CMOS 이미지 센서용 효과적인 인터폴레이션 구현

이동훈, 손승일
한신대학교 정보통신학과

Implementation of an Efficient Interpolation for CMOS Image Sensor

Dong Hun Lee, Seung Il Sonh
Dept. of Information and Communication Hanshin University
e-mail : asickorea@hs.ac.kr

요 약

본 논문에서는 영상 입력 장치 또는 카메라 이미지 센서로부터 얻은 Bayer Data입력 포맷을 우리가 디스플레이 장치로 보는 영상으로 출력하기 위해 전처리 작업을 수행한다. 먼저 들어오는 Bayer Data Format은 인터폴레이션을 수행하여 컬러영상을 표현하기위한 한 픽셀 표현 R, G, B값을 구한다. 본 논문에서는 연산량과 필요한 레지스터의 수를 줄이고 칩의 성능을 향상시키기위해 기존 3x3 라인 쓰지 않고 2x2라인을 이용한 인터폴레이션을 수행한다. 또한 Bayer Data입력에 대한 이미지 스케일링 작업과 인터폴레이션 수행 작업을 동시에 수행한다. 이를 구현하기위해 원본 이미지 사이즈를 640x480으로 입력 데이터를 사용하고, 소프트웨어로 전처리하여 이미지 결과를 확인한 후, 최적화된 알고리즘를 적용하여 VHDL설계언어를 이용한 하드웨어 설계후, ModelSim 6.0a를 이용하여 데이터를 검증한다.

1. 서 론

요즘 카메라폰의 사용자는 복합된 기능과 고화질, 소형화, 저비용, 저전력등을 요구하는 추세이며, 이제 경쟁력있는 제품을 기업들은 단기간에 생산하여 소비자 욕구를 충족시키고자 한다. CMOS 이미지 센서는 칩 공정과정에서 공정비용이 저렴하며, 대량생산이 가능하고 다양한 기능블록 모듈을 하나로 통합하여 SoC (System on Chip)화된 제품으로 만들 수 있는 장점을 갖고 있다. 그러나 CMOS 이미지 센서는 영상의 고화질에 대한 개선 문제점을 안고 있다. 이처럼 고화질 문제를 기본렌즈의 개선 문제와 카메라 센서의 화소(픽셀) 사이즈 확대 뿐만아니라 ISP(Image Singal Processing)처리부에서 효과적으로 영상제품에 맞는 유형으로 처리하여 화질 개선 문제를 해결해야 한다. ISP처리에서는 다양한 기능을 처리하는데 그중에 감마정정, 컬러 인터폴레이션, 컬러정정, 컬러 공간적 변환, 이미지 스케일(확대/축소), Image Effect(세피아, 그레이, 반전)처리등 다양한 전처리 기능을 수행해야 한다[1][2]. 본 논문에서는 영상 입력 장치 또는 카메라 렌즈를 통해 받아들인 Bayer Data를 통해 이미지 컬러 인터폴레이션을 수행하고 효과적으로 이미지 스케일링을 위해 Bayer Data와 인터폴레이션을 동시해 수행하여 설계하였다. 기존 인터폴레이션에서는 3x3라인을 적용하여 설계를 진행하였다. 그러나 2x2라인을 적용하여 설계 하였을 때 라인버퍼의 수도 줄고, 필요한 레지스터 수도 감소하는 장점을 가져 본 논문에서는 2x2라인 설계 하였다. 두가지 알고리즘에 대한 소프트웨어 검증에서 이미지의 차이가 거의 없고, 효과적인면에서 2x2라인 인터폴레이션이 우수하고 칩사이즈를 줄일 수 있었다. 또한 이미지 스케일링을 위해 앞단에서 얻은 Bayer 데이터 포맷 이미지를 1/2으로 줄이위한 4개의 픽셀데이터에 대해서 하나의 픽셀로 처리하였고 픽셀 R, G, B값으로 인터폴레이션을 동시에 수행하는 효과적인 처리 결과를 얻었다. 이와 마찬가지로 이미지를 1/4로 줄이기위해 16개의 픽셀 데이터를 하나의 픽셀로 처

리하고 R, G, B로 동시에 인터폴레이션이 수행하도록 설계하였다. 소프트웨어 검증을 통해 이를 최적화된 알고리즘으로 VHDL하드웨어 설계언어를 이용하여 코딩한 후 ModelSim 6.0a를 이용하여 파형을 검증하고 원본 이미지와 대상이미지에 대한 PSNR값을 비교하여 설계에 대한 결론을 내리고자 한다[1][2].

2. 이미지 센서의 전처리(PreProcessing) 개요

2-1. Bayer 포맷

다음 그림 1은 ISP(Image Signal Processing) 처리 블록으로 들어오는 Bayer포맷의 형태를 보여주고 있다. Bayer 포맷은 두 개의 G컴포넌트와 각각 하나의 R, B컴포넌트로 이루어져 있다. 또한 짝수라인에서는 연속적인 G, B컴포넌트로 들어오고, 홀수 라인에서는 R, G컴포넌트로 라인별로 연속적인 값으로 이루어져 있는게 특징이다. 반드시 각 라인에서 시작하는 컴포넌트는 그림과 같지 않으며 R, G, B의 시작위치는 초기 레지스터 설정값에 따라 바뀐다.

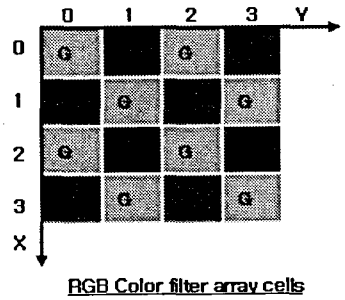

그림 1. Bayer Format 필터 어레이

또한 Bayer 데이터의 기본 R, G, B색상 어레이를 가지고는 우리가 눈으로 식별할 수 있는 이미지를 도출 할 수 없다. 그러므로 ISP기능을 수행 해야 한다[1][2][3].

2-2. ISP 처리 블록도

영상 입력으로 들어오는 Bayer 데이터 입력을 받아서 기본적인 ISP처리 기능을 수행하는데 감마정정, 컬러 인터폴레이션, 컬러정정, 컬러 공간적 변환, 화이트 밸런스, AE(Auto Exposure), BLC(Black Level Correction)등과 같은 기능을 사람의 눈은 완벽하게 수행 할 수 있지만, 카메라 렌즈는 ISP처리를 해야만 사람이 볼 수 있는 이미지가 된다.

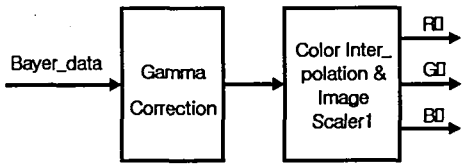

그림 2. ISP전처리 과정 블록도

그림 2에서는 Bayer 데이터 입력으로부터 감마정정을 수행하고 다음 데이터 입력을 받아 본 논문에서 제시한 알고리즘을 적용하여 이미지 사이즈 640x480 픽셀 어레이에 대한 2x2라인 인터폴레이션과 이미지 스케일링 1/4, 1/16 픽셀 축소를 함께 수행하여 설계를 진행 하였다.

2-3. R, G, B 인터폴레이션

입력 Bayer 데이터를 입력받아 한 픽셀을 구성하기위해 주위의 인접한 픽셀로 부터 색을 보간하며 각 R, G, B에 대한 픽셀을 구하여 수행하는 기능이 R, G, B 인터폴레이션이다. 정해진 픽셀 어레이에서 각 라인별 보간을 수행한다. 두 라인을 비교하여 수행하는데 이전 라인과 현재 라인을 가지고 하나의 라인을 구한다. 그러나 첫 번째 라인에 대한 보간은 예외처리적으로 이전 라인 데이터가 존재하지 않으므로 현재 라인 데이터만 가지고 보간을 수행한다. 그리고 두 번째 라인부터는 이전 라인이 계속적으로 존재하므로 반복적으로 2x2라인을 이용하여 각 라인에 대한 보간을 수행 할 수 있다. 다음 그림 3의 픽셀 어레이에서 표-1과 같이 R, G, B값을 구하여 처리한다[2][3].

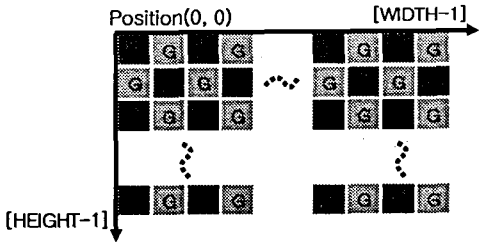

그림 3. 2x2라인 인터폴레이션

표-1. 각 픽셀 위치별 R, G, B 계산식

(홀수행, 짝수열)	(홀수행, 홀수열)
R[1,0] = R(0,0) G[1,0] = [G(0,1) + G(1,0)]/2 B[1,0] = B(1,1)	R[1,1] = R(0,0) G[1,1] = [G(0,1) + G(1,0) G(1,2)]/3 B[1,1] = B(1,1)
(짝수행, 짝수열)	**(짝수행, 홀수열)**
R[2,0] = R(2,0) G[2,0] = [G(1,0) + G(2,1)]/2 B[2,0] = B(1,1)	R[2,1] = [R(2,0) + R(2,2)]/2 G[2,1] = G(2,1) B[2,1] = B(1,1)

2-4. 이미지 스케일링

이미지 스케일링을 위해 다음 그림 4와 같이 2x2 픽셀 어레이 사이즈에서 식 1)~3)와 같이 각 R, G, B에 대한 이미지 스케일과 인터폴레이션을 동시에 수행하면 효과적인 처리를 할 수 있다.

R = R(0,0) ---------- 식 1)

G = G(0,1) + G(1,0) ----- 식 2)

B = B(1,1) ---------- 식 3)

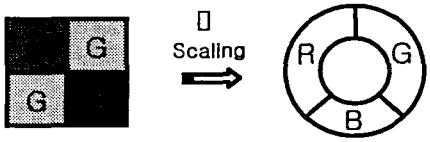

그림 4. 1/4 스케일링에서 구한 R, G, B픽셀

3. 인터폴레이션과 이미지 스케일링 블록의 설계

3-1. 2x2라인 인터폴레이션 블록도

제안한 블록 그림 5에서는 2x2라인 인접한 픽셀에 대한 보간을 수행하기 위해 이전 라인 데이터와 현재 라인 데이터를 보간에 사용하고 이전 데이터에 대한 버퍼링을 고려해서 라인버퍼 1개와 최소 지연 타이밍에 필요한 4개의 레지스터를 두고 각 라이별 처리를 수행한다. 행렬 라인에 대한 처리는 짝수행(짝수, 홀수열)과 홀수행(짝수, 홀수열)별로 처리하고 이를 위해 행 카운트와 열 카운트를 사용한다. 그리고 각 행렬 연산을 수행한다. 그림 5에서는 2x2라인 인터폴레이션 수행을 위한 세부 연산 블록도이고 연산에 필요한 라인버퍼 1개, 4개의 레지스터, 행렬 카운트 MUX, 레지스터 선택 MUX, 나눗셈 수행을 위한 오른쪽 쉬프트, 덧셈기 등으로 구성되어 있다[4][5].

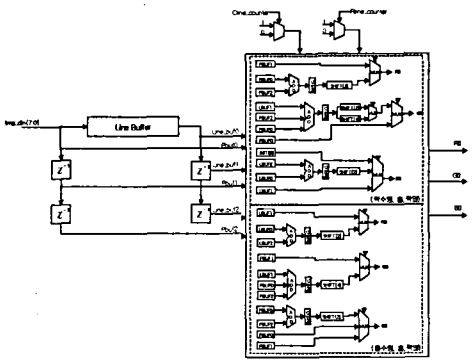

그림 5. 2x2라인 인터폴레이션 제안 블록도

3-2. 이미지 스케일링 처리 블록도

제안한 블록도는 입력 Bayer 데이터에 대해 1/4, 1/16 이미지 스케일링을 수행하는 처리 블록도이다. Sel_Mode 신호가 '1'값일때 정상모드(2x2 라인) 인터폴레이션을 수행하고 Sel_Mode 신호가 '0'값일때 이미지 스케일링이 활성화 된다. Mode 선택신호가 '1'값일때 1/4처리를 수행하고 '0'일때 1/16로 처리하기 위한 선택모드이다. 1/4이미지 스케일 처리는 2개의 라인에 대해서 4개의 픽셀 묶음이 하나의 픽셀값 R, G, B로 사용하여 처리하므로 동시에 인터폴레이션과 이미지 스케일을 처리하는 효과적인 결과로 처리 한다. 이전에 사용한 2개의 라인은 오버랩하여 쓰지않고 다시 새로운 2라인을 읽어 처리 수행 한다. 첫 연산 결과

는 2번째 라인 2번째 열까지 지연 클럭을 가지고 그 이후 클럭부터 연속적인 데이터 처리를 수행한다. 그림 6은 제안한 1/4, 1/16처리 블록도이고 이를 수행하기 위해 라인버퍼 1개, 타이밍에 필요한 2개 레지스터, 각 픽셀 R, G, B에 대한 중간 결과를 저장하기 위한 기존 라인버퍼 보다 1/4사이즈 줄어든 라인버퍼 컴포넌트 3개, 나눗셈 연산을 위한 오른쪽 쉬프트, 행렬 라인 카운트 MUX로 구성되어 있다.

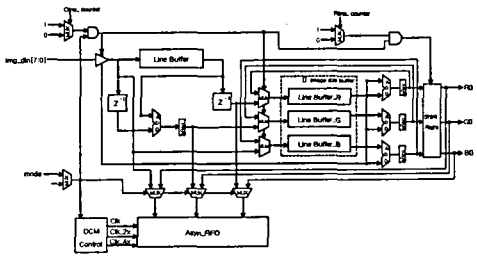

그림 6. 제안한 이미지 스케일링 블록도

또한 Mode가 '0'일때 1/16이미지 스케일 수행하여 16개의 픽셀 묶음이 하나의 픽셀값 R, G, B로 사용하고 1/4와 마찬가지로 효과적으로 두가지 처리를 수행하는 결과를 갖는다. 1/16 이미지 축소 처리는 4개의 라인을 사용하고 2개의 라인 처리는 1/4 이미지 축소 처리와 동일하게 처리하고 3번째, 4번째 라인에 대해서는 실시간 처리를 위해 두 번째 결과를 읽고, 3번째 입력이 들어오면 연산을 수행하고 이를 1/4사이즈의 라인버퍼에 담아두고, 4번째 라인에 데이터가 입력되면 라인버퍼에 3번째까지 처리된 결과를 읽어 마지막 라인에 대한 결과를 처리하고 최종 오른쪽 쉬프트를 처리한 후 실시간 데이터 처리가 가능하게 설계하였다[4][5].

4. 파형 분석

다음 파형 분석에 필요한 이미지 정보는 원본 이미지 사이즈 (640x480)이고, 입력 데이터는 Bayer 데이터 이다. 그림 7은 본 논문에서 제안한 2x2 라인 인터폴레이션 수행한 결과이다. 입력 데이터로부터 출력 데이터는 2클럭 이후 연속적으로 수행한 결과를 도출하였다.

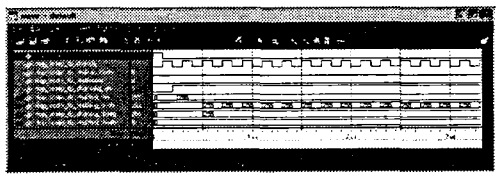

그림 7. 2x2라인 인터폴레이션 파형 시뮬레이션

다음 그림 8, 그림 9는 이미지 스케일 수행 파형 결과이다.

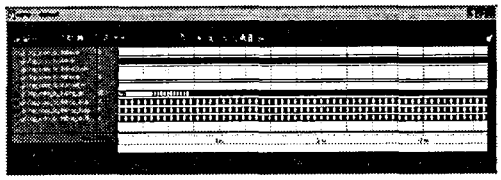

그림 8. 1/4 이미지 스케일링 파형 시뮬레이션

그림 8에서는 1/2 축소 이미지 처리로 출력 데이터는 두 번째 라인 2클럭 이후 연속적으로 이미지 결과를 얻었다. 그림 9에서는 1/4 축소 이미지 처리로 입력 데이터로부터 출력 데이터는 4번째 라인 4클럭 이후부터 연속적인 데이터 값을 도출하였다.

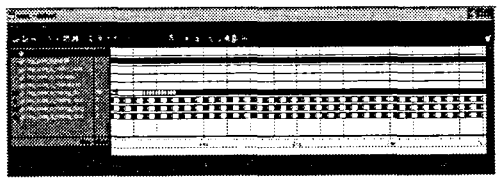

그림 9. 1/16 이미지 스케일링 파형 시뮬레이션

5. 설계 결과

본 논문의 수행 결과를 테스트 하기 위해 그림 10과 같은 두개의 원본 이미지와 Bayer 데이터로부터 입력 받아서 감마정정, 인터폴레이션을 수행할 때 감마정정과 인터폴레이션 처리 위치를 바꾸어서 처리 결과를 테스트 하였고, 인터폴레이션 라인버퍼의 수를 3x3, 2x2 라인을 적용하여 테스트 한 결과 그림 11과 같은 PSNR결과를 도출하였다. 그림 10의 첫 번째 원본이미지의 PSNR값은 그림 11에서 넘버값이 작을때 더 좋은 PSNR값을 가졌고 그리고 그림 10의 두 번째 원본 이미

지를 처리 하였을때는 각 픽셀값의 PSNR값이 동일하게 도출되었다. 결과적으로 실험 이미지에서 원본이미지와 대상 이미지와의 차이가 거의 없는 영상을 얻을 수 있었다[4][5].

그림 10. 원본 영상

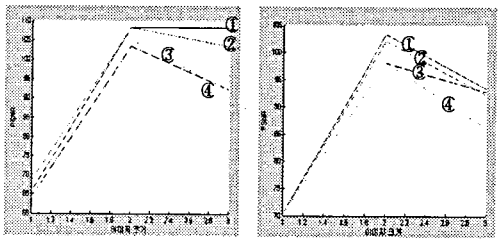

그림 11. 영상 이미지 PSNR 차트

제안한 블록의 각 등가게이트수와 타이밍 결과는 표-2 회로합성 결과표와 같이 나타났다.

표-2 회로 합성 결과

Block	No. of Gates	Timing	Target Device
2x2 Interpolaton Block	35,928	Mininum Period : 10.184ns Max. Frequency : 98.193Mhz	XCV1000e -hq240
Image Scaler Block	85,856	Mininum Period : 10.928ns Max. Frequency : 91.508Mhz	

6. 결론

카메라 이미지 센서용, 영상 입력 장치로부터 얻은 Bayer 데이터를 ISP 전처리 과정에서 앞단의 수행 처리가 하드웨어 설계시 연산량이 적게 처리되고 또한 이미지 스케일링과 인터폴레이션을 동시에 처리할 수 있는 효과적인 알고리즘 처리로 하드웨어 설계 효율을 높였고 이미지 결과에서도 좋은 영상 이미지를 도출 할 수 있었다. 결과적으로 카메라 이미지 센서용 ISP(Image Singal Processing)처리 적용 칩에서 제안한 알고리즘이 효과적으로 처리 할 수 있을것으로 사료된다[1][2].

7. 참고문헌

[1]. Yun Ho Jung, Jae Seok Kim, Bong Soo Hur and Moon Gi Kang, Department of Electronic Engineering Yonsei University, Seoul, Korea, "Design of Real-Time Image Enhancement Preprocessor for CMOS Image Sensor", IEEE Trans. Consumer Electronics, Vol. 46, No 1, February 2000.

[2]. Bongjun Lee, Jaeseok Kim and Chulhee Lee, Dept. Electrical and Computer Engineering, Yonsei University, 134 Shin chon-Dong, Seodaemoon-Gu, Seoul 120-749, Korea, "High Quality Image Interpolation for Color Filter Arrays" IEEE Trans. 2000.

[3]. Marc J. Loinaz, Member, IEEE, Kanwar Jit Singh, Member, IEEE, Andrew J. Blankby, Student Member, IEEE, David A. Inglis, Kamran Azadct, Member, IEEE and Bryan D. Ackland, Fellow IEEE. "A 200-mW, 3.3V, CMOS Color Camera IC Producing 352x288 24-b Video at 30 Frames/s. IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 33, No. 12, December 1998.

[4]. Randy Crane, Hewlett-Packard Company. "A Simplified Approach to Image Processing". Prentice Hall PTR

[5]. Rafael C. Gonalez, Richard E. Woods. "Digital Image Processing". Addison Wesley.