논 문
59-3-28

임베디드 영상 응용을 위한 GP_SoC

A SoC based on the Gaussian Pyramid (GP) for Embedded image Applications

이 봉 규*
(Bongkyu Lee)

Abstract - This paper presents a System-On-a-chip (SoC) for embedded image processing and pattern recognition applications that need Gaussian Pyramid structure. The system is fully implemented into Field-Programmable Gate Array (FPGA) based on the prototyping platform. The SoC consists of embedded processor core and a hardware accelerator for Gaussian Pyramid construction. The performance of the implementation is benchmarked against software implementations on different platforms.

Key Words : Image pyramid, Multi-resolution, Visual system, Gaussian pyramid

1. 서 론

휴대폰이나 개인정보단말기(PDA)와 같은 개인용 이동기기에 카메라(영상입력 장치)가 보편화되면서, 기존에 데스크탑 PC나 서버 등에서 이루어지던 영상처리/인식 기능을 개인용 이동기기에 구현하려는 연구가 활발히 진행되고 있다. PDA를 이용한 얼굴인식 시스템 [1], 웨어러블 (wearable) 컴퓨터에 개인 얼굴 인식시스템 구축 [2], 그리고 모바일용 포터블 번역기 [3] 등은 영상인식 기능에 대한 구현 예이다. 영상처리 분야로써는 개인용 기기에 저장된 영상의 검색을 지원하는 Thumbnail 브라우징 기능을 들 수 있다 [4]. 그림 1에서 보는 것과 같이 휴대폰에 저장된 다수의 영상에 대해서 화질의 큰 변화없이 축소하여 하나의 화면에 보여 주는 것은 사용자가 필요로 하는 영상을 검색하는데 유용하게 사용될 수 있다. 이러한 영상처리/인식 관련 응용분야에 효과적으로 사용될 수 있는 것이 영상피라미드 (Gaussian Pyramid, GP) [5]이다. GP는 원 영상에 대해서 일정비율로 줄어드는 다단계의 영상들을 만드는 방법으로 영상인식 분야에서는 크기에 무관한 인식 기능을 구현하는데 사용이 가능하고, 영상처리 분야에서는 다수의 영상을 하나의 작은 화면 (그림 1)에 출력할 수 있도록 영상의 크기 변환에 사용이 가능하다.

이러한 GP알고리즘을 임베디드 디바이스에 구현하는 방법은 소프트웨어적 구현 (Fully Software Implementation)과 하드웨어적 구현이 (Fully Hardware Implementation) 있다. 소프트웨적 구현방법은 구현이 용이하다는 장점이 있기 때문에 보편적으로 사용되는 방법이다. 그러나 인터폴레이션 (Interpolation)과 데시메이션 (Decimation)과 같은 실수연산을 사용하는 GP알고리즘을 Float Point Unit (FPU)가 없는 낮은 성능의 하드웨어 자원으로 구성된 임베디드 디바이스에 소프트웨어로 적용할 경우 실시간 처리가 힘든 단점이 있다 [4]. 따라서 GP알고리즘을 수행하는 전용 하드웨어를 설계하여 임베디드 응용에 적용하는 것이 바람직하다 [6]. 이런 이유로 GP알고리즘의 하드웨어적 구현이 많이 연구되어 왔으며 다수의 결과가 보고되고 있다. 그러나 기존의 결과들은 실제 임베디드 응용에 바로 활용하는데 적합하지 못한 구조적인 문제점을 가지고 있다. 이에 본 연구에서는 실제 임베디드 디바이스에 구현되는 응용에 적용이 가능한 단일칩 형태의 새로운 GP_SoC를 구현하는 것을 목표로 한다.

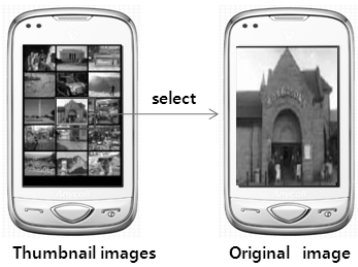


그림 1 개인용 기기에 적용된 영상검색 예
Fig. 1 Selection of the target using thumbnail images

본 논문에서는 GP알고리즘을 필요로 하는 임베디드 영상처리나 패턴인식 응용을 구현하는데 효과적으로 사용할 수 있는 GP_SoC를 설계하는 것을 목적으로 한다. 구현 과정은 다음과 같다. 먼저 목표로 하는 SoC의 설계/구현/검증을 위하여 Field Programmable Gate Array (FPGA) 기반의 개발 플랫폼을 구현한다. 구현되는 플랫폼은 FPGA를 중심으로 외부 메모리 시스템, 디바이스 인터페이스, 버튼 및 LED 등으로 구성된다. 구현된 플랫폼의 FPGA내에 제어용 CPU로 사용할 LEON2 [7], GP 전처리 블록, 메모리 컨트롤러와 버

* 종신회원 : 제주대학교 전산통계학과 교수
E-mail : bklee@cheju.ac.kr
접수일자 : 2009년 12월 17일
최종완료 : 2010년 1월 20일

스 시스템을 통합 구현하고 Register Transfer Level (RTL)에서 검증한다. 구현된 SoC상에서 얻어지는 실제 영상에 대한 GP 구축 결과를 통하여 설계한 SoC가 패턴/영상 인식에 관련된 임베디드 응용에 효과적으로 사용될 수 있음을 보인다.

2. GP 알고리즘 및 관련 연구

GP 알고리즘은 입력 영상을 단계 0으로 하여 정해진 단계별로 일정 비율씩 축소되는 N개의 복수영상 집합을 만든다. 입력 영상 g_0 (C X R 크기)에 대해서 단계 1에 있는 영상 g_1은 일정 비율 (Factor)만큼 줄어든 크기의 영상으로 정의된다. 하위 단계의 영상으로부터 상위 단계의 영상을 구성하는 계산과정은 수식 (1)에 있는 $REDUCE$ [8] 함수로 정의할 수 있다. 여기서 N은 목표 피라미드 단계의 수를 나타내며, M과 w는 내부의 저대역 (low-pass filter)필터에서 사용되는 윈도우 크기 및 확률밀도함수이다. 그리고 C_k, R_k는 k단계에서의 영상의 크기를 나타낸다. 그림 2는 수식 (1)에 나타난 계산과정을 도식화 한 것이다. 그림에 보듯이 $REDUCE$ 함수는 저대역 필터와 스케일링 연산으로 구성이 된다. 실제 설계에서 사용되는 축소비율은 다양한 영상응용을 고려하여 1.2 (이전 단계의 83%에 해당)로 정한다. 만약 영상의 축소에 사용될 경우에는 수식 (2)을 통하여 목표 단계 T_k를 정하고 그 단계까지 반복 처리하여 결과를 얻을 수 있다.

$$g_k(i,j) = REDUCE(g_{k-1}(i,j)) = \sum_{m=-\frac{M}{2}}^{m=\frac{M}{2}} \sum_{n=-\frac{M}{2}}^{n=\frac{M}{2}} w(m,n) g_{k-1}(2i+m, 2j+n) \quad (1)$$

$$0 \le k < N, 0 \le i < C_k, 0 \le j < R_k$$

$$T_k \approx \sqrt[1.2]{\frac{O_{size}}{T_{size}}} \quad (O_{size}\text{:원 영상크기, } T_{size}\text{: 목표 영상크기}) \quad (2)$$

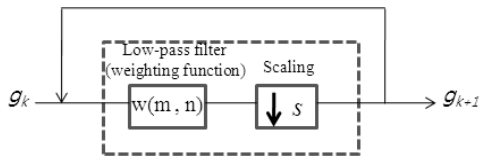


그림 2 순환적 GP생성 블록도
Fig. 2 Iterative pyramid generation

GP알고리즘을 임베디드 디바이스에 적용하기 위한 하드웨어 설계 및 구현에 대한 연구 결과가 [8, 9, 10, 11, 12] 등에 나타나 있다. [8] 에서는 크기불변 특징을 추출과정에서 사용할 수 있는 GP 하드웨어를 제안하였다. [9]에서는 얼굴인식을 위한 얼굴검출 하드웨어의 일부분으로 Image Pyramid Generation Unit (IPGU) 하드웨어를 구현하였다. [10, 11]에서는 2단계 및 3단계의 GP를 생성하는 하드웨어를 제안하였으며 [12]에서는 파이프라인 방식의 하드웨어 구조를 제안하였다. 그러나 이런 여러 가지 선행 연구 결과를 실제 응용에 적용하는 것은 어렵다. 그 이유는 구현된 하드웨어가 고정되어 있기 때문에 다양한 크기의 영상이나 필요한 단계 수 및 변환 비율에 대한 가변성을 가지지 못하기 때문에 다양한 크기의 영상에 대해서 동적으로 적용 할 수 없다. 또한 목표단계를 동적으로 변화시키는 것이 불가능하다. 이런 문제점은 응용에 따라서 하드웨어를 가변적으로 제어할 수 있는 구조를 설계함으로써 해결이 가능하다. 본 연구에서는 GP알고리즘을 위한 하드웨어 블록을 소프트웨어 수행을 위한 기본 CPU 코어, 영상의 입력 및 출력 기능을 담당하는 디바이스 제어 블록, 내부 버스 인터페이스 등과 함께 하나의 칩에 집적시킨 GP_SoC를 통하여 문제점을 해결한다.

3. SoC 개발을 위한 개발 플랫폼 구성

2.1 개발 플랫폼

SoC 개발을 효과적으로 진행하기 위해서는 적절한 개발 플랫폼의 활용이 매우 중요하며 개발 기간의 단축 및 검증의 편리성을 얻을 수 있다. 현재 일반적인 개발 플랫폼은 프로세서가 장착되고 메모리, 주변 제어 장치 및 관련 소프트웨어를 가지는 구조가 된다. 그러나 이런 개발 플랫폼은 본 논문에서와 같이 범용 프로세서 코어의 구현까지를 포함하는 경우에는 적합지 못하다. 따라서 기존의 IP들을 이용하여 새로운 기능의 SoC를 구현하는데 효과적으로 사용할 수 있도록 새로운 형태의 FPGA기반 플롯폼을 제작하여 실제 개발과정에서 사용한다. FPGA에 LEON2 코어를 비롯하여 설계되는 모든 하드웨어 블록들을 구현할 수 있기 때문에, 개발 플랫폼은 설계한 전체 하드웨어를 (프로세서, 사용자 하드웨어 블록들) 단일 칩 형태로 개발 및 검증할 수 있는 기능을 제공한다. 개발 플랫폼에서 지원하는 외부 장치는 LCD와 카메라이다. 그 이유는 목표로 하는 SoC가 영상처리/인식에 관련되기 때문에 목표 SoC 자체에 LCD와 카메라를 하드웨어적으로 제어할 수 있는 기능이 필요하기 때문이다. 또한 플랫폼에는 외부 데이터 및 프로그램 저장을 위하여 FPGA외부에 SRAM기반의 메모리와 플래시 기반의 메모리를 가지고 있다. 그림 3은 실제 구성된 개발 플랫폼의 블록도와 실제 사진을 보여주며 표 1은 구현된 개발 플랫폼의 주요 사양을 보여준다.

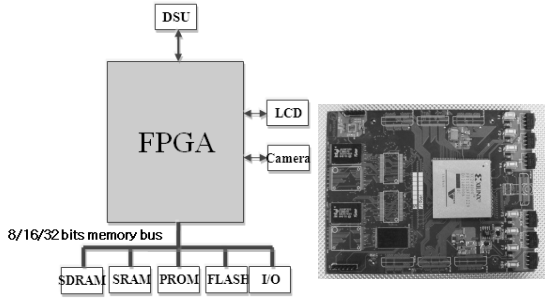


그림 3 개발 플랫폼
Fig. 3 The prototyping platform

표 1 구현된 개발플랫폼의 주요 특징
Table 1 Major components of the platform

구성 요소	목적
FPGA(XILINX X2CV8000)	하드웨어 구현
PROM (XILINK XCF32PV048)	Fixed 로직 저장
SRAM(SAMSUNG, 128M)	데이터 저장
FLASH(Intel strata, 8M)	부트 프로그램, 응용프로그램 저장
카메라 (MICRON MT9V112)	영상 입력
LCD (SAMSUNG)	출력용
시리얼 인터페이스 (DSU)	다운로딩/디버깅

SoC의 제어 프로세서로 선택된 LEON2 프로세서는 32비트 프로세서로써 VHDL 소스 형태로 제공되어 설계자들이 자신의 응용에 맞추어 재구성(Reconfiguration)을 할 수 있는 프로세서이다. 구조적인 측면에서 LEON2는 코어 부부분과 시스템버스 및 기본 하드웨어 모듈들(인터럽트 콘트롤러, 타이머, UART, I/O)로 구성되어있다 (그림 4). 정수 유닛(Integer Unit)은 SPARC V8 구조를 가지는 5단계의 명령어 파이프라인을 가진다. 내부 캐시(cache)는 명령어와 데이터로 분리되어 있으며, 2-32 레지스터 집합을 지원하는 구조이다. 메모리 콘트롤러는 외부에 있는 메모리를 제어하는 유닛으로 8/16/32 비트 모드로 구성할 수 있으며, 최대 2Gbyte 주소공간을 관리할 수 있다. 구성요소간의 인터페이스를 위하여 LEON2에서는 표준의 AMBA AHB/APB 버스를 지원한다. AHB버스는 고속의 데이터 전송에 사용되어지며, APB버스는 주로 코어 외부에 있는 주변장치들의 온-칩 레지스터에 접근하는데 사용된다. AHB/APB 브리지(Bridge)는 두 버스 시스템 (AHB, APB)간의 인터페이스를 담당한다. 이들 장치들은 메모리 맵 방식으로 주소 공간을 할당받는 메모리 맵 (Memory Map)방식으로 수행된다.

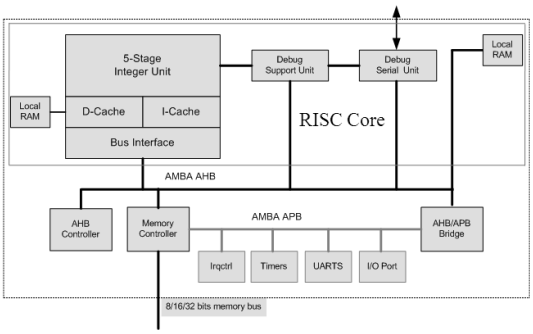


그림 4 LEON2에 대한 블록도
Fig. 4 The block diagram of LEON2

구현되는 SoC는 CPU를 가지고 있으며, 이를 초기화 시키고 운영하기 위해서는 운영 소프트웨어가 필요하다. 또한 SoC에 새로운 기능을 소프트웨어적으로 추가할 경우, 이를 개발할 수 있는 환경을 만들어 주어야 한다. 이런 점을 고려하여 개발 SoC에 맞는 최적화된 운영 환경과 소프트웨어 개발 환경을 동시에 지원하도록 하기 위해 그림 5와 같은 부터로더 기반의 소프트웨어 개발체계를 구축한다. 이 체계를 이용하면 SoC에 구현되는 기능 중 소프트웨어로 구현되는 부분은 이 방법에 의해서 부터로더에 직접 삽입하여 구현할 수 있기 때문에 추가적인 소프트웨어 오버헤드(Overhead)를 가지지 않도록 할 수 있다.

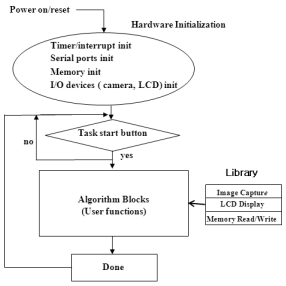


그림 5 응용시스템 흐름도
Fig. 5 The flowchart of the application system

4. SoC 설계

구현되는 하드웨어를 응용에 따라서 동적으로 활용할 수 있는 GP_SoC의 구현을 위해서는 소프트웨어 수행을 위한 기본 CPU 코어, 영상의 입력을 담당하는 이미지 센서 제어 블록, 내부 버스 인터페이스를 포함하는 구조를 가져야 한다. 이러한 요소들을 모두 고려하여 설계한 LEON2 중심의 SoC의 블록 다이어그램이 그림 6에 나타나 있다. LEON2의 코어부분은 Gaisler Research Group에서 제공하는 VHDL 소스를 기반으로 구현한다. CPU외에 내부에 구현된 하드웨어 요소는 버스 시스템을 위한 AHB_APB 브리지 (Bridge), 카메라 인터페이스 콘트롤러 및 메모리 콘트롤러이다. CPU 부분과 다른 하드웨어 블록간의 인터페이스를 위한 버스 시스템은 AMBA AHB/APB을 사용한다. 실제 설계된 AHB 버스의 경우 마스터 (Master)와 슬레이브 (Slave)의 구성에 따라서 AHB_1과 AHB_2로 분리하여 구현한다. 이들 2층 버스들은 디코더에 의해서 하나의 시스템 버스로 상호 연결이 되는 구조를 가진다. APB 버스 시스템은 내부 콘트롤러와 I^2C 인터페이스를 지원하는 외부 카메라와의 연결을 위하여 사용되는 것으로 카메라로부터 들어오는 영상을 외부의 메모리 시스템 (SDRAM)에 저장하는 역할을 한다. 또한 표준의 I^2C를 사용하였기 때문에 이 인터페이스에 접속이 가능한 카메라는 모두 제어가 가능하다.

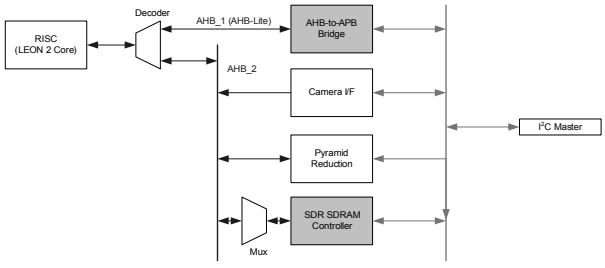
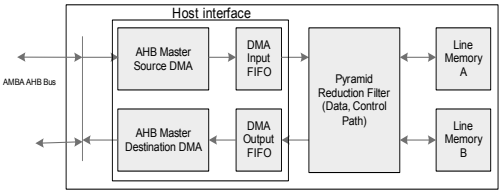


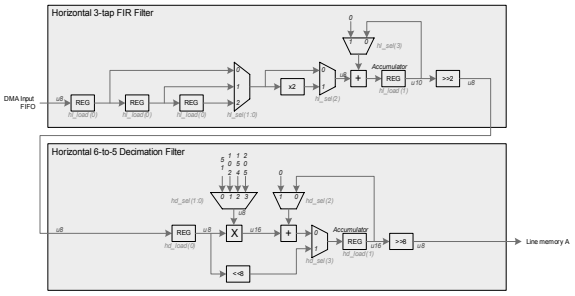
그림 6 구현된 내부 구조도
Fig. 6 The block diagram of the implementation

그림 7은 설계된 GP 구축 알고리즘을 위한 Pyramid Reduction (PR) 블록의 하드웨어 구조와 핵심 요소인 Pyramid Reduction Filter (PRF) 모듈의 RTL 다이어그램을

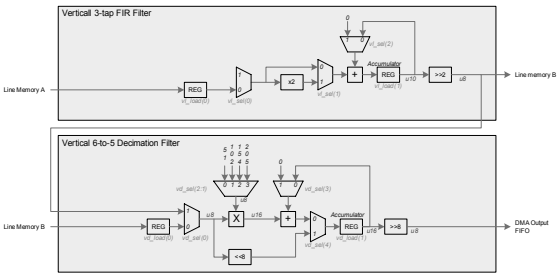
보여준다. PR 블록은 메모리 접근 및 버스 인터페이스에 사용되는 'Host interface' 부분과 실제 피라미드 연산을 수행하는 PRF 모듈로 구성된다. 'Host interface' 모듈은 PRF와 다른 콘트롤러 사이의 인터페이스를 제공하는 것으로, 직접 메모리 접근 (DMA) 관련 회로인 'Source DMA' 블록과 'Destination DMA'로 구성된다. DMA 사용함으로써 PRF가 CPU를 거치지 않고 직접 메모리에 접근이 가능하게 하기 위한 설계이다. 'Source DMA' 블록은 외부 SDRAM메모리에 있는 영상 데이터를 PRF에 전달하는 기능을 수행한다. PRF는 입력으로 받은 2차원 영상에 대해서 3-탭 저대역 필터로 컨볼루션 (Convolution)을 실행함으로써 피라미드 연산을 수행하다. 실제 필터의 구현은 2개의 1차원 수평 및 수직 필터로 구성이 된다.



(a) GP 블록 다이어그램



(b) 수평방향 (Row) 필터링 및 데시메이션



(c) 수직방향 (Column) 필터링 및 데시메이션

그림 7 PR 연산 블록

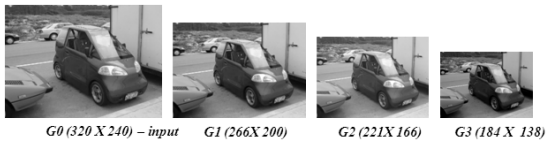
Fig. 7 The block diagram of the pyramid reduction block

전체적인 PRF의 데이터 흐름 (Data Flow)은 다음과 같다. 'Source DMA' 블록은 SRAM에 있는 입력 영상에서 매번 16 픽셀을 'Input-FIFO' 버퍼로 읽어온다. 먼저 수평방향 PRF는 버퍼에 있는 픽셀을 대상으로 수평(Row)방향의 컨볼루션과 데시메이션을 수행하여 얻어진 결과영상을 내부 메모리인 'A'에 저장한다. 수직방향 PRF는 'A'에 저장된 픽셀을 대상으로 수직(column) 방향으로 컨볼루션과 데시메이션을 수행한 후, 결과를 내부 메모리 'B'에 저장한다. 내부 메모리 'B'에 저장된 픽셀들은 'Output-FIFO' 버퍼로 이동된 후, DMA에 의해서 최종적으로 SRAM에 저장된다. 이런 연산 과정은 영상인식을 위하여 전체 피라미드를 구축할 때에는 탐색 패턴의 크기에 따라서 가변적으로 반복되고, 특정 단계의 영상을 얻을 때에는 수식 (2)에서와 같은 방법으로 얻어지는 단계까지 반복된다.

5. 구현 및 검증

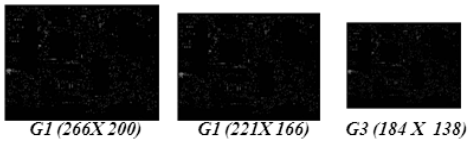
GP_SoC의 모든 하드웨어 요소들은 VHDL로 구현되어 하나의 FPGA에 통합적으로 매핑 (Mapping)되어 구현되기 때문에, 단일 칩에서와 같은 수준에서 하드웨어/소프트웨어 연계 동작과정을 검증할 수 있다. 이 장에서는 구현된 SoC 상에서 실제 입력 영상을 받아 피라미드 영상들을 구성하는 실험을 통하여 얻어지는 결과를 바탕으로 유용성 (데스크톱의 결과와 비교) 및 효율성 (소프트웨어 구현과 비교한 속도 향상)을 검증한다.

구현되는 영상 시스템은 이미지 센서 (마이크론사의 MT9V112 [13])를 이용하여 320X240크기의 입력 영상을 획득하고 PR 블록을 이용하여 13단계 피라미드 영상을 구성한 후, 결과를 SRAM 메모리에 저장하는 기능을 수행한다. 이러한 과정을 통하여 얻어지는 피라미드 영상들에 대한 영상 품질에 대한 평가를 통하여 실제 응용에 사용될 수 있는지 확인한다. 평가를 위하여 GP알고리즘을 FPU를 가지는 데스크톱 (Intel Xeon)에서 소프트웨어로 구현하여 얻은 결과들과 비교한다. GP_SoC를 통하여 처리한 피라미드 영상의 일부가 그림 8의 (a)에 나타나 있으며, 그림 8의 (b)는 데스크톱에서 얻은 결과를 보여준다. 그림 8의 (c)에서는 정확한 분석을 위하여 같은 단계의 두 영상간의 차이를 보여준다 (검은 부분 : 차이 없음, 흰 부분 : 차이 있음). SoC에서의 연산정밀도와 데스크톱의 정밀도 차이로 인하여 미세한 품질의 변화 (전체 픽셀의 1% 이내)를 확인할 수 있으나, 매우 적은 편차를 보이기 때문에 기존의 데스크톱과 유사한 품질의 영상을 GP_SoC를 통하여 얻을 수 있음을 확인하였다.



(a) SoC에서 생성된 피라미드 일부영상들

(b) 데스크톱에서 생성한 일부 영상들



(c) 결과 영상의 차이

그림 8 실험 결과

Fig. 8 Experimental results

유용성과 더불어 구현된 SoC에 대한 성능 척도로 사용한 것은 효율성이다. GP알고리즘을 하드웨어 로직으로 구성함으로써 얻어지는 성능향상을 객관적으로 확인하기 위하여 SoC가 알고리즘을 수행하는데 필요한 실행시간을 측정한다. 또한 비교를 위하여 같은 영상에 대한 피라미드 영상 구성 알고리즘을 개발플랫폼 (LEON2), 상용 개발플랫폼 (ARM920T, LINUX), 데스크톱 (Intel Xeon CPU, Linux), Sun Enterprise 450 (UltraSparc II, Unix) 등에 'C'언어로 작성한 소프트웨어로 구현하고, 실행 시간을 측정한다. 320X240크기의 입력 영상에 대한 피라미드 (13단계) 구축에 소요되는 시간을 측정한 결과가 표 2에 나타나 있다. 실험결과 GP_SoC에 구현된 시스템은 320X240 크기의 영상에 대해서 0.16초를 소요하였다. 표 2에서 보듯이 LEON2에 소프트웨어로 구현된 시스템은 63초를 요구하는 것으로 나타났기 때문에 하드웨어를 통하여 소프트웨어에 비하여 393배의 성능 향상을 이룰 수 있음을 확인하였다. 이러한 분석결과는 구현된 GP_SoC를 활용하면 소프트웨어 방식으로는 불가능한 실시간 영상 관련 응용을 임베디드 시스템에 효과적으로 구현할 수 있음을 보여준다.

표 2 성능 평가 결과

Table 2 Results of Performance evaluations

구현방법	Clock속도	소요시간(sec)
GP_SoC (HW)	30MHZ	0.16
GP_SoC (SW)	30MHZ	198
ARM920T(SW)	300MHZ	38
Intel Xeon (SW)	3.06GHZ	0.021
UltraSparc II (S/W)	400MHZ	0.223

5. 결 론

본 논문에서는 낮은 하드웨어 사양을 가지는 스마트 디바이스에 영상 인식/처리에 관련된 응용을 효과적으로 구현할 수 있는 GP_SoC를 제안하고 설계하였다. 설계 및 검증과정에서는 FPGA를 중심으로 구성된 새로운 형태의 개발 플랫폼을 이용하여 단일 칩과 같은 환경에서 RTL에서 기능 및 작동을 검증하였다. 또한 영상 획득에서 피라미드 구성까지의 모든 과정을 GP_SoC에 구현하고 성능을 측정하는 실험을 통하여 GP_SoC의 효율성 및 유용성을 증명함으로써 실제 임베디드 영상시스템 구축에 활용이 가능함을 보였다.

참 고 문 헌

[1] J. Yang, X. Chen, W. Kunz, "A PDA-based Face Recognition System", Proceedings of WACV 2002, 2002.

[2] Jong Bae Kim, "A personal identity annotation overlay system using a wearable computer for augmented reality," IEEE Transactions on Consumer Electronics, vol. 49, no. 4, pp. 1457-1467, Nov., 2003.

[3] H. Nakajima, Y. Matsuo, M. Nagata and K. Saito, "Portable Translator capable of Recognizing Characters on Signboard and Menu Captured by built-in camera," Proc. of the ACL Interactive Poster and Demonstration Sessions, pp. 61 - 64, June, 2005.

[4] A. Greaves, A. Hang and E. Rukzio, "Picture Browsing and Map Interaction using a Projector Phone," Proceeding of MobileHCI 2008, pp. 527-530, Amsterdam, the Netherlands, 2008.

[5] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," IEEE Transactions on communications, vol. 31, no.4, pp. 532 - 540, April, 1983.

[6] P. G.. D. Valle, D. Atienza, G. Paci and F. Poletti, "Application of FPGA Emulation to SoC Floorplan and Packaging Exploration," XXII Conference on Design of Circuits and Integrated System, pp. 236 - 240.

[7] LEON2 processor user's manual, Gaisler Research, http://www.gaisler.com

[8] B. Blair and C. Murphy, "Difference of Gaussian Scale-Space Pyramids for SIFT Feature Detection," Complex Digital Systems Design, Final report, 2007.

[9] O. Sims and J. Irvine, "An FPGA implementation of pattern-selective pyramidal image fusion," Proceedings of 2006 international conference of field programmable logic and application, Mardrid, Spain, August 2006.

[10] Theocharides, G. Link, N.Vijaykrishnan, M. J. Irwin and W. Wolf, "Embedded Hardware Face Detection", Proceedings of the 17th International Conferenceon VLSI Design (VLSID'04), Jan., 2004.

[11] N. Petterson and L. Petterson, "Online stereo calibration using FPGAs," IEEE Proceedings of Intelligent vehicles symposium, 2005.

[12] A. Darabiha, W. J. MacLean and J. Rose, "Reconfigurable hardware implementation of a phase-correlation stereo algorithm," Machine Vision and Applications, vol. 17, no. 2, pp. 116 - 132, 2006.

[13] MT9V112 manual, Micron Technology Inc., http://www.micron.com

저 자 소 개

이 봉 규 (李 鳳 奎)

제주대학교 전산통계학과 교수

Tel : (064)754-3593

E-mail : bklee@venus1.cheju.ac.kr

관심분야 : 영상처리 SoC 설계