Verilog HDL을 이용한 SDTV용 8bit 색상 보정기의 설계

*전병웅, 송인채
숭실대학교 전자공학과
e-mail : hamm6@naver.com, isong@ssu.ac.kr

Design of an 8-bit Color Adjustor for SDTV Using Verilog HDL

*Byoung-Woong Jeon, Inchae Song
School of Electronic Engineering, Soongsil University

Abstract

In this paper, we designed an 8-bit color adjustor for SDTV using Verilog HDL. The conversion block requires a lot of multiplication. So we adopted Booth algorithm to reduce amount of operation and processing time. To improve speed, we designed the system output as parallel structure. We synthesized the designed system using Xilinx ISE and verified the operation through simulation using Modelsim.

I. 서론

다양한 서비스를 제공하는 고성능 SDTV에는 컬러 보정시스템이 필수적이다. RGB의 색 보정을 하기 위해선 Conversion block에서 YCbCr 신호로의 변환 후 보정을 해야 한다. YCbCr로 인코딩 한 후 Adjustment 블록을 통한 보정 후 디스플레이 패널 입력신호로 만들기 위해 다시 신호를 RGB 신호로 디코딩해야 한다. 이와 같은 시스템은 VIPER와 같은 on-chip video decoder에서 널리 쓰이고 있다.

Ⅱ. 구조 및 Algorithm

신호를 보정하기 위해선 영상 압축 Algorithm을 통하여 시스템을 설계 할 수 있다. 본 논문에서는 기존의 serial 방식의 data out을 parallel 방식의 설계로 제시했고, 연산 처리 시간과 밀접한 관계를 가지고 있는 곱셈기를 Booth algorithm을 사용하여 곱셈 연산을 줄임으로써 전체 연산 처리 속도의 향상을 가져왔다.
다음은 RGBtoYCbCr Block에서 사용된 알고리즘이다.

- Y = 0.299*R + 0.587*G + 0.114*B
  4Dh*R + 96h*G + 1D*B
- Cb = - 0.172*R - 0.339*G + 0.511*B + 128
  - 2Ch*R - 57h*G + 83h*B + 128
- Cr = 0.511*R - 0.428*G - 0.083*B + 128
  83h*R - 6Eh*G - 15h*B + 128

(명암도 = 0.299R + 0.587G + 0.114B 명도에 대한 NTSC 표준)

위에서 보는 바와 같이 초기 RGB 값들은 계수 값과 곱해져서 각각의 Y,Cb,Cr 신호로 변환 된다. 위의 식을 일반화하여 표현하면 다음 수식과 같다.

$$Z = aR + bG + cB + d$$

여기서 a, b, c, d는 register value이며 이들의 합은 1을 넘지 않는다. YCbCr data 값은 이 연산기를 사용하여 계산한다.

2-1. Color conversion calculator block

그림 1에 보여 지는 color conversion calculator

block은 Booth Block(BB)으로 구현한다. 동일한 module을 이용하여 설계 multiplier와 adder가 pipeline으로 구성되어 있다. input enable을 pipeline에 동기시켜서 output enable을 출력 해야 한다. 따라서 input enable signal도 Flip Flop으로 latch시켜 출력한다. 출력은 상위에서 16bit를 출력한다. 연산된 값은 유효자리 이하는 반올림된 뒤 register에 latch되어 출력된다. 초기 R,G,B는 모두 8bit unsigned integer이고, a,b,c,d coefficient는 모두 8bit signed fixed point value가 된다. aR의 결과는 16bit가 되고, 각각의 register에 저장되어 add 연산을 하게 된다. 하지만 coefficient의 합이 1보다 크지 않으므로 bit의 증가는 없게 된다. coefficient(a,b,c,d)는 모두 register에 저장되게 된다. 따라서 총 12개의 register가 필요하며, YCbCr의 최대값과 최소값을 설정하여 그 범위를 벗어날 경우에 그 값을 saturation값으로 제한한다. 각 register의 값은 항상 출력하여 color conversion calculator block으로 전달되고, address에 의해 muxing된 값은 read data port로 출력된다. 만일 선택된 register가 없으면 read data port로는 0을 출력하게 된다. register를 2조 내장하여 register값을 아무 때나 변경하고 한 frame이 완료되어야 변경된 data값을 실제 연산에 반영할 수 있도록 한다. 이 기능을 위해 updata라는 신호를 사용하며 이 신호의 rising edge다음 clock에서 register를 update한다.

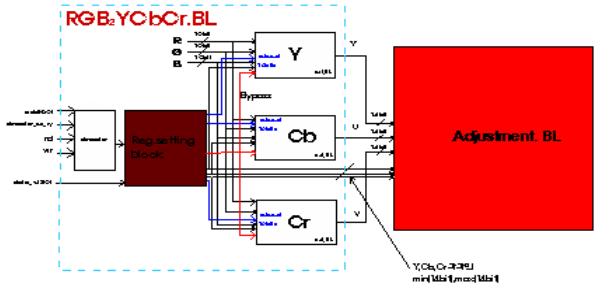

그림 1. RGB to YCbCr conversion block

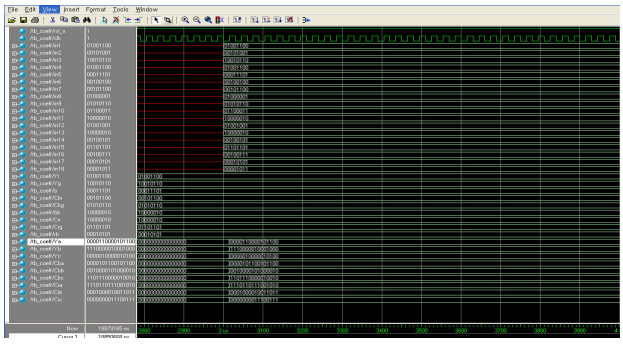

그림2. Parallel out data

그림 2는 한 block에서 data가 병렬 출력되는 것을 보여주고 있다. 이러한 병렬 출력들이 9개의 모듈로부터 한 클럭에서 동시 출력된다.

2-2. YCbCr → RGB algorithm 및 dataflow

- R = Y+1.371*(Cr - 128)
- G = Y-0.698*(Cr - 128)-0.336*(Cb - 128)
- B = Y+1.732*(Cb - 128)

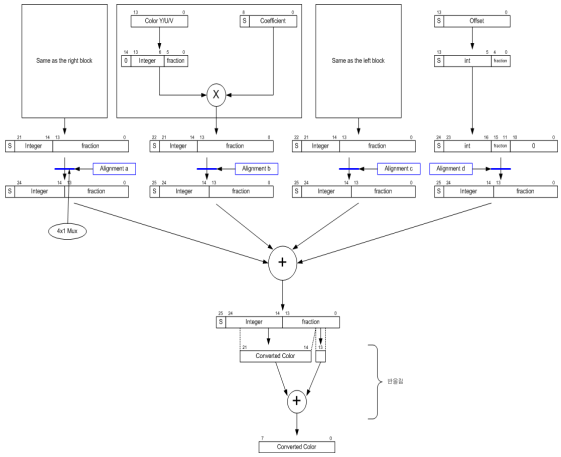

그림3. YCbCr to RGB dataflow

위 식은 변형 및 보정된 Y,Cb,Cr값들이 계수와 곱해져서 top block에서 출력하고자 하는 R,G,B 값들을 수식으로 보여주고 있다. 그림 3은 각각의 Y,Cb,Cr입력값들의 data flow를 보여주는 그림이다. 역시 YCbCrtoRGB blcok의 수식도 일반식으로 변환하면, Z = aY+ bCb + cCr + d 로 표현할 수 있다. 입력 YCbCr은 14bit unsigned 값으로 8bit의 integer와 6bit의 fraction으로 구성되어 있다. 출력 RGB는 14bit으로 YCbCr과 마찬가지로 구성되어 있다. 계수 a, b, c는 9bit으로 signed fixed point number이다. 그리고 Coefficient register와 Offset register의 최상위 bit은 coefficient와 offset의 point를 선택할 수 있도록 하는 align값을 가진다. Offset d는 14bit signed 값으로 8bit의 integer와 5bit의 fraction으로 구성되어 있다. 또한 Coefficient와 마찬가지로 point를 이동시킬 수 있다. 연산과정을 살펴보면 곱셈에서 생성되는 결과는 sign bit까지 총 22bit이 된다. 이 곱셈 결과들은 더하기 전에 point를 align해주어야 한다. Align 정보는 coefficient register와 offset register의 최상위 bit에 저장되어 있다. 이 정보를 이용하여 곱셈결과를 상위로 shift시키면 최대 25bit까지 확장된다. 확장은 4가지 case에 대한 값을 생성하고 mux에 의해서 선택되어진다. Point가 하위에 위치하면 상위는 sign bit을 반복하게 되고, point가 상위에 위치하면 하위에 0이 추가 된

다. 소수점이 align된 4개의 값들은 모두 더한다. Sign 연산을 수행하므로 출력 결과는 256보다 작게 된다. 따라서 정수부는 8bit를 초과하는 상위는 버린다. Fraction 부분은 상위 14bit만 취하고 하위는 버리는데 이때 반올림을 하게 된다.

2-2-1. Coefficient example

RGB → YCbCr(SDTV)

- R = Y + 1.371*(Cr - 128)
- G = Y - 0.698*(Cr - 128) - 0.336*(Cb - 128)
- B = Y + 1.732*(Cb - 128)

Address	Register	Decimal	Hex	PP	Register Value
0	Red a	1	200	1	4200
2	Red b	1.371	2BD	1	42BD
4	Red c	0	0	0	0
6	Red d	-175.488	-15F0	0	2A10
8	Green a	1	200	1	4200
10	Green b	-0.698	-2CB	0	0D34
12	Green c	-0.336	-158	0	0EA8
14	Green d	132.352	108B	0	108B
16	Blue a	1	200	1	4200
18	Blue b	1.732	377	1	4377
20	Blue c	0	0	0	0
22	Blue d	-221.696	-7094	0	244A

표.1 Coefficient example

표1은 RGB 신호가 binary 1111_1111로 saturation 됐을 때 coefficient 을 나타낸 표이다.

2-2-2. Min/Max Limitation

RGB의 상한과 하한의 한계를 설정한다. 기본적으로 RGB는 0 ~ 255까지이며 이 범위의 값이 부호를 가지지 않고 초기치로 설정되어 있다. 각각은 8bit의 범위를 가진다.

2-2-3. Sync signals

Hsyncb, Vsyncb 이 두개의 sync signal은 active상태를 알려주며, delay만 주어 다음 block으로 전달한다. Skip_pixelb는 연산을 하지 않고 그냥 bypass할 pixel의 정보를 알려주며, control register skip_en에 의해서 masking 된다. Control register bypass는 입력신호를 연산할지 그냥 bypass할지를 선택하게 한다. 모든 경우에 있어서 동일하게 5 clock의 delay를 갖게 된다.

2-3. Adjustment

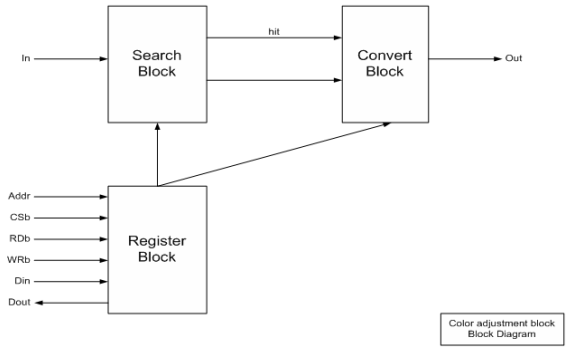

그림4. Adjustment Block diagram

그림 4는 Adjustment block의 세부 daigram이다. 최초의 Adjustment Block의 입력은 search block을 통해 변환 가능한지 판가름한 후 mux의 control signal인 hit 신호를 보냄으로써 Convert block에서의 연산이 이루어진다.

2-3-1. Search Register

Search register는 총 4 개의 slice로 구성되며, 각 slice는 하나의 1차 방정식을 표현한다. 4개의 slice는 모두 동일한 구조를 가진다. 각 1차 방정식의 연산 결과가 register에 저장된다. Y영역 검사를 위해서 14bit register 2개가 필요하다.

Slice 1의 방정식은 다음과 같이 표현된다.

$$aCb + bCr + c = ?$$

이와 같이 총 3개의 register로 구성된다. a, b는 9bit register로 구성되며, c는 16bit register로 구성된다. 모든 register가 8bit 이상이므로 각각은 2byte의 address 공간을 가진다. 따라서 하나의 slice는 6byte의 address 공간을 갖는다. 이중 a에 할당된 address의 상위 바이트중 최상위 bit은 연산 결과가 참인 부호의 값을 저장한다. 모든 slice의 address 공간은 총 4개의 slice이고, 각 slice는 6byte를 사용하므로 총 24byte의 공간을 사용한다. 여기에 Y영역 검사를 위해서 4byte가 더 필요하다. 따라서 총 28byte 가 필요하다.

2-3-2 Convert Register

변환 모듈은 2개의 1차 방정식으로 구성된다. 각 방정식은 하나의 color를 변환하는데 사용된다. Y는 1차 방정식을 사용하며, 계수로 2개의 register를 사용한다. 방정식 변환에 사용되는 1차 방정식은 다음과 같이 표현된다.

$$Cb' = ACb + BCr + C$$

$$Cr' = DCb + ECr + F$$

여기서 A, B, D, E는 모두 16bit register를 사용하고, C, F는 20bit register를 사용한다. 따라서, 하나의 방정

식을 표현하기 위해서 7byte의 address 공간을 사용한다. Y변환은

$$Y = Gy + H$$

로 표현할 수 있다. 여기서 G, H 는 각각 13bit, 15bit으로 구성된다.

2-3-3. 좌표변환 검증.

우선 hardware에서 unsigned Cb, Cr를 사용하므로 연산하기 전에 signed 값으로 변환하여 위 식을 적용하여야 하고, 연산 결과를 다시 unsigned 값으로 변환하여야 한다. 이를 식으로 표현하면 다음과 같다.

Cb' = A(Cb-128) + B(Cr-128) + C
= a(Cb-128) + 2b(Cr-128) + c + 128
= aCb + 2bCr + c - 128a - 256b+ 128
Cr' = D(Cb-128) + E(Cr-128) + F
= d/2(Cb-128) + e(Cr-128) + f/2 + 128
= dCb/2 + eCr + f/2 + 128 - 64d - 128e
A_Cb = a
B_Cb = 2b
C_Cb = c - 128a - 256b+ 128
D_Cr = d/2
E_Cr = e
F_Cr = f/2 + 128 - 64d - 128e

좌표변환 검증을 통해 하드웨어에 맞는 계수로의 변환을 하게 된다.

Ⅲ. Synthesis 및 Simulation

그림 5는 Xilinx ISE를 사용하여 Conversion calculation block을 합성한 결과를 보여주고 있다. 그림 6은 Modelsim을 사용하여 top block에서의 출력인 r_o, g_o, b_o 값들의 simulation한 결과를 보여주고 있다. 여기서 사용된 설계 tool은 IDEC에서 제공되었다.

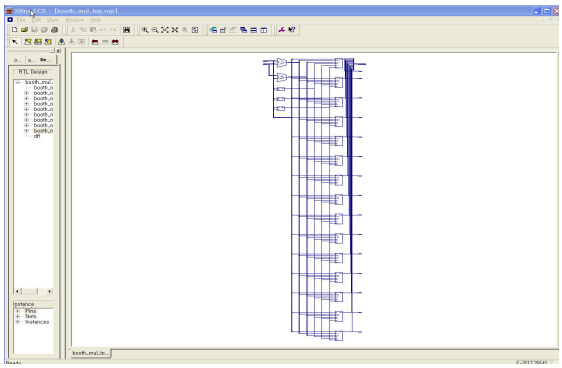

그림5. Conversion calculation block synthesis

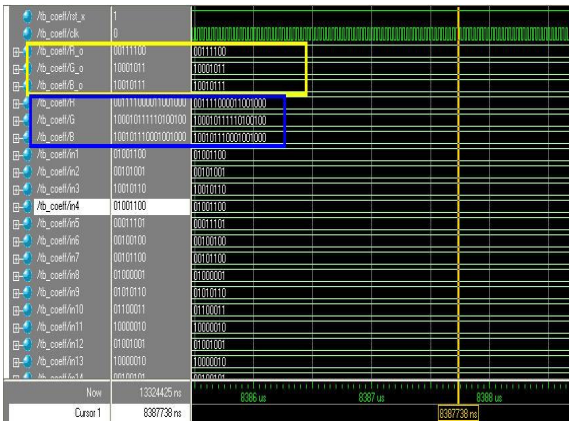

그림6. Converted & Adjusted R_out, G_out, B_out data

Ⅳ. 결론

본 논문에서는 Verilog HDL을 사용하여 SDTV용 8bit 색상보정기를 설계하였다. 곱셈 연산이 많은 Conversion block이 존재함에 따라 연산량과 처리시간을 줄이기 위해 Booth algorithm을 사용하였다. 속도를 높이기 위하여 parallel 출력방식으로 설계하였다. Xilinx ISE을 사용하여 합성하였으며, Modelsim을 사용하여 simulation을 수행하여 동작을 확인하였다.

참고문헌

[1] K. Jack, Video Demystified; A Handbook for the digital Engineer, 3rd Ed, Eagle Rock; LLH Technology Publishing, 2001.

[2] M. D. Ciletti, Advanced Digital Design with the Verilog HDL, New Jersey; Prentice Hall, 2000.

[3] S. Abet and G. Marcu, "A neural network approach for RGB to YMCK color conversion" Proceedings of TENCON '94, pp. 22-26, Aug. 1994.

[4] D. Chai and A. Bouzerdoum, "A Bayesian approach to skin color classification in YCbCr color space" Proceedings of TENCON 2000, vol. 2, pp. 24-27, Sept. 2000.

[5] H. M. Kim; W.-S. Kim; D.-S. Cho, "A new color transform for RGB coding" International Conference on Image Processing, vol. 1, pp. 24-27, Oct. 2004.

[6] S. Palnitkar, Verilog® HDL: A Guide to Digital Design and Synthesis, 2nd ed., Prentice Hall Publishing, 2003.

[7] http://www.opencores.org/

[8] http://www.asicfpga.com/