모리 어드레스는 데이터를 SRAM들 속에 저장하기 위해 생성된다. SRAM 칩 enable(CE)과 기록 enable(WE)은 SRAM 기록 주기의 타이밍을 충족시키기 위해 사용된다. <그림 14(b)>는 데이터를 화면에 보내는데 필요한 타이밍을 보여 준다. FPGA 2는 SRAM과 연결된 어드레스들을 출력한다. 그 다음 혼합된 데이터는 3클럭 주기 이후 부호기로 보내어 진다. LDV는 부호기가 생성한 신



(a) PE 배열로부터 버퍼 2로 전달하는 타이밍도



(b) 버퍼 2로부터 화면으로 전송되는 데이터 타이밍도

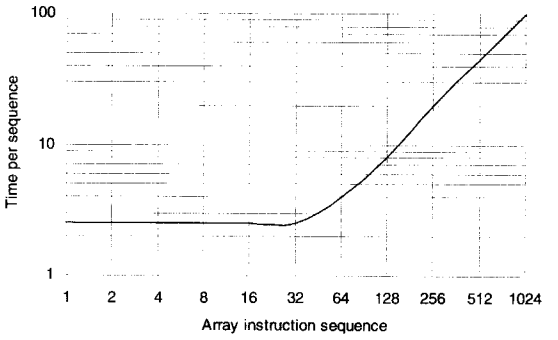
그림 14. PE 배열로부터 처리된 데이터를 화면으로 전송하는 타이밍도
Fig. 14. Timing for sending processed data from PE array to display.

호이며, 디지털 데이터를 부호기 안의 입력 레지스터로 래치하는데 이용된다.

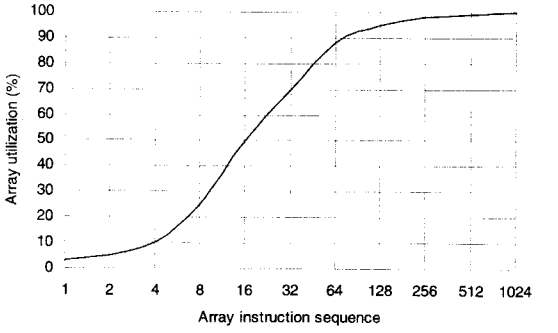
2. 제어 패스 성능 평가

본 실험에서는 Wire-Wrap VMEbus panel을 사용하는 기본 컨트롤러를 구현하였다. 이 기본 컨트롤러는 16 b 마이크로프로그램 순차, 4개의 메모리 모듈, 버스 인터페이스의 구성요소, 그리고 많은 PLD(programmable logic device)와 레지스터 칩을 이용한다. 이러한 실험을 함으로써 시스템 동작의 특징을 알고 이 설계의 고밀도 패키지에 대한 유동성에 초점을 맞추었다. 프로그램이 가능한 논리 장치와 레지스터 칩을 게이트 배열로 대체함으로써 훨씬 더 밀도 있는 구현이 이루어졌다. 컨트롤러는 PE 장치의 10MHz에서 최적으로 기능을 수행하게 된다. 제어 패스 설계의 실행을 위해 10MHz에서의 1명령에서 1024까지의 명령의 길이를 가진 순차에 대한 제어 패스의 실행을 평가하였다.

<그림 15>는 실험 결과를 보여주고 있다. <그림 15(a)>는 호스트 컴퓨터에서부터 컨트롤러까지 시작



(a)순차의 길이와 순차 호출시간



(b)순차의 길이와 배열의 관계

그림 15. 제어 패스 성능 평가
Fig. 15. performance estimate of control pass.

주소를 전송하는데 필요한 시간의 양을 보여준다. 그림에서 곡선이 굴곡 되는 부분을 보면 짧은 순차에서는 순차가 호스트 컴퓨터에 의해 호출할 때마다 약 2.5μs로 통과하지만 긴 순차에서는 각 순차를 이동시키는데 걸리는 시간이 시작 주소를 전송하는데 필요한 시간을 초과한다. 따라서 실행시간은 순차의 길이에 따라 달라짐을 알 수 있다. <그림 15(b)>는 순차의 길이와 배열의 활용과의 관계를 보여준다. 만약 순차가 짧으면 컨트롤러는 다음 시작 주소를 받기 전에 각 명령들의 순차를 배열로 전달하는 것을 끝낸다. 그 결과 배열 이용율은 낮다. 만일 순차가 길면 컨트롤러는 각 순차의 전달이 끝나기 전에 다음 시작 주소를 받는다. 그렇기 때문에 배열의 이용은 각 순차를 시작하기 위해 컨트롤러가 필요로 하는 외부의 클록 주기에 의해서만 제한을 받는다. 따라서 30명령보다 긴 순차들이 효율적으로 실행된다.

3. 화소-병렬처리 결과

시스템이 실행되는 것과 프로그래밍의 구성의 이용을 보여주기 위해 2개의 응용기술 예를 제시하였다. 이러한 기술은 일반적인 하위수준 영상 처리의 계산에 필요한 것을 보여주기 위한 것이며 특정한 영상 처리 문제에 대한 결과를 제시하는 것은 아니다.

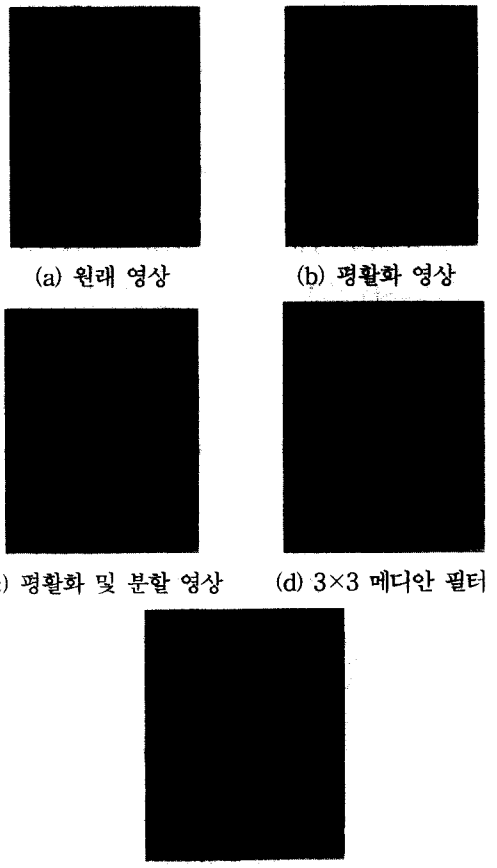
응용 코드는 컨트롤러와 PE 배열의 시뮬레이터를 사용하여 실험하였다. 4개의 연산 병렬 프로세서 장치, 기본형 컨트롤러, 그리고 SUN 웍스테이션을 이용하여 데모 시스템을 구축했다. 각각의 연산 병렬 프로세서 장치는 32×32 PE 배열의 ¼인 256개의 PE를 제공하게 된다. 실험용 데이터 패스 유니트는 연속적인 테스트 영상들을 공급하고 처리된 영상들을 실시간으로 변환시킨다. 이 유니트로 인하여 시스템의 기능성을 검증할 수 있었다.

영상 획득시 생기는 잡음은 평활화 강도를 변화시킴으로써 감소시킬 수 있다.

3×3 커널 $\frac{1}{8}\begin{bmatrix}0&1&0\\1&4&1\\0&1&0\end{bmatrix}$을 지닌 영상을 반복적으로 발산하는 것은 가우스의 평활화 동작과 유사하다. 적용된 컨볼루션의 수는 가우스 필터의 편차를 결정한다. <그림 16>에서 볼 수 있듯이 단순한 평활화는 더 높은 공간의 주파수를 억제하면서 잡음을 감소시킬 뿐 아니라 에지를 무디게 한다. 평활화와 분할 과정은 날카로

운 에지를 보존하면서 잡음을 감소시킨다. 각각의 컨볼루션 전에 각 픽셀의 값은 가장 가까이 이웃해 있는 4개의 픽셀의 값과 비교되어진다. 그 차이가 분할 임계치 때보다 더 크면 3×3 커널은 국부적으로 수정되어 강도의 변화를 보존하게 된다. 예를 들어, 어떤 화소의 값이 임계치 보다 더 많아짐으로써 이웃해 있는 화소의 값과 다르지만 다른 그 밖의 이웃 화소들과는 처음보다 적은 값을 갖게 되어 달라지는 경우,

수정된 3×3 평활화 커널 $\frac{1}{8}\begin{bmatrix}0&1&0\\1&5&0\\0&1&0\end{bmatrix}$이 적용될 것이다. 3×3 커널을 수정하고 적용하는 과정이 각각의 화소마다 실행되어야 하기 때문에 평활화와 분할작업은 화소-병렬 하드웨어에 자연스럽게 병합된다.



(a) 원래 영상 (b) 평활화 영상

(c) 평활화 및 분할 영상 (d) 3×3 메디안 필터

(e) 5×5 메디안 필터

그림 16. 필터링후의 여러 가지 영상처리결과
Fig. 16. Image processing result after filtering.

평활화와 분할과 같은 메디안 필터링(Median Filtering)은 영상 잡음을 줄이기 위해 적용된다. 각 출력 화소의 값은 그 출력 화소에 집중되는 부분의 모든 입력 값의 메디안이다. 메디안 필터링은 날카로운 에지는 유지하고 화소값의 단조로운 변화를 보존하는 반면 곡선의 스파이크는 제거한다. <그림 16(d), (e)>는 메디안 필터링의 효과를 보여주고 있다. 3×3 메디안 필터에 있어서 각 출력 값은 3×3 픽셀 영역에 있는 9개의 입력 값의 메디안이고 5×5 메디안 필터에서의 각 출력 값은 25개의 입력 값의 메디안이다.

4. 기존 시스템과의 비교 결과

<그림 17>에서는 기존 시스템과의 비교 결과를 나타내었다. 화소 병렬 영상 프로세서의 면적 시간 및 에너지 특징은 확실히 다른 구조보다 우수하며 단순한 마이크로프로세서는 폭넓고 다양한 응용분야에 사용되지만 단순한 영상 처리 작업의 경우에는 아주 비효율적이다. DSP는 마이크로프로세서보다 단순한 영상 처리 작업에 더 적합하지만 화소 병렬 영상 프로세서보다는 효율적이지는 못하다. 단순한 영상 처리 작업의 경우 FPGA는 화소 병렬 영상 프로세서보다 다소 더 큰 에너지를 필요로 한다. FPGA는 다양한 응용분야에서 사용되지만 좀 더 복잡한 영상 처리 작업을 수행하는데는 부적합하다.

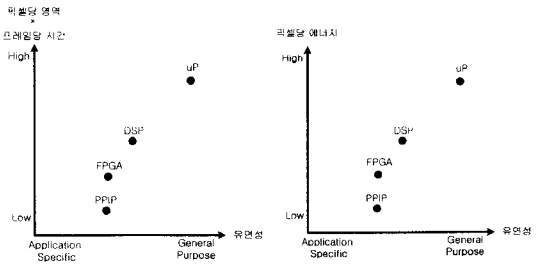


그림 17. 영상처리를 적용한 구조별 특성
Fig. 17. Characteristics of architectures applied to image processing.

V. 결 론

영상 데이터는 A/D 변환기에 의해 한 화소씩 교대로 출력된다. 그러나 데이터를 PE 배열에 신속히 전송하고 그 배열을 효율적으로 이용하기 위해서는 데이터를 재배열할 필요가 있다. 포맷 변환기를 포함하는 하드웨어는 카메라로부터 얻은 실-시간 영상은 물론이고 호스트 컴퓨터로부터 얻은 영상 데이터를 처리할 수 있을 만큼 편리함을 보여 줄 수 있다.

PE 배열이 각 라인마다 단지 256화소를 처리하기 때

문에 만일 디지털 데이터를 출력하는데 있어서 화소당 200ns로 각기 다른 샘플링 비율을 지닌 다른 A/D 변환기가 사용된다면 데이터 패스 보드는 20MHz(50ns)에서 동작한다. 따라서 각각의 메모리 버퍼를 구현하는데 필요한 SRAMs의 수는 절반으로 줄일 수 있다. 각 포맷 변환기마다 2개의 메모리 버퍼가 필요한 것임에도 불구하고 보드상의 SRAM의 총 개수는 변함이 없다. 1개의 화소를 저장하는데 다양한 판독과 기록 주기가 요구되므로 훨씬 더 풍부한 단어를 지닌 SRAM이 메모리 버퍼를 구현하는데 사용된다. 이렇게 해서 하드웨어상의 SRAM 칩수를 줄일 수 있다.

본 논문에서는 포맷 변화기를 이용하여 원래 영상을 평활화, 평활화 및 분할, 메디안 필터링을 하는 작업을 구현하였다. 단순한 평활화는 더 높은 공간의 주파수를 억제하면서 잡음을 감소시킬 뿐 아니라 에지를 무디게 할 수 있으며 평활화와 분할과 같은 메디안 필터링기법은 영상 잡음을 줄이기 위해 적용될 수 있고 날카로운 에지는 유지하면서 스파이크 성분을 제거하고 화소 값에서 단조로운 변화를 유지 할 수 있었다.

참 고 문 헌

[1] Jeffery C. Gealow, Frederick p. Herrmann, Lawrence T. Hsu, and Charles G. Sodini, "System Design for Pixel-parallel Image Processing", IEEE Trans. on VLSI systems, pp. 32~41, Mar. 1996.

[2] F. P. Herrmann and C. G. Sodini, "A 256-element Associative Parallel Processing." in Symp, VLSI Circuits : Dig. Tech. Papers, pp. 99~100, June 1994.

[3] F. P. Herrmann and C. G. Sodini, "A Dynamic Associative Processor for Machine Vision Applications." IEEE Micro, Vol. 12, No. 3, pp. 31~41, June 1992.

[4] 조화현, 최절호, 권병헌, 최명렬, "실시간 처리를 위한 콘트라스트 조정기법," 한국정보처리학회논문지 제7권 제6호, pp. 1988~1995, June. 2000

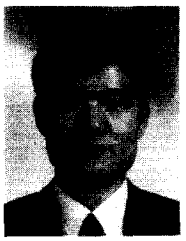
[5] Jonathan Rose, Abbas El Gamal, and Alberto Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays," Proc. of the IEEE, Vol. 81, No. 7, pp. 1013~1029, July 1993.

[6] Xilinx, "The Programmable Logic Data Book," 1994.

[7] D. Bursky, "Programmable Array Mix FPGA and ASIC Blocks," Electronic Design, pp. 69~74, Oct. 14. 1996.

[8] Dinesh Bhatia, "Field Programmable Gate Arrays - a Cheaper Way of Customizing Product Prototypes." IEEE Potentials, 1994.

저 자 소 개



金 玄 起(正會員)

1986년 호서대학교 정보통신과 졸업(공학사). 1992년 호서대학교 대학원 정보통신과 졸업(공학석사). 2002년 청주대학교 대학원 전자공학과 졸업(공학박사). 1996년-현재 극동정보대학 전자통신과 조교수. <주관심분야 : VLSI & CAD, 실시간 정보처리, 컴퓨터 네트워크>

李 天 熙(正會員)

1971년 한양대학교 전자공학과 졸업(공학사). 1975년 성균관대학교 대학원 전자자료처리과 졸업(석사). 1981년 한양대학교 대학원 전자공학과 졸업(석사). 1987년 성균관대학교 대학원 전자공학과 졸업(박사). 1971년 2월 - 1972년 8월 한국 마벨(전자업체). 1972년 9월 - 1977년 2월 수송전기 공업고등학교 교사. 1977년 3월 - 1979년 2월 동양공업전문대학 전자과 전임강사. 1979년 3월 - 현재 청주대학교 전자공학과 교수. 1983년 8월 - 1985년 8월 미국 산호세 캘리포니아 주립대학교 전산과 객원교수. <주관심분야 : VLSI & CAD, System Design>