Implementing Color Reproduction for the TC236 CCD Sensor Using the TMS320C2xx DSP

APPLICATION REPORT: SPRA427

Vivian Shao
Texas Instruments Taiwan Limited

Digital Signal Processing Solutions
April 1998

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Implementing Color Reproduction for the TC236 CCD Sensor Using the TMS320C2xx DSP

Abstract

The Charge-Coupled Device (CCD) sensor is widely used in many video applications such as scanners, digital cameras, video camcorders, etc. In these video systems, a processor or hardware circuit processes the CCD raw data before displaying or implementing other image processing algorithms. The process of producing color data from the CCD raw data is called color reproduction.

In this application note, color reproduction is implemented for the Texas Instruments (TI™) TC236 CCD image sensor using a TI TMS320C2xx digital signal processor (DSP) and color reproduction software. The integration of the TI TC236 and TMS320C2xx is an important process for success in the digital camera market.

Product Support

Related Documentation

The following list specifies product names, part numbers, and literature numbers of corresponding TI documentation.

- *TMS320C2xx User's Guide*, January 1997, Literature number SPRU127B
- Data Sheet, *TC236P 680- x 500-Pixel CCD Image Sensor*, June 1996, revised April 1997, Literature number SOCS055A

World Wide Web

Our World Wide Web site at **www.ti.com** contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to **dsph@ti.com**. Questions receive prompt attention and are usually answered within one business day.

Introduction

The raw data captured by the CCD sensor must be processed before it can be displayed on a digital video device, such as a cathode-ray tube (CRT). Color reproduction is one of the most important and essential image processing algorithms because the color data captured by the CCD provides only one color data (Red, Green, or Blue) per pixel as specified by the TI TC236 CCD image sensor (see the TI data sheet, *TC236P 680- x 500-Pixel CCD Image Sensor*). To make sure each pixel has real color (a combination of three basic colors R, G, B), color reproduction is required.

There are three main sections in this application note. The first section, *Color Reproduction Algorithm*, describes the TC236 color-filter topology map and the formula for calculating reproduced data. The second section is *Software Description*, which explains how to implement the algorithm using the TMS320C2xx software. The third section is the *Summary*. The complete software source code is included in Appendix A.

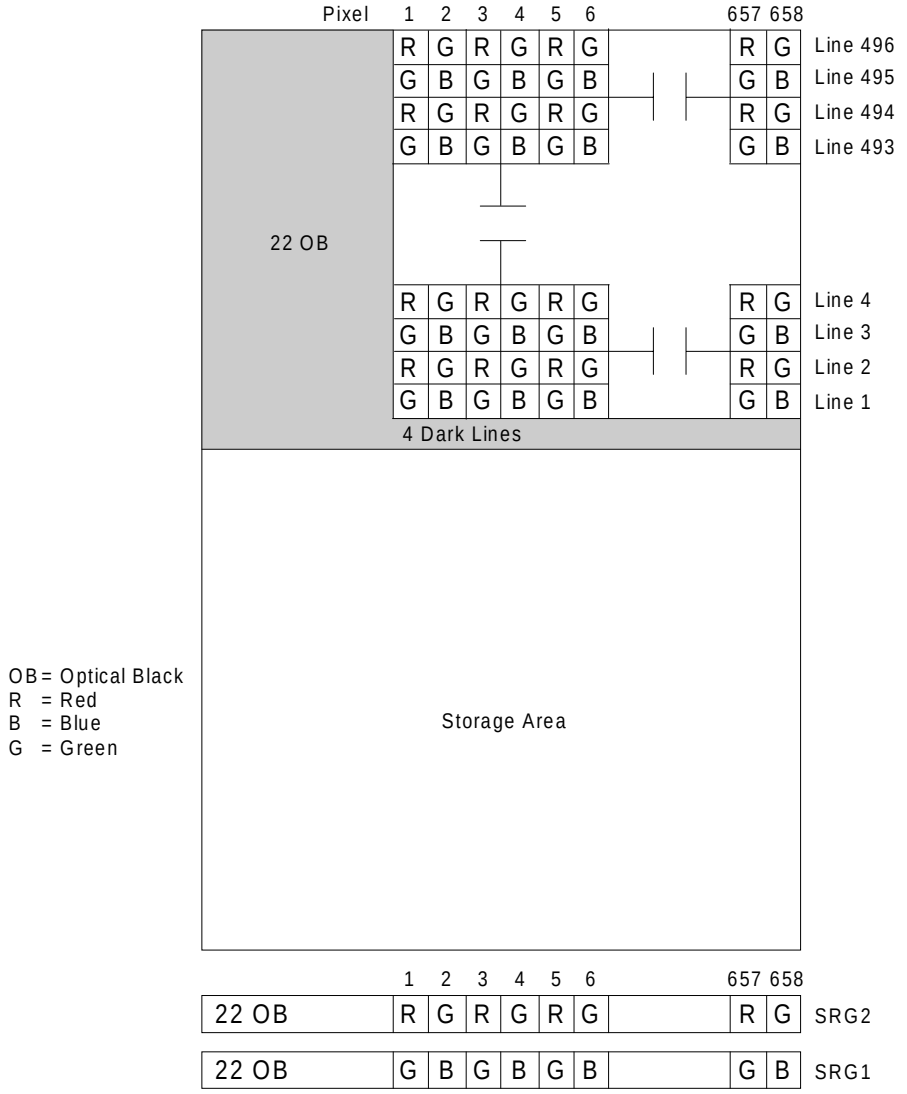
Color Reproduction Algorithm

The color components of TC236 are R, G, B, and the color filter topology map is shown in Figure 1. According to Figure 1, there are 500 lines per frame and 680 pixels per line; each line includes 22 black pixels and 658 colored pixels. After capturing the image, the CCD raw data will be shifted out serially, pixel by pixel. Therefore, there will be 658x4 black pixels (OB), then 22 black pixels of line 1, then the colored pixels R (line 1, pixel 1), G (line 1, pixel 2), R (line 1, pixel 3), etc.

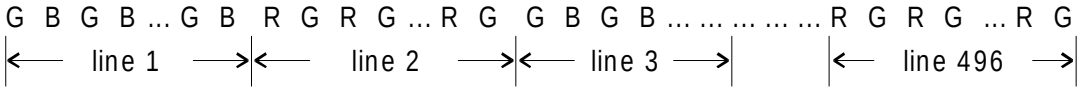
Figure 1. Color-Filter Topology Map

TC236P
680 x 500 PIXEL CCD IMAGE SENSOR

SOCS055A - JUNE 1996 - REVISED APRIL 1997



Assume that only the pixels with color information are stored in memory. The data in memory will be located sequentially as:



The objective of color reproduction is to produce {R, G, B}$_{i,j}$, the three-color data, for each pixel (i, j), where i, j means the j^{th} pixel of the i^{th} line. The color reproduction algorithm used here interpolates the data of adjacent pixels to generate the missing color information of the present pixel. The general formula used to calculate the data is:

Odd Line, Odd Pixel:

$$R_{i,j} = \frac{1}{2}\left[R_{i-1,j} + R_{i+1,j}\right]$$
$$G_{i,j} = G_{i,j}$$
$$B_{i,j} = \frac{1}{2}\left[B_{i,j-1} + B_{i,j+1}\right]$$

Odd Line, Even Pixel:

$$R_{i,j} = \frac{1}{4}\left[R_{i-1,j-1} + R_{i-1,j+1} + R_{i+1,j-1} + R_{i+1,j+1}\right]$$
$$G_{i,j} = \frac{1}{4}\left[G_{i,j-1} + G_{1,j+1} + G_{i-1,j} + G_{i+1,j}\right]$$
$$B_{i,j} = B_{i,j}$$

Even Line, Odd Pixel:

$$R_{i,j} = R_{i,j}$$
$$G_{i,j} = \frac{1}{4}\left[G_{i,j-1} + G_{1,j+1} + G_{i-1,j} + G_{i+1,j}\right]$$
$$B_{i,j} = \frac{1}{4}\left[B_{i-1,j-1} + B_{i-1,j+1} + B_{i+1,j-1} + B_{i+1,j+1}\right]$$

Even Line, Even Pixel:

$$R_{i,j} = \frac{1}{2}\left[R_{i,j-1} + R_{i,j+1}\right]$$
$$G_{i,j} = G_{i,j}$$
$$B_{i,j} = \frac{1}{2}\left[B_{i-1i,j} + B_{i+1,j}\right]$$

The exceptions are the corner and edge pixels. The formulas for the four corner pixels are as follows:

$$R_{1,1} = R_{2,1} \qquad R_{1,658} = R_{2,657}$$
$$G_{1,1} = G_{1,1} \qquad G_{1,658} = \tfrac{1}{2}\left[G_{1,657} + G_{2,658}\right]$$
$$B_{1,1} = B_{1,2} \qquad B_{1,658} = B_{1,658}$$

$$R_{658,1} = R_{658,1} \qquad R_{658,658} = R_{658,657}$$
$$G_{658,1} = \tfrac{1}{2}\left[G_{657,1} + G_{658,2}\right] \qquad G_{658,658} = G_{658,658}$$
$$B_{658,1} = B_{657,2} \qquad B_{658,658} = B_{657,658}$$

The formulas for the first line (i=1) pixels are:

$$j = \text{even:}\quad \begin{aligned} R_{1,j} &= \tfrac{1}{2}\left[R_{2,j-1} + R_{2,j+1}\right] \\ G_{1,j} &= \tfrac{1}{4}\left[G_{1,j-1} + 2 \times G_{2,j} + G_{1,j+1}\right] \\ B_{1,j} &= B_{1,j} \end{aligned}$$

$$j = \text{odd:}\quad \begin{aligned} R_{1,j} &= R_{2,j} \\ G_{1,j} &= G_{1,j} \\ B_{1,j} &= \tfrac{1}{2}\left[B_{1,j-1} + B_{1,j+1}\right] \end{aligned}$$

The formulas for the last line (i=496) pixels are:

$$j = \text{even:}\quad \begin{aligned} R_{496,j} &= \tfrac{1}{2}\left[R_{496,j-1} + R_{496,j+1}\right] \\ G_{496,j} &= G_{496,j} \\ B_{496,j} &= B_{4961,j} \end{aligned}$$

$$j = \text{odd:}\quad \begin{aligned} R_{496,j} &= R_{496,j} \\ G_{496,j} &= \tfrac{1}{4}\left[G_{496,j-1} + 2 \times G_{496,j} + G_{496,j+1}\right] \\ B_{496,j} &= \tfrac{1}{2}\left[B_{496,j-1} + B_{496,j+1}\right] \end{aligned}$$

The formulas for the first column (j=1) pixels are:

$$R_{i,1} = R_{i,1}$$

$$i = \text{even:}\quad G_{i,1} = \tfrac{1}{4}\left[G_{i-1,1} + 2 \times G_{i,2} + G_{i+1,1}\right]$$

$$B_{i,1} = \tfrac{1}{2}\left[B_{i-1,2} + B_{i+1,21}\right]$$

$$R_{i,1} = \tfrac{1}{2}\left[R_{i-1,1} + R_{i+1,1}\right]$$

$$i = \text{odd:}\quad G_{i,1} = G_{i,1}$$

$$B_{i,1} = B_{i,2j}$$

The formulas for the last column (j=658) pixels are:

$$R_{i,658} = R_{i,657}$$

$$i = \text{even:}\quad G_{i,658} = G_{i,658}$$

$$B_{i,658} = \tfrac{1}{2}\left[B_{i-1,658} + B_{i+1,658}\right]$$

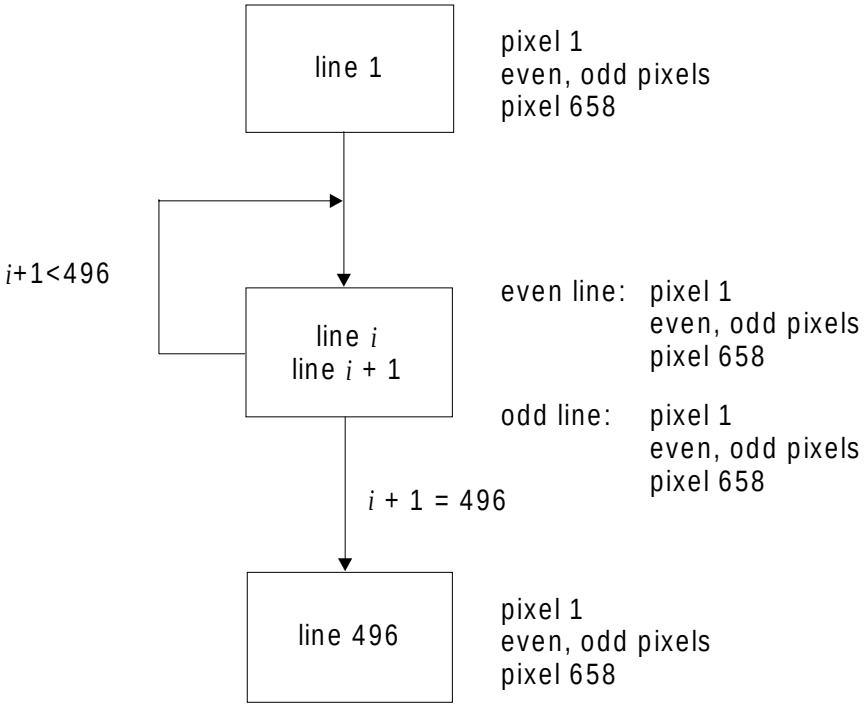
$$R_{i,658} = \tfrac{1}{2}\left[R_{i-1,657} + R_{i+1,657}\right]$$

$$i = \text{odd:}\quad G_{i,658} = \tfrac{1}{4}\left[G_{i-1,658} + 2 \times G_{i,657} + G_{i+1,658}\right]$$

$$B_{i,658} = B_{i,658}$$

We can calculate the R, G, B of each pixel by the formula shown above. The program flow chart to process an image frame is shown in Figure 2.

Figure 2. Color Reproduction Flow Chart



Software Description

The software description includes three parts. The first declares data segments and variables. The second initializes auxiliary registers as data pointers. The third implements the formulas shown in the previous section, *Color Reproduction Algorithm*.

Variable Declaration

Two segments of data are needed, one for CCD raw data and the other for the reproduction data. Therefore, two data sections are declared:

```
CFA_BLOCK       .usect  ".rgb_blk", size_of_CCD

RGB_BLOCK       .usect  ".rgb_blk", 3*size_of_CCD
```

Note the real size of TC236 colored data is 658x496=326386. This number has exceeded the memory space of C2xx, which is 64K. Therefore, the data is stored in several "banks". However, the bank switch algorithm is not included in this application report. Some modification of the source code is needed to apply this application to the hardware.

Initializing Auxiliary Registers

The data sequence stored in memory is

G B G B G B G B … line 1

R G R G R G R G … line 2

G B G B G B G B … line 3

R G R G R G R G … line 4

For the efficiency of the C2xx program, three pointers were assigned for the raw data: one pointer for the result, two pointers for loops, and AR0 is reserved as the pointer increment index.

```
;-------------------------------------------------
;  COLOR FILTER ARRAY (CFA)
;  AR1  -> [G] B G B G B G B G
;  AR2  -> [R] G R G R G R G R
;  AR3  -> [G] B G B G B G B G
;     [R] G R G R G R G R
;  RESULT
;  AR4  -> [R G B][R G B][R G B]
;-------------------------------------------------
   LAR   AR0, #2
   LAR   AR1, #CFA_BLOCK
   LAR   AR2, #CFA_BLOCK+CCD_WIDTH
   LAR   AR3, #CFA_BLOCK+CCD_WIDTH+CCD_WIDTH
   LAR   AR4, #RGB_BLOCK
   LAR   AR6, #P_LOOP_NO
   LAR   AR7, #L_LOOP_NO
```

Color Reproduction Implementation

Using the “Even Line, Even Pixels” and “Even Line, Odd Pixels” formulas from Color Reproduction Algorithm as an example, the color reproduction software is as follows:

```
CFA_EVEN_LINE_LOOP:
  ;----- j = even -----;
  LACC   *0+, 15        ; R = (Ri,j-1 +Ri,j+1)/2
  ADD    *-, 15, AR4    ;                                AR2 = Gi,j
  SACH   *+, 0, AR1
  LACC   *+, 0, AR4     ; G = Gi,j                       AR2 = Ri,j+1
  SACL   *+, 0, AR1     ;
  LACC   *+, 15, AR3    ; B = (Bi-1, j +Bi+1,j)/2
  ADD    *+, 15, AR4    ;                                AR1 = Gi-1,j+1
  SACH   *+, 0, AR2     ;                                AR3 = Gi+1,j+1
  ;----- j = odd ------;
  LACC   *-, 0, AR4     ; R = Ri,j                       AR2 = Ri,j
  SACL   *+, 0, AR1
  LACC   *-, 14, AR2    ; G = (Gi-1,j + Gi+1,j + Gi,j-1 + Gi,j+1)/4
```

```
ADD    *0+, 14         ;                                    AR1 = Bi-1,j-1
ADD    *-, 14, AR3     ;                                    AR2 = Ri,j
ADD    *-, 14, AR4     ;                                    AR3 = Bi+1,j-1
SACH   *+, 0, AR1
LACC   *0+, 14         ; B = (Bi-1,j-1+Bi-1,j+Bi+1,j-1+Bi+1,j+1)/4
ADD    *, 14, AR3      ;                                    AR1 = Bi-1,j+1
ADD    *0+, 14
ADD    *, 14, AR4      ;                                    AR3 = Bi+1,j+1
SACH   *+, 0, AR6
BANZ   CFA_EVEN_LINE_KIIO, *-, AR2
```

Similarly, “Odd Line, Even Pixels” and “Odd Line, Odd Pixels” data can be obtained, as well as data for the corner pixels. The complete program is “CFA.ASM”, which is shown in Appendix A.

Summary

The color reproduction algorithm used here is similar to an interpolation algorithm that averages adjacent data. The benefit of this algorithm is the picture will look smoother. There are other algorithms for color reproduction that use adjacent pixel data. Image characteristics and complexity requirements determine which algorithm should be used.

It takes about twelve cycles for the program to produce the R, G, B data for each pixel. That means it takes about 0.1 second to complete the color reproduction of a TC236 image frame by using 40 MIPS processing algorithms. Flexibility is one of the benefits of using DSP. This application note is a starting point for CCD image processing. The designer can implement this by adding other functions or algorithms to their specific applications.

Appendix A. TMS320C2xx Source Code for TC236 Color Reproduction

```
;-------------------------------------------------
;   Filename:       CFA.ASM
;   Description:
;   INTERPOLATION FOR COLOR FILTER ARRAY
;   Author:         Vivian Shao
;   Date:           03/20/1997
;-------------------------------------------------
      .def    CFA
size_of_CCD   set    160*120
width_of_CCD  set    160
hight_of_CCD  set    120
;-------------------------------------------------
;   Define variables & data blocks
;-------------------------------------------------
CFA_BLOCK     .usect  ".rgb_blk", size_of_CCD
RGB_BLOCK     .usect  ".rgb_blk", 3*size_of_CCD

      .text
CFA:
;-------------------------------------------------
;   Status initialization
;-------------------------------------------------
    SPM     0
    SETC    SXM
    CLRC    OVM
;-------------------------------------------------
;   AR assignment:
;   AR1 -> CFA_BLOCK          ; 1st line of CFA
;   AR2 -> CFA_BLOCK + 21     ; 2nd line of CFA
;   AR3 -> CFA_BLOCK + 40     ; 3rd line of CFA
;   AR4 -> RGB_BLOCK          ; result
;   AR6 -> Counter for frame
;   AR7 -> Counter for line
;-------------------------------------------------
    LAR    AR0, #2
    LAR    AR1, #CFA_BLOCK
    LAR    AR2, #CFA_BLOCK+width_of_CCD+1
    LAR    AR3, #CFA_BLOCK+width_of_CCD+width_of_CCD
    LAR    AR4, #RGB_BLOCK
    LAR    AR6, #hight_of_CCD/2-1
CFA_LOOP:
;-------------------------------------------------
;   COLOR FILTER ARRAY (CFA)
;   AR1 -> (B) G  B  G  B  G  B  G
;   AR2 -> G (R) G  R  G  R  G  R
;   AR3 -> (B) G  B  G  B  G  B  G
;          G  R  G  R  G  R  G  R
;
```

```
;   RESULT (RGB AFTER INTERPOLATION)
;   AR4 ->  (R  G  B)  (R  G  B)  (R  G  B)
;-------------------------------------------------
    LAR     AR7, #width_of_CCD/2-1
CFA_LOOP1:
;-------------------------------------------------
;   CASE 1:
; AR1  (B) G  B  Ri,j = Ri,j
; AR2   G (R) G  Gi,j = [Gi-1,j + Gi+1,j + Gi,j-1 + Gi,j+1] / 4
; AR3  (B) G  B  Bi,j = [Bi-1,j-1+Bi-1,j+1+Bi+1,j-1 + Bi+1,j+1] / 4
;-------------------------------------------------
    MAR   *, AR2       ;                                *=AR2
    LACC  *-, 0, AR4   ;                       AR2-1;   *=AR4
    SACL  *0+, 0, AR1  ; Ri,j = Ri,j;          AR4+2;   *=AR1

    LACC  *0+, 14      ; ACCH = (Bi-1,j-1)/4;  AR1+2;   *=AR1+2
    ADD   *-, 14, AR3  ; ACCH + (Bi-1,j+1)/4;  AR1+1;   *=AR3
    ADD   *0+, 14      ; ACCH + (Bi+1,j-1)/4;  AR3+2;   *=AR2+2
    ADD   *-, 14, AR4  ; ACCH + (Bi+1,j+1)/4;  AR3+1;   *=AR4+2
    SACH  *-, 0, AR1   ; Bi,j = ACCH;          AR4+1;   *=AR1+1

    LACC  *, 14, AR2   ; ACCH = (Gi,j-1)/4;    AR1+1;   *=AR2-1
    ADD   *0+, 14      ; ACCH + (Gi-1,j)/4;    AR2+1;   *=AR2+1
    ADD   *, 14, AR3   ; ACCH + (Gi+1,j)/4;    AR2+1;   *=AR3+1
    ADD   *, 14, AR4   ; ACCH + (Gi,j+1)/4;    AR3+1;   *=AR4+1
    SACH  *0+, 0, AR7  ; Gi,j = ACCH;          AR4+3;   *=AR7
;-------------------------------------------------
;   CASE 2:
; AR1  (G) B  G   Ri,j = [Ri-1,j + Ri+1,j] / 2
; AR2   R (G) R   Gi,j = Gi,j
; AR3  (G) B  G   Bi,j = [Bi,j-1 + Bi,j+1] / 2
;-------------------------------------------------
    MAR   *, AR2       ;                                *=AR2
    LACC  *-, 0, AR4   ; ACCL = Gi,j;          AR2-1;   *=AR4
    MAR   *+           ;                       AR4+1;   *=AR4+1
    SACL  *-, 0, AR2   ; Gi,j = Gi,j;          AR4;     *=AR2-1

    LACC  *0+, 15      ; ACCH = (Ri-1,j)/2;    AR2+1;   *=AR2+1
    ADD   *, 15, AR4   ; ACCH + (Ri+1,j)/2;              *=AR4
    SACH  *0+, 0, AR1  ; Ri,j = ACCH;          AR4+2;   *=AR1

    MAR   *+           ;                       AR1+1;   *=AR1+1
    LACC  *, 15, AR3   ; ACCH = (Bi,j-1)/2;              *=AR3
    MAR   *+           ;                       AR3+1;   *=AR3+1
    ADD   *, 15, AR4   ; ACCH + (Bi,j+1)/2;              *=AR4+2
    SACH  *+, 0, AR7   ; Bi,j = ACCH;          AR4+3;   *=AR7
    BANZ  CFA_LOOP1, *-, AR2

    MAR   *, AR1       ; change to next line
    MAR   *0+, AR2     ; pointer + 2
    MAR   *0+, AR3
    MAR   *0+, AR6
```

```
    LAR   AR7, #width_of_CCD//2-1
CFA_LOOP2:
;-------------------------------------------------
;   CASE 3:
; AR1  (G) R  G   Ri,j = [Ri,j-1 + Ri,j+1] / 2
; AR2   B (G) B   Gi,j = Gi,j
; AR3  (G) R  G   Bi,j = [Bi-1,j + Bi+1,j] / 2
;-------------------------------------------------
    MAR   *, AR1       ;                              *=AR1
    MAR   *+           ;                         AR1+1;  *=AR1+1
    LACC  *, 15, AR3   ; ACCH = (Ri,j-1)/2;      AR1+1;  *=AR3
    MAR   *+           ;                         AR3+1;  *=AR3+1
    ADD   *, 15, AR4   ; ACCH + (Ri,j+1)/2;      AR3+1;  *=AR4
    SACH  *+, 0, AR2   ; Ri,j = ACCH;            AR4+1;  *=AR2

    LACC  *-, 0, AR4   ; ACCL = Gi,j;            AR2-1;  *=AR4+1
    SACL  *+, 0, AR2   ; Gi,j = ACCL;            AR4+2;  *=AR2-1

    LACC  *0+, 15      ; ACCH = (Bi-1,j)/2;      AR2+1;  *=AR2+1
    ADD   *, 15, AR4   ; ACCH + (Bi+1,j)/2;      AR2+1;  *=AR4+2
    SACH  *+, 0, AR7   ; Bi,j = ACCH;            AR4+3;  *=AR7
;-------------------------------------------------
;   CASE 4:
; AR1  (R) G  R  Ri,j = [Ri-1,j-1+Ri-1,j+1+Ri+1,j-1 + Ri+1,j+1] / 4
; AR2   G (B) G  Gi,j = [Gi-1,j + Gi+1,j + Gi-1,j-1 + Gi+1,j+1] / 4
; AR3  (R) G  R  Bi,j = Bi,j
;-------------------------------------------------
    MAR   *, AR1       ;                              *=AR1
    LACC  *0+, 14      ; ACCH = (Ri-1,j-1)/4;  AR1+2;  *=AR1+2
    ADD   *-, 14, AR3  ; ACCH + (Ri+1,j-1)/4;  AR1+1;  *=AR3
    ADD   *0+, 14      ; ACCH + (Ri-1,j+1)/4;  AR3+2;  *=AR3+2
    ADD   *-, 14, AR4  ; ACCH + (Ri+1,j+1)/4;  AR3+1;  *=AR4
    SACH  *0+, 0, AR2  ; Ri,j = ACCH;          AR4+2;  *=AR2

    LACC  *-, 0, AR4   ; ACCL = Bi,j;          AR2-1;  *=AR4+2
    SACL  *-, 0, AR1   ; Bi,j = ACCL;          AR4+1;  *=AR1+1

    LACC  *, 14, AR2   ; ACCH = (Gi,j-1)/4;            *=AR2-1
    ADD   *0+, 14      ; ACCH + (Gi-1,j)/4;    AR2+1;  *=AR2+1
    ADD   *, 14, AR3   ; ACCH + (Gi+1,j)/4;            *=AR3+1
    ADD   *, 14, AR4   ; ACCH + (Gi,j+1)/4;    AR3+1;  *=AR4+1
    SACH  *0+, 0, AR7  ; Gi,j = ACCH;          AR4+3;  *=AR7
    BANZ  CFA_LOOP2, *-, AR2

    MAR   *, AR1       ; change to next line
    MAR   *0+, AR2     ; pointer + 2
    MAR   *0+, AR3
    MAR   *0+, AR6
    BANZ  CFA_LOOP, *-, AR2
    RET
```