논문 2007-44SD-4-6

얼굴 검출을 위한 SoC 하드웨어 구현 및 검증

(A design and implementation of Face Detection hardware)

이 수 현*, 정 용 진**

(Su-Hyun Lee and Yong-Jin Jeong)

요 약

본 논문에서는 실시간 처리를 위한 얼굴 검출 알고리즘의 하드웨어 엔진을 설계하고 검증하였다. 얼굴 검출 알고리즘은 주어진 이미지에서 학습된 얼굴의 특징데이터를 통하여 얼굴의 대략적인 위치를 찾는 연산을 수행한다. 얼굴 검출 알고리즘을 하드웨어 구조로 설계하기 위해 Integral Image Calculator, Feature Coordinate Calculator, Feature Difference Calculator, Cascade Calculator, Window Detector 등의 5 단계로 구조를 나누었으며, On-Chip Integral Image memory 와 Feature Parameter Memory를 설계하였다. 삼성전자의 S3C2440A 프로세서 칩과 Xilinx사의 Virtex4LX100을 이용하여 검증 플랫폼을 구축하고, CCD카메라를 통하여 실제 얼굴의 영상을 받아들여 얼굴 검출을 실시간으로 구동시켜 검증하였다. 설계된 하드웨어는 Virtex4LX100 FPGA를 타겟으로 합성 시에 3,251 LUTs 를 사용하고, 24MHz의 동작 속도에서 검색 윈도우의 이동 간격에 따라 프레임 당 1.96~0.13 초의 실행속도를 가진다. 그리고 매그나칩 0.25um ASIC 공정으로 제작 시 41만 게이트 (Combinational area 약 34.5만 게이트, Noncombinational area 약 6.5만 게이트)의 크기를 가지며, 100MHz의 동작 속도에서 프레임 당 0.5초 미만의 실행 속도로, 임베디드 시스템의 실시간 얼굴 검출 솔루션에 적합함을 보여준다. 실제 XF1201칩의 일부 모듈로 구현되어 동작함이 확인되었다.

Abstract

This paper presents design and verification of a face detection hardware for real time application. Face detection algorithm detects rough face position based on already acquired feature parameter data. The hardware is composed of five main modules: Integral Image Calculator, Feature Coordinate Calculator, Feature Difference Calculator, Cascade Calculator, and Window Detection. It also includes on-chip Integral Image memory and Feature Parameter memory. The face detection hardware was verified by using S3C2440A CPU of Samsung Electronics, Virtex4LX100 FPGA of Xilinx, and a CCD Camera module. Our design uses 3,251 LUTs of Xilinx FPGA and takes about 1.96~0.13 sec for face detection depending on sliding-window step size, when synthesized for Virtex4LX100 FPGA. When synthesized on Magnachip 0.25um ASIC library, it uses about 410,000 gates (Combinational area about 345,000 gates, Noncombinational area about 65,000 gates) and takes less than 0.5 sec for face realtime detection. This size and performance shows that it is adequate to use for embedded system applications. It has been fabricated as a real chip as a part of XF1201 chip and proven to work.

Keywords: Face Detection, Integral Image, AdaBoost Cascade

I. 서 론

얼굴 인식은 중요 생체 인식 기술 중의 하나로써, 지문 인식과 같은 다른 생체 인식 기술과는 달리 비접촉식 방법이라는 장점을 가지고 있으며, 금융기기, 로봇, 출입통제, 감시 카메라, 모바일기기 및 자동차 등 다양한 분야에서 활용될 수 있는 차세대 인식기술로서 관심이 집중되고 있다.

얼굴 인식 시스템은 크게 얼굴 검출과 얼굴 인식의 두 부분으로 나눌 수 있다. 얼굴 검출은 입력된 영상으로부터 얼굴의 후보영역을 검출하는 부분과 검출된 후보영역으로부터 눈의 위치를 찾아 정확한 얼굴의 위치를 찾고 검증 하는 과정을 얘기하며, 얼굴 인식 부분에서는 검색된 얼굴 영상을 정규화하는 부분과 데이터베이스에 저장된 대상과 비교하여 판단하는 과정이 이루어진다.

* 학생회원, ** 정회원, 광운대학교 전자통신공학과
(Dept of Electronics and Communication Engineering Kwangwoon University)

※ 본 연구는 서울시의 나노 SoC 클러스터 사업과 정보통신부의 선도개발 기술 사업의 지원으로 이루어졌습니다.

접수일자: 2006년8월24일, 수정완료일: 2007년3월14일

본 논문에서 구현한 내용은 얼굴 검출에서 눈 검출 및 얼굴 검증 부분을 제외한 얼굴 후보 영역 검출이다. 얼굴 후보 영역 검출은 얼굴 인식을 위한 얼굴 검출의 주요 알고리즘으로써 얼굴의 대략적인 위치를 파악하는 알고리즘이다. 얼굴 검출은 데이터 입력 방식에 따라 2차원 영상처리[1][2]와 3차원 영상처리[3]로 나눌 수 있으며, 데이터의 특징에 따라 컬러 데이터를 사용하여 얼굴 색상을 찾아 검출하는 방식[4][5][6]과, 흑백 데이터를 사용하여 얼굴의 윤곽 또는 특징 점 등을 사용하는 방식[1][2] 등으로 나눌 수 있다.

현재까지 구현된 얼굴 검출 알고리즘은 주로 일반 PC기반에서 소프트웨어로 구현되었으며, 임베디드 환경에서는 프로세서의 성능 때문에 실시간 처리가 어려웠다. 본 논문에서는 이차원 흑백 영상을 입력을 받아, 학습된 얼굴의 특징 점을 이용하여 얼굴 후보 영역을 검출하는 알고리즘[1]을 하드웨어로 설계하였으며, 이를 FPGA로 구현하여 임베디드 환경에서 실시간 처리가 가능함을 검증하였다. 논문의 구성은 다음과 같다. Ⅱ장에서는 사용된 얼굴 검출 알고리즘에 대해 설명하고, Ⅲ장에서는 구현된 하드웨어의 구조를 알아본다. Ⅳ장에서는 검증 및 성능을 분석하며, Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. 얼굴 검출 알고리즘

본 논문에서는 현존하는 많은 얼굴 검출 알고리즘 중 하드웨어 설계와 성능을 고려하여 Paul Viola 와 Michael Jones가 제안한 Robust Real-Time Object Detection 방법[1]을 이용하였다. 해당 알고리즘은 2차원 8-Bit 흑백영상을 사용하여 학습된 특징 데이터(Feature Data)를 중심으로 비교 연산하여 얼굴을 검출하는 알고리즘으로 3차원 영상을 사용한 알고리즘[3], 컬러 영상을 사용한 알고리즘[4][5][6] 등과 비교하여 하드웨어로 설계할 경우 요구되는 메모리 용량에서 많은 차이를 보일 뿐만 아니라, 연산처리의 병렬화를 통한 처리 성능 향상 면에서 더 효율적이다.

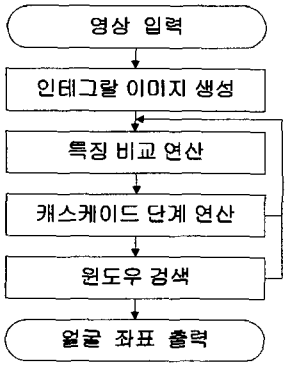


그림 1. 얼굴 검출 순서도
Fig. 1. Flow Chart of Face Detection.

해당 알고리즘은 그림 1에 보이는 바와 같이 입력된 영상 데이터를 적분하여 인테그랄 이미지(Integral Image)를 만들고, 이미지 크기 내에 존재하는 윈도우를 최소부터 최대 크기까지 이동하면서 특징 데이터(Feature Data)를 이용하여 검색한다. 얼굴 영역 검출을 위해 8개 형태의 Haar-like 특징[7]을 이용한 학습 알고리즘으로 생성한 파라미터(Parameter)를 사용하며, 특징 데이터의 연산 시간을 빠르게 하기 위하여 인테그랄 이미지를 사용한다. 그리고 보다 빠른 검출을 위해 영상내의 거대한 특징 데이터 집합으로부터 중요한 작은 특징 집합을 선택하여 연산량을 줄이는 방법인 AdaBoost 방법을 이용한 캐스케이드(Cascade) 구조를 사용한다[1][2].

1. 인테그랄 이미지 (Integral Image)

인테그랄 이미지는 영상 데이터를 얼굴 검출 알고리즘의 빠른 영상처리를 위해 만든 적분영상으로 그림 2와 같이 영상 내에서 특정 사각형 영역 내에 존재하는 픽셀의 합을 식 (1)을 이용하여 변환한다. 식 (1)에서 $i(x,y)$는 좌표 (x,y)상의 영상 픽셀 값이며, $ii(x,y)$는 좌표 (x,y)상의 인테그랄 이미지 픽셀 값이다.

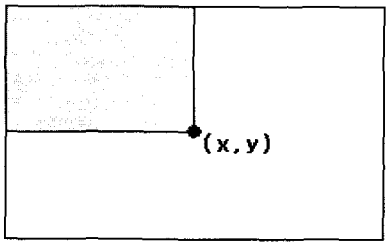


그림 2. 인테그랄 이미지
Fig. 2. Integral Image.

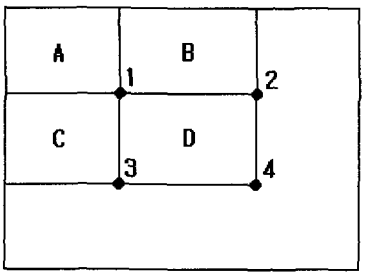


그림 3. 인테그랄 이미지를 이용한 연산의 예
Fig. 3. Example of Calculation using Integral Image.

$$ii(x,y)=\sum_{x'\le x,\,y'\le y} i(x',y') \qquad (1)$$

인테그랄 이미지를 이용하면 그림 3과 같이 특정 영역의 픽셀의 합을 식 (2)와 같이 간단히 연산 할 수 있다. 식(2)에서 $Space\,D$는 그림 3의 D공간의 모든 픽셀 값의 합이다.

$$Space\,D(Sum\,of\,Pixel) = ii(4)+ii(1)-ii(2)-ii(3) \qquad (2)$$

2. 특징(Feature) 비교연산

본 논문에서는 얼굴을 검출하기 위해 Papageoriou 등에 의해 제안된 Haar-like 특징[7] 중에서, Paul Viola 와 Michael Jones 의 Rapid object detection using a boosted cascade of simple features[8]에서 사용한 특징과 Lienhart, R 과 Maydt, J 의 An extended set of Haar-like features for rapid object detection[9]에서 사용한 특징을 함께 사용하였다. 그림 4에 얼굴 검출을 위해 사용된 특징의 형태를 나타내었다. 얼굴 검출을 위해 구성된 8개의 특징으로 AdaBoost 방법을 적용하여 학습시켜 51,392개의 특징 데이터를 만들었으며, 연산량을 줄이고, 검출 속도를 높이기 위해 51,392개의 데이터 중 검출 성능이 높은 674개를 추출하여 사용하였다.

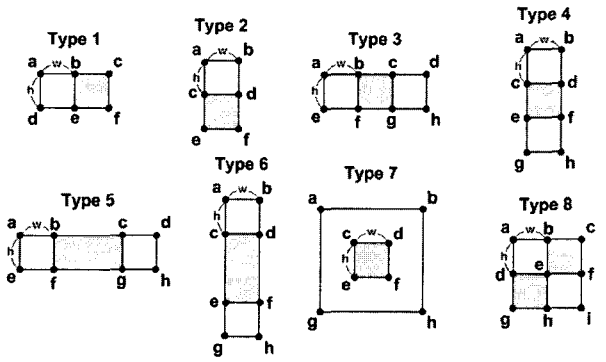


그림 4. 특징 형태
Fig. 4. Feature Type.

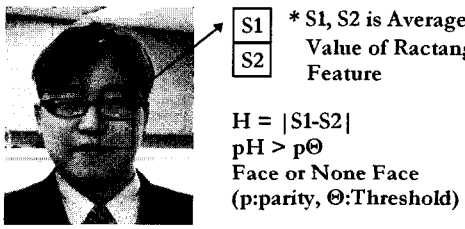


그림 5. 특징을 이용한 얼굴검출 연산의 예
Fig. 5. Example of Face Detection using Feature.

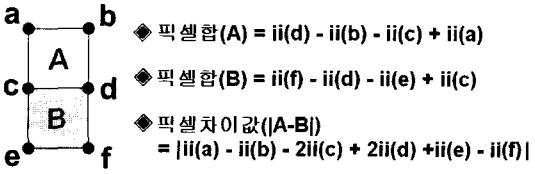


그림 6. 인테그랄 이미지를 이용한 특징 연산의 예
Fig. 6. Example of Feature Calculation using Integral Image.

학습된 특징 데이터를 이용하여 얼굴을 찾는 방법은 그림 5와 같이 특징의 두 영역의 차이 값을 문턱값(Threshold)과 비교하여 얼굴인지 아닌지를 판단한다. 이때 그림 6과 같이 인테그랄 이미지를 사용함으로써 두 영역의 픽셀의 차이 값을 빠르게 연산을 빠르게 할 수 있다.

3. 캐스케이드 단계 연산 및 윈도우 검출

본 단계에서는 특징 데이터 연산을 줄이기 위하여 캐스케이드라는 조건적 처리 단계를 사용하였다. 그림 7과 같이 AdaBoost 방법을 적용하여 학습시킨 특징 데이터를 캐스케이드 구조로 만들어 적용하였다. 캐스케이드 구조는 여러 개의 단계를 나누고 단계별로 특징 데이터의 수를 다르게 하여 수행하는 방법으로, 앞 단계에 변별력이 높은 특징을 분포시키고, 다음 단계로 넘어 갈수록 앞 단계보다 변별력이 낮은 특징을 보다 많이 분포시키는 방법이다. 이러한 방법을 이용할 경우 대부분 처음 스테이지에서 배경부분(얼굴이 아닌 영상)

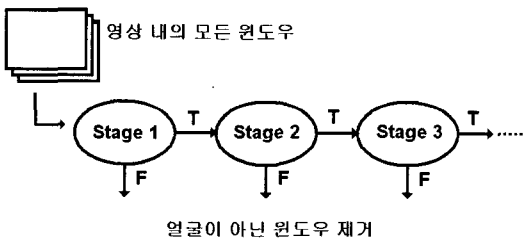


그림 7. 얼굴 검출 연산을 위한 캐스케이드 구조
Fig. 7. Cascade Structure for Face Detection.

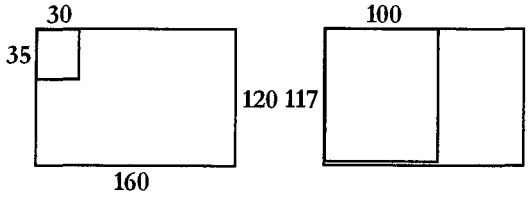


그림 8. 160x120 영상에서의 최소 윈도우와 최대 윈도우
Fig. 8. Maximum & Minimum Window Size in 160×120 Image.

은 제거되는 기능을 가지고 있어, 캐스케이드 방법을 적용하지 않았을 때 보다 월등히 빠른 연산을 수행할 수 있다.

얼굴 검출을 위한 케스케이드 단계의 연산과정은 다음과 같다.

① 특징 파라미터 데이터를 이용하여, 먼저 비교할 특징의 좌표를 계산해 낸다.

② 좌표 정보를 이용하여, 인테그랄 이미지 값을 읽어들이고, 특징의 차이 값을 계산한다.

③ 특징의 차이 값을 문턱값과 비교하여, 특징 별 가중치의 캐스케이드 단계별 누적 값을 구하고 해당 캐스케이드 단계의 문턱값과 비교를 한다.

④ 해당 캐스케이드 단계의 문턱값을 넘지 못할 경우 다음 윈도우 연산으로 넘어가고, 문턱값을 넘을 경우에는 해당 윈도우의 다음 캐스케이드 단계의 특징 비교 연산을 수행한다.

⑤ 전체 캐스케이드 단계와 문턱값을 모두 넘었을 경우, 해당 윈도우에 얼굴이 존재 하고 있음으로 간주하고 윈도우의 좌표정보와 가중치의 최종 합산값을 저장 한다.

이 과정을 160x120 크기의 입력영상을 이용하여, 그림 8과 같이 얼굴 검증 및 인식에서 사용될 수 있는 최소의 이미지 크기인 30x35 크기의 최소 윈도우부터 100x117 크기의 최대 윈도우까지 약 0.1배씩 증가하며, 영상 전체를 1~8픽셀 간격으로 이동하여 검색한다. 그리고 최종 문턱값을 넘은 윈도우 중에서 가장 높은 가중치를 가지는 윈도우가 얼굴 영역이라고 결정한다.

III. 하드웨어 구조 설계

본 논문에서는 해당 알고리즘을 하드웨어로 설계하기 위해 고정소수점 모델의 구현과 성능 분석[10]을 통하여, 데이터의 크기를 정하고 연산기의 구조를 설계하였다. 메모리에 입력된 영상으로부터 얼굴의 좌표가 추출되기까지의 전체 알고리즘을 하드웨어에서 구현이 편리하도록, 그림 9와 같이 각 기능별로 Integral Image Calculator, Feature Coordinate Calculator, Feature Difference Calculator, Cascade Calculator, Window Detector 등의 5 개의 모듈로 분류하였으며, 각 모듈의 기능은 앞에서 설명한 알고리즘의 순서와 같다.

1. Integral Image Calculator

그림 10에 설계한 Integral Image Calculator 블록의

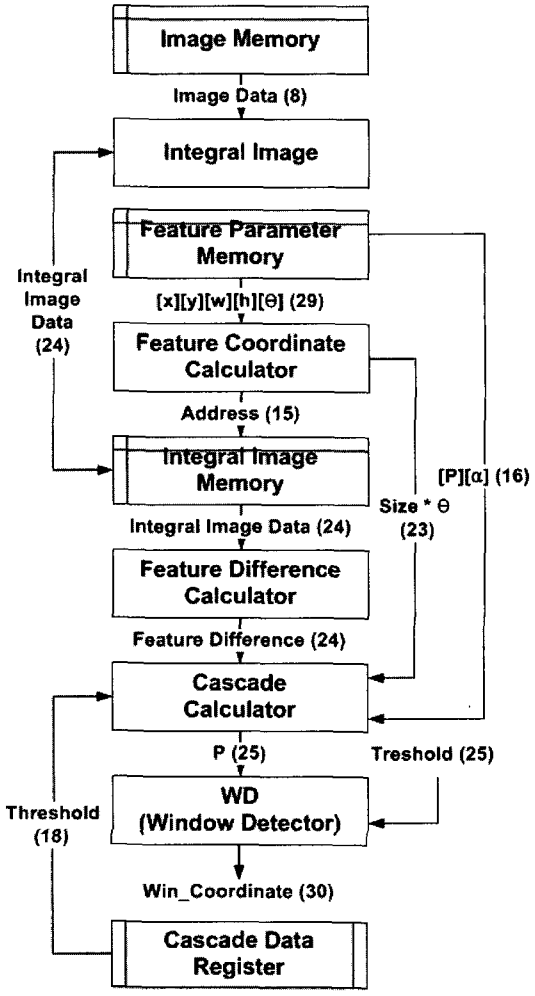


그림 9. Face Detector 전체 내부 구조
Fig. 9. Face Detector Block Diagram.

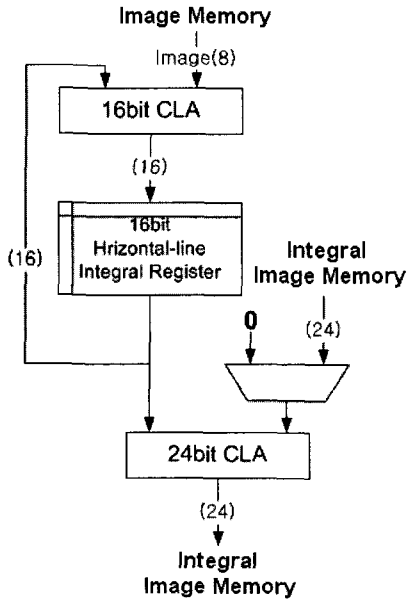


그림 10. Integral Image Calculator 블록 내부 구조
Fig. 10. Integral Image Calculator Block Diagram.

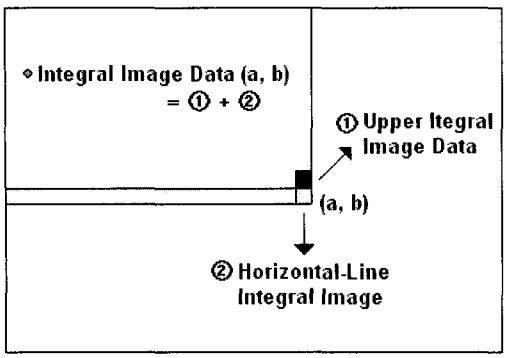


그림 11. 하드웨어의 인테그랄 이미지 연산
Fig. 11. Integral Image Calculation of Hardware.

구조를 나타내었다. 연산 방법은 그림 11과 같이 이미지 메모리로부터 들어온 8-비트 이미지 데이터를 먼저 가로줄로 합산하여, 다시 인테그랄 이미지 메모리로부터 바로 위쪽 픽셀의 값을 읽어 합하여 출력한다. 영상에서 외곽 라인의 경우 인접 픽셀의 값이 없으므로 MUX를 이용하여 0(Zero)을 입력한다.

2. Feature Coordinate Calculator

그림 12에 설계한 Feature Coordinate Calculator 블록의 구조를 나타내었으며, 각 특징 형태 별로 필요한

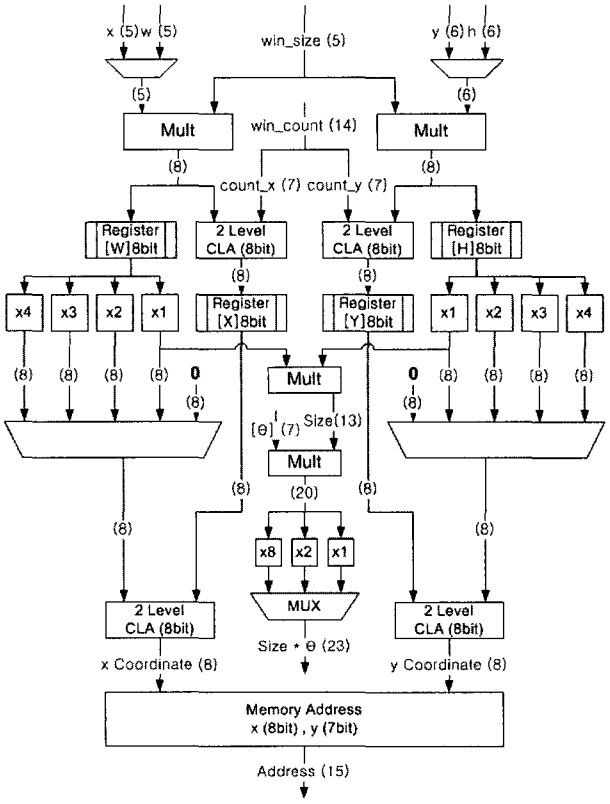


그림 12. Feature Coordinate Calculator 블록 내부 구조
Fig. 12. Feature Coordinate Calculator Block Diagram.

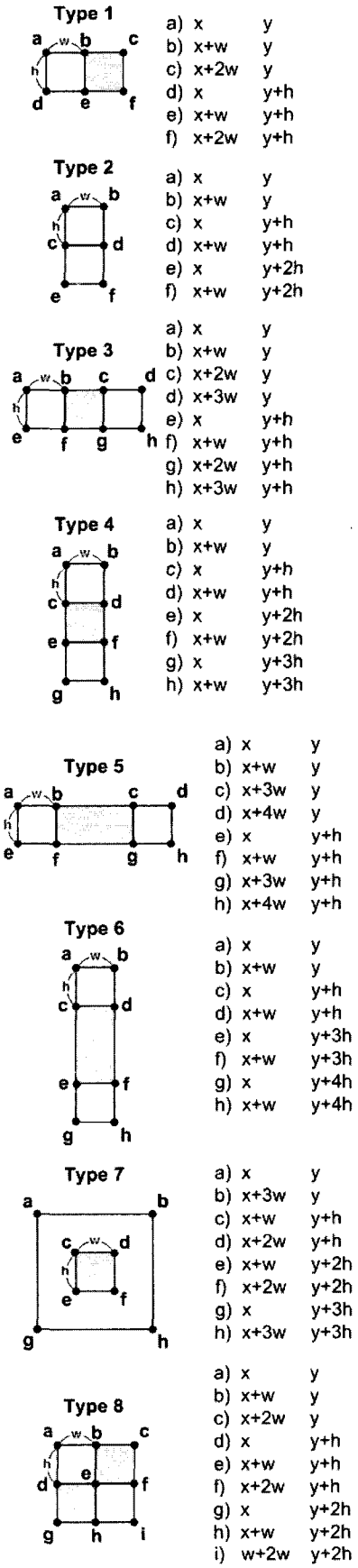


그림 13. 특징(feature) 형태별 좌표 연산
Fig. 13. Coordinate Calculation of Feature.

좌표를 연산하는 방법을 그림 13에 정리하였다. 그림 13에서 보는 것과 같이 특징의 좌표를 연산하기 위해서는 기준 점 a의 좌표(x, y)와 특징의 단위 너비(w)와 높이(h)의 1~4배의 값이 필요 하다.

특징 파라미터로부터 특징의 위치(x, y)와 크기(w, h)를 입력받아 윈도우의 크기(win_size)와 곱하여, 해당 윈도우의 크기에 맞는 좌표(X, Y)와 크기(W, H)를 계산하고, 다시 윈도우의 위치 좌표(count_x, count_y)와 더하여 기준 좌표를 저장한다. 그리고 MUX를 이용하여 형태별 좌표 연산의 조합을 만들고, CLA를 이용하여 특징의 기준좌표와 더하여 주소를 출력한다.

캐스케이드 연산 단계에서 특징 차이 값의 평균을 구하는 연산(나눗셈)을 없애기 위해 미리 특징의 가로, 세로 크기(W, H)와 기준 문턱값을 곱하여 출력한다. 하드웨어 구조에서는 나눗셈기의 연산이 곱셈의 연산보다 구조가 복잡하고, 더 많은 시간을 소모하기 때문이다.

그리고 특징의 기준 좌표 X와 Y, 너비(W)와 높이(H)가 동시에 계산 될 수 있도록 병렬 처리 하였으며, 리소스를 줄이기 위해 곱셈기와 덧셈기를 공유하였다.

3. Feature Difference Calculator

그림 14에 Feature Coordinate Calculator 블록의 구조를 나타내었으며, 각 특징 형태 별 차이값 연산 식을 그림 15에 나타내었다. 그림 15에서 a~i는 각 해당 위치의 인테그랄 이미지 픽셀값이며, s는 특징의 단위 너비(w)와 높이(h)를 곱한 값이다. 해당 식은 각 특징에 나타난 어두운 부분의 평균 픽셀값과 밝은 부분의 평균

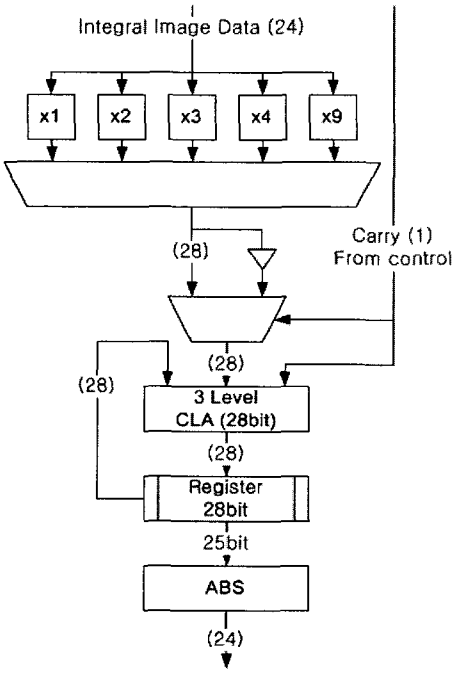


그림 14. Feature Difference Calculator 블록 내부 구조
Fig. 14. Feature Difference Calculator Block Diagram.

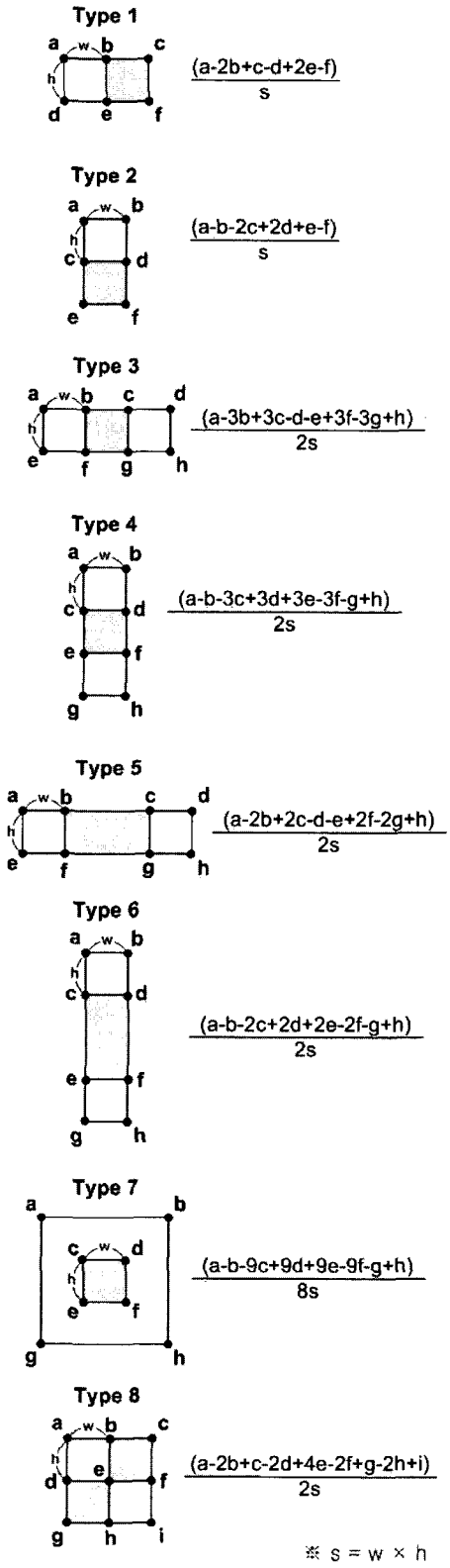


그림 15. 특징 차이 연산
Fig. 15. Feature Difference Calculation.

픽셀값의 차이를 연산하는 식이다. 여기서 s, 2s, 8s의 나눗셈 연산을 피하기 위해 2절의 Feature Coordinate Calculator 블록에서 미리 문턱값(Θ)에 s를 곱하여 출력한다.

해당 블록은 특징 형태별로 Feature Coordinate Calculator 블록에서 연산된 좌표를 이용하여 해당 좌표의 인테그랄 이미지를 읽어 들이고, 그림 15에 나타난 식의 분자 부분(특징 차이 값)을 연산하여 출력한다. 해당 식의 x2, x4, x8 등의 연산은 하위 비트에 제로(0)를 삽입하는 방법을 사용하였으며, x3, x9 등의 연산은 CLA를 사용하여 x2값과 x8값에 x1값을 더해주는 방법을 사용하였다. x1, x2, x3, x4, x9로 연산된 값은 MUX를 통하여 뺌셈과 덧셈에 필요한 값으로 선택되고, CLA를 통해 연산된다. 연산된 차이값은 레지스터에 누적되어 저장된 후에 연산이 끝나면 절대값(ABS)으로 연산되어 출력된다.

4. Cascade Calculator

그림 16에 설계한 Cascade Calculator 블록의 구조를 나타내었다. 해당 블록은 Feature Difference Calculator 블록으로부터 입력받은 차이 값과 문턱값을 특징 데이터의 비교 방향(dir : > 또는 <을 0과 1로 표현)을 이용하여 비교를 한다. 비교 결과에 따라 해당 특징의 가중치(α)값을 누적하여 저장한다. 캐스케이드 각 단계가 끝날 때 마다 해당 단계의 캐스케이드 문턱값과 비교하여, 다음 단계로 넘어갈 것인지를 출력하고, 각 단계별 가중치의 누적 값(H=Σαh)을 다시 누적하여 해당 윈도우의 확률 값(P=ΣH)을 출력한다.

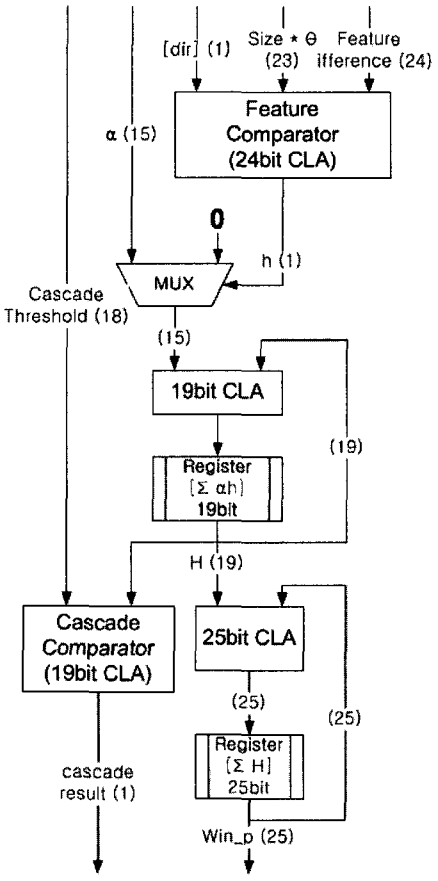


그림 16. Cascade Calculator 블록 내부 구조
Fig. 16. Cascade Calculator Block Diagram.

5. Window Detector

그림 17에 설계한 Window Detector의 구조를 나타내었다. Cascade Calculator로부터 입력된 확률 값(P)을 문턱값(Threshold)과 비교하여, 얼굴 검색 레지스터(Face Search Resister)에 ROM 테이블과 윈도우 슬라이딩 카운터 값을 이용해 연산된 윈도우 좌표정보와 함께 최대값을 갱신하여 저장하고 출력하도록 설계하였다.

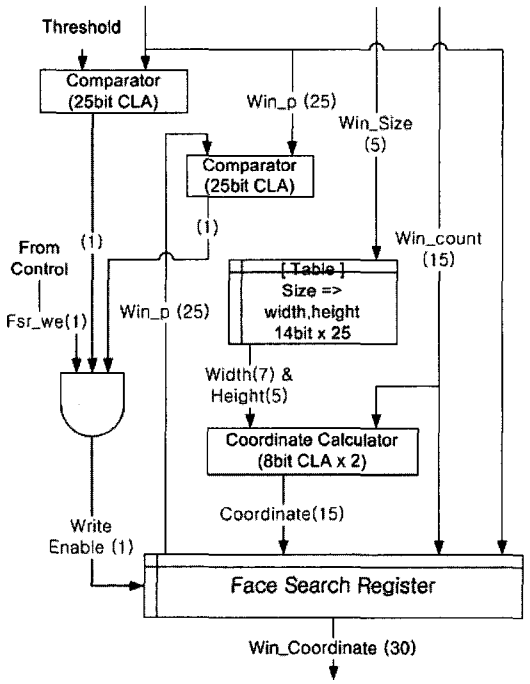


그림 17. Window Detector 블록 내부 구조
Fig. 17. Window Detectore Block Diagram.

Ⅳ. 설계 검증 및 성능 분석

본 논문에서 사용된 알고리즘은 인테그랄 이미지를 사용함으로써 하드웨어 구조로 설계할 경우 표1과 같이 입력영상의 몇 배에 달하는 메모리 사용량을 요구하며, 연산속도 또한 크게 저하되는 문제점이 있다. 해당 문제

표 1. 입력영상에 따른 메모리 사용량 및 연산 시간
Table 1. Using Memory & Processing Time by Input Image.

입력영상	메모리 사용량 (Image Memory + Integral Image Memory)	예상 연산 시간 (100Mhz 기준)
160x120	74,400 byte	0.47 초
320x240	316,800 byte	1.88 초
640x480	1,766,400 byte	7.52 초

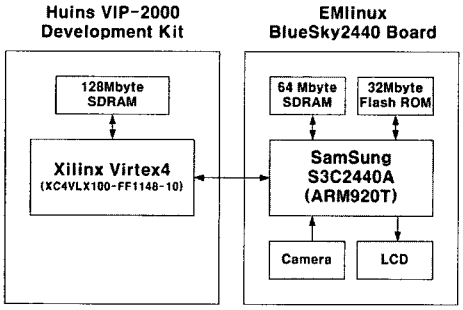


그림 18. 검증 환경 구성도
Fig. 18. Design Verification Component.

점은 입력영상 메모리와 인테그랄 이미지 메모리의 크기를 늘리고, 연산기의 병렬화를 통하여 성능을 높일 수 있지만, 성능에 비하여 하드웨어의 크기가 너무 커지게 된다. 본 논문에서는 하드웨어의 연산 시간 및 성능, 크기 등을 고려하여, 해당 하드웨어의 구조에 알맞은 160x120 크기를 입력영상으로 제시하고 해당 영상의 크기에 알맞게 메모리와 연산구조를 설계 및 구현하였다.

구현된 하드웨어는 입력 영상, 인테그랄 이미지, 특징 정보 등의 데이터를 저장하기 위해 약 88 Kbyte의 내부 SRAM(이미지 메모리 : 약 20.5Kbyte, 인테그랄 이미지 메모리 : 약 61.5Kbyte, 특징 파라메터 메모리 약 6Kbyte)을 사용한다. 이에 알맞은 검증 환경을 구성하기 위해 내부 메모리 용량을 고려하여 Xillinx사의 Virtex4 시리즈를 타겟 FPGA로 선정하였다.

이후 본 논문에서 설계한 구조와 함께 얼굴 검증 및 얼굴 인식 구조를 함께 검증하기 위해 Virtex4 시리즈 중에서 상대적으로 사이즈가 큰 LX100(1,000만 게이트 FPGA)모델을 사용할 수 있고, 외부 SDRAM을 사용할 수 있도록 구성되어있는 (주)휴인스의 VIP-2000 보드[11]를 FPGA 보드로 사용하였다. 그리고 디바이스 드라이버 및 소프트웨어 검증을 위해 삼성의 S3C2440A Embedded CPU[12]를 사용한 EMlinux사의 BlueSky2440 보드[13]를 외부 확장 핀을 이용하여 VIP-2000 보드와

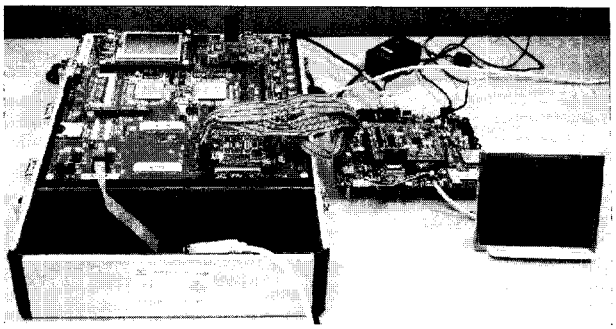


그림 19. 검증 환경
Fig. 19. Verification Environment.

표 2. PGA(Xilinx Virtex4 LX100) 합성 결과 요약
Table 2. PGA Synthesis Result Summary.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops:	1,165	98,304	1%
Number of 4 input LUTs:	3,249	98,304	3%
Logic Distribution:			
Number of occupied Slices:	2,156	49,152	4%
Number of Slices containing only related logic:	2,156	2,156	100%
Number of Slices containing unrelated logic:	0	2,156	0%
Total Number 4 input LUTs:	3,251	98,304	3%
Number used as logic:	3,249		
Number used as a route-thru:	2		
Number of bonded IOBs:	41	768	5%
Number of BUFG/BUFGCTRLs:	1	32	3%
Number used as BUFGs:	1		
Number used as BUFGCTRLs:	0		
Number of FIFO16/RAMB16s:	43	240	17%
Number used as FIFO16s:	0		
Number used as RAMB16s:	43		

연결하여 사용하였다.

VIP-2000 보드와 BlueSky2440 보드의 구성도를 그림 18에 나타내었으며, 그림 19에 검증 환경의 사진을 나타내었다.

Xilinx Virtex4 LX100(Total 98,304 LUTs & 240 x 18kbits Memory Block)을 타겟으로 Synplify Pro 8.1에서 합성한 결과 표1에서 보이는 바와 같이 3%의 LUT (3,251 LUTs)를 사용하고, 17% 가량의 메모리 공간(43 x 18kbits)을 사용하였다.

설계된 하드웨어는 매그나칩 반도체(구 하이닉스)사의 0.25um ASIC 공정을 타겟으로 Synopsys Design Compiler(2004-12 SP2)에서 합성한 결과 259,051 μm^2 (약 6.5만 Gate)의 Combinational area와 1,387,779 μm^2 (약 34.5만 Gate)의 Noncombinational area로 메모리를 포함하여 전체 1,646,885 μm^2(약 41만 Gate)을 사용하였으며, 100MHz 이상의 동작 예상 주파수가 나옴을 확인하였다.

그림 20은 본 논문에서 설계한 얼굴 검출 하드웨어를

얼굴 인식칩에 포함시켜 매그나칩 0.25um ASIC 공정을 위해 합성하여 얻은 레이아웃(Layout) 결과다. PLL 및 각각의 메모리 블록을 표시 하였으며, 얼굴 검출에서 사용된 Image Buffer Memory, Integral Image Memory, Feature Parameter Memory 등은 점선으로 묶어서 나타내었다.

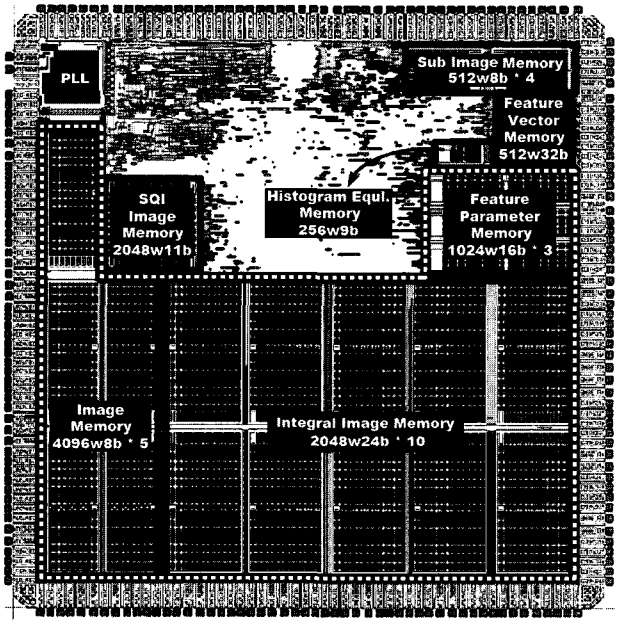


그림 20. 얼굴 인식칩 레이아웃
Fig. 20. Face Recognition Chip Layout.

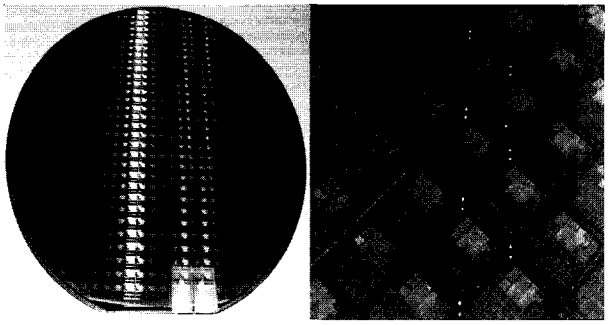


그림 21. 얼굴 인식칩 웨이퍼 사진[14]
Fig. 21. Wafer of Face Recognition Chip[14].

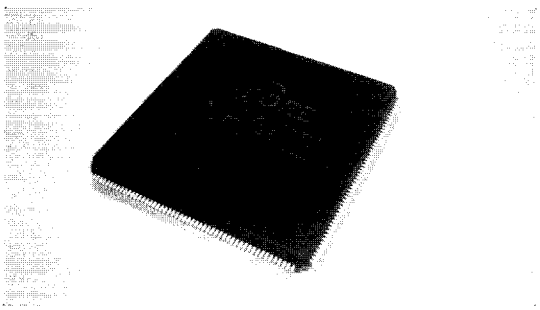


그림 22. 얼굴 인식칩 사진[14]
Fig. 22. Face Recognition Chip[14].

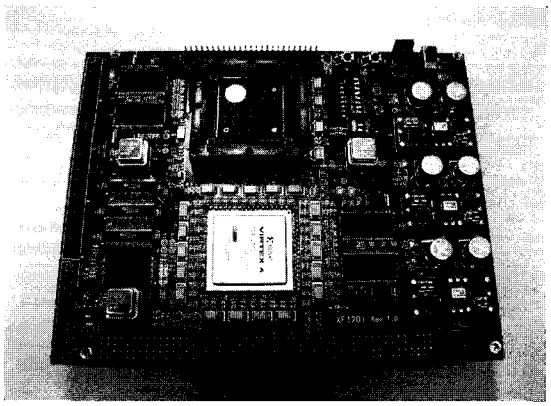


그림 23. 얼굴 인식칩 검증 보드 사진
Fig. 23. Verification Board for Face Recognition Chip.

그림 21과 그림 22에 제작된 얼굴 인식칩의 웨이퍼 사진과 QFP(Quad Flat Package) 208핀 크기로 패키징하여 완성된 얼굴 인식칩의 사진을 나타내었다[14]. 그림 23은 완성된 칩을 테스트하기위해 제작된 보드로 FPGA검증 환경과 같이 BlueSky2440 보드에 연결하여 사용할 수 있도록 구성되어 있다.

얼굴 검출의 성능은 포항공과대학교의 IMLAB에서 제공한 3,327개의 얼굴 영상[15]을 이용하여, BlueSky2440 보드에서 ARM920T 코어로 400MHz환경에서 소프트웨어로 실행한 결과와 Virtex4 FPGA에서 24MHz의 속도로 실행한 결과와 비교하여 검증하였다.

검출 성능은 제공된 영상에 정확한 얼굴 중심을 추출한 데이터 또는 얼굴 검출에 대한 기준 데이터가 포함

그림 24. 얼굴 검출에 성공한 영상의 예
Fig. 24. Example Images of Face Detection Success.

그림 25. 얼굴 검출에 실패한 영상의 예
Fig. 25. Example Images of Face Detection Failure.

표 3. 평균 실행 속도 비교 (단위 : 초)
Table 3. Face Detection Processing Time Comparison.

실행 환경 (CCD 카메라에서 입력받은 영상)		얼굴 검출 윈도우 이동 간격		
		1픽셀	2픽셀	4픽셀
소프트웨어 (ARM920T Core 400MHz)	부동소수점 모델	276.61	73.24	18.28
	고정소수점 모델 (하드웨어 모델)	4.75	1.57	0.58
하드웨어	FPGA 24MHz	1.96	0.49	0.13
	ASIC 100MHz	0.47	0.11	0.03

되어 있지 않아 얼굴 검출 윈도우의 좌표가 출력된 경우 성공으로 간주하고 측정하였다. FPGA에서 실행시킨 결과 검출된 얼굴좌표는 약간의 오차가 있지만, 부동소수점모델과 흡사한 약 99%의 검출율을 나타내었으며, 고정소수점모델과는 검출 결과가 모두 결과가 동일하였다. 그림 24와 그림 25에 얼굴 검출에 성공한 결과와 실패한 결과의 예시 영상을 나타내었다(예시 영상은 직접 CCD 카메라를 통하여 입력 받은 영상 사용).

성능검증을 위해 사용된 영상은 주로 그림 24과 같이 영상의 밝기가 적당한 수준의 영상이었다. 하지만 실제 CCD카메라를 통하여 입력받은 영상을 이용하여 조명을 다양하게 변화시켜 확인한 결과 그림 25와 같이 주로 조명이 어두운 환경에서 검출을 실패하는 경우가 많았다. 특징 데이터의 학습에 사용된 영상이 일정수준의 밝기를 가지는 영상을 기준으로 만들어 졌기 때문으로 예상되며 이러한 경우 히스토그램 평활화(Histogram Equalization)와 같은 영상 보정을 통하여 어느 정도 검출을 높일 수 있었다.

표 3에 각각의 실행 환경과 얼굴 검출 윈도우의 이동 간격에 따른 검출 속도의 결과를 나타내었다. 검출 시간은 인테그랄 이미지 연산의 시작부터 얼굴 검출 데이터가 나오기까지의 시간을 측정하였다.

위 결과에서 소프트웨어에서 부동소수점 모델과 고정소수점 모델의 연산 속도 차이가 심한 것은 ARM 920T 코어의 부동소수점 곱셈 연산이 매우 느리기 때문이다. 고정소수점 모델과 FPGA에서 24MHz에서 실행한 속도는 약 3배의 차이를 보였으며, ASIC 100MHz의 실행 속도는 고정소수점 모델과 비교할 경우 약 10배의 차이를 보였다. 이 결과로서 본 논문에서 제시한 하드웨어 구조가 임베디드 시스템의 실시간 얼굴 검출 솔루션으로서 필수적임을 알 수 있다.

VI. 결 론

본 논문에서는 입력된 영상으로부터 얼굴 후보 영역을 좌표로 출력하는 얼굴검출 알고리즘을 하드웨어로 설계하고 FPGA에서 검증하였다. 각각의 연산과정을 하드웨어의 구조에 알맞게 총 5개의 모듈과 On-Chip SRAM으로 구성하였으며, 검증 결과 임베디드 환경에서 충분한 성능이 구현될 수 있음을 확인하였다.

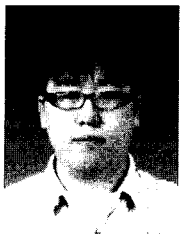
설계한 얼굴 검출 하드웨어 구조를 얼굴 검증 및 얼굴 인식 하드웨어 구조와 통합하여, 매그나칩 0.25um ASIC 공정을 거쳐 제작된 XF1201 칩의 핵심 모듈로 구현하였으며, 정상적으로 동작함이 확인되었다. 현재 보다 높은 인식률 및 성능 향상을 위하여 변경 가능한 학습 데이터를 수정 및 보완 중에 있다. 제작된 칩의 빠른 얼굴 검출 속도를 바탕으로 보안 통제 분야 및 다양한 응용 분야에서 다용도로 활용이 가능할 것으로 예측된다.

참 고 문 헌

[1] Paul Viola, and Michael Jones, "Robust Real-time Object Detection," Second International

Workship on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sam pling. Vancouver, Canada, July 13, 2001.
[2] 이호근, 정성태, "실시간 얼굴 검출 시스템 설계 및 구현," Journal 0f Korea Multimedia Society, Vol.8, No.8, pp.1057-1068, August 2005.
[3] 이영학, 심재창, "표면 곡률을 이용하여 깊이 가중치 Hausdorff 거리를 적용한 3차원 얼굴 영상 인식," Journal 0f Korea Multimedia Society, Vol.8, No.1, pp.34-45, January 2005.
[4] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, Anil K. Jain, "Face Detection in Color Images," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.24, No.5, pp.696-706, May 2002.
[5] 조한수, "분포맵에 기반한 얼굴 영역 검출," Juornal of Korea Multimedia Society, Vol.9, No.1, pp.11- 22, January 2006.
[6] 권혁봉, 권동진, 장언동, 윤영복, 안재형, "YCbCr 색공간에서 피부색과 윤곽선 정보를 이용한 얼굴 영역 검출," Journal of Korea Multimedia Society Vol.7 No.1, pp.27-34, January 2004.
[7] C. Papageorgiou, M. Oren, and T.Poggio. "A general framework for Object Detection," In International Conference on Computer Vision, 1998.
[8] Paul Viola, and Michael Jones, "Rapid object detection using a boosted cascade of simple features," Computer Vision and Pattern Recognition 2001(CVPR 2001), Vol.1, pp.511-518, Dec. 2001.
[9] Lienhart, R and Maydt, J, "An extended set of Haar-like features for rapid object detection," International Conference on Image Processing 2002(ICIP2001), Vol. 1, pp.900-903, Sept. 2002.
[10] 김영진, 정성윤, 정용진, "SoC 하드웨어 설계를 위한 얼굴 인식 알고리즘의 고정 소수점 모델 구현 및 성능 분석," SoC 학술대회, 대학전자공학회, May, 2006.
[11] Verification IP AMBA Platform User's Guide ver 1.5, (c)Huins, August 23, 2005.
[12] S3C2440A 32-Bit CMOS Microcontroller User's Manual Revision 1, (c)Samsung Electronics, 2004.
[13] Bluesky2410/2440 Reference Manual V1.0, (c)EMlinux, 2004.
[14] XF1201 Data Sheet Rev.01, (c)Core Information System, July 2006.
[15] POSTECH. Intelligent Multimedia Laboratory, "Face Image Database", http://nova.postech.ac.kr

저 자 소 개



이 수 현(학생회원)

2005년 광운대학교 전자공학부 학사 졸업.
2007년 광운대학교 전자통신 공학과 석사 졸업.
2007년 3월~현재 광운대학교 전자통신공학과 박사과정 재학.

<주관심분야 : SoC 설계, 영상처리 및 인식, 임베디드 시스템 설계>

정 용 진(정회원)

1983년 서울대학교 제어계측 공학과 학사 졸업.
1983년 3월~1989년 8월 한국전자통신연구원.
1995년 미국 UMASS 전자전산 공학과 박사 졸업.
1995년 4월~1999년 2월 삼성전자 반도체 수석 연구원.
1999년 3월 광운대학교 전자통신공학과 부교수

<주관심분야 : 무선통신, 정보보호, SoC 설계, 영상처리 및 인식, 임베디드 시스템>