Efficient transform using canonical signed digit in reversible color transforms

Se Jung Yang
Byung-Uk Lee
Ewha W. University
Department of Electronics Engineering
Seoul 120-750, Korea

Abstract. Color transforms are important methods in the analysis and processing of images. Image color transform and its inverse transform should be reversible for lossless image processing applications. However, color conversions are not reversible due to finite precision of the conversion coefficients. To overcome this limitation, reversible color transforms have been developed. Color integer transform requires multiplications of coefficients, which are implemented with shift and add operations in most cases. We propose to use canonical signed digit (CSD) representation of reversible color transform coefficients and exploitation of their common subexpressions to reduce the complexity of the hardware implementation significantly. We demonstrate roughly 50% reduction in computation with the proposed method. © 2009 SPIE and IS&T.

1 Introduction

Many color coordinate systems are employed for various purposes in color image processing. Color transform should have an exact inverse in lossless image processing applications. Most color spaces consist of three components, such as RGB; therefore, a color transform requires a 3-by-3 matrix. Since the entries in color transform matrices are not usually in fixed-point form, the matrices are approximated by binary numbers when a fixed-point processor is used. Unfortunately, the approximated matrices are in most cases irreversible and contain residual errors due to integer-rounding. After approximating a color transform as a binary number and quantizing multiplication results, the reversibility property is lost, and this has drawbacks to performance in image processing applications. In other words, when one transforms an RGB coordinate system to another system for some image processing applications and then transforms back to the RGB system, the process may cause errors if the reversibility is not guaranteed. For instance, watermarks hidden in the least significant bit (LSB) will be destroyed if the LSB of the original image is lost. In order to preserve the reversibility, a floating-point processor should be used, although this is time-consuming and inefficient. To overcome this problem, many researchers have developed integer color transforms. Among these, the state-of-the-art technology is a reversible integer color transform developed by Pei and Ding. Although integer color transforms are not accurate due to approximations, this integer transform has the benefit of reversibility without using a floating-point processor and thus has less complexity. Pei and Ding proposed a systematic algorithm based on matrix factorization to derive reversible integer color transforms. We will describe the method proposed by Pei and Ding briefly in this section. Let matrix $A$ be a normalized color transform matrix such that $\det(A) = 1$, and after performing permutation, scaling, and sign changing operations for $A$, we can obtain $C$:

$$C = D_1 P_1 A P_2 D_2,$$

where $P_1$ is a row-permuting matrix, $P_2$ is a column-permuting matrix, and $D_1$, $D_2$ are diagonal matrices having the effects of scaling and sign changing:

$$D_1[m,m] = \pm 2^{-k_n},$$

$$D_2[m,m] = \pm 2^{k_n} \quad (m = 1, 2 \text{ and } 3),$$

$$D_1 \text{ or } D_2[m,n] = 0 \quad \text{when } m \neq n,$$

where $k_n$ is an integer so that scaling can be implemented by bit shifting.

Color transform matrix $C$ can be decomposed into four one-row matrices as

$$C = T_4 T_3 T_2 T_1,$$

where

$$T_1 = \begin{bmatrix} 1 & t_1 & t_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1 & 0 & 0 \\ t_5 & 1 & t_4 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_5 & t_6 & 1 \end{bmatrix}, \quad T_4 = \begin{bmatrix} 1 & t_7 & t_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$
Yang and Lee: Efficient transform using canonical signed digit in reversible...

where \( t_1 = \frac{(c_{22} - 1)}{c_{21}} \), \( t_2 = -(t_1 z_2 + z_1) \),

\[
\begin{bmatrix}
  z_1 \\
  z_2
\end{bmatrix} = \begin{bmatrix}
  c_{21} & c_{22} \\
  c_{31} & c_{32}
\end{bmatrix}^{-1} \begin{bmatrix}
  -c_{23} \\
  1 - c_{33}
\end{bmatrix},
\]

\( t_3 = c_{21}, \quad t_4 = -z_2, \quad t_5 = c_{22} c_{31} - c_{21} c_{32}, \)

\( t_6 = c_{32} - t_1 c_{31}, \quad t_7 = c_{12} - t_1 c_{11} - t_2 g_6, \)

\( t_8 = c_{13} + z_1 c_{11} - z_2 c_{12}. \)

Note that \( C \) has eight degrees of freedom since \( |\det(C)| = 1 \); therefore, it can be represented with eight variables \( t_i \) through \( t_8 \).

Then, binary values \( g_n \) are approximated by coefficients \( t_n \) after truncation, i.e.,

\[ g_n = Q_n(t_n), \quad n = 1, 2, \ldots, 8, \]

where \( Q_n \) is a truncation operation that throws the bits less than \( 2^{-5} \). \( T_1 T_2 T_3 T_4 \) is therefore approximated to \( V_1 V_2 V_3 V_4 \) after truncation.

\[
V_1 = \begin{bmatrix}
  1 & g_1 & g_2 \\
  0 & 0 & 1
\end{bmatrix}, \quad V_2 = \begin{bmatrix}
  1 & 0 & 0 \\
  g_3 & 1 & g_4
\end{bmatrix}, \quad V_3 = \begin{bmatrix}
  1 & 0 & 0 \\
  g_5 & g_6 & 1
\end{bmatrix}, \quad V_4 = \begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0
\end{bmatrix}.
\]

Let \( B \) be the approximated transform matrix of \( A \), and then \( B \) can be expressed using Eq. (1) as

\[ B = P_1 P_2 D_2 V_4 V_4 V_3 V_2 V_1 D_1 P_1. \]

Since \( B^T B = I \) is satisfied, \( B \) is a reversible color integer transform. The method by Pei and Ding was implemented using a two’s complement. It is efficient; however, we can improve the efficiency further by changing the binary number expression.

In this paper, we propose to adopt the concept of common subexpression sharing to the reversible integer color transform for efficient implementation. The common subexpression sharing method shares common additions and subtractions among different constant multiplication coefficients. However, traditional techniques based on two’s complements are not suitable for subexpression sharing. Therefore, this paper demonstrates that inner products can be efficiently implemented by employing common subexpressions after two’s complement representation is converted to canonical signed digit (CSD) representation.

This paper is organized as follows. In Sec. 2, we give a brief overview of the CSD system. The common subexpression sharing method is described in Sec. 3. In Sec. 4, we present a new transform design that exploits the best structure of the transform system to obtain superior performance. In Sec. 5, the cost comparison and analysis are presented. Concluding remarks are given in Sec. 6.

2 Canonical Signed Digit

A signed digit number system was developed in order to improve the speed of arithmetic computation in the 1950s. The CSD number system decreases the number of nonzero digits and thus can reduce the number of partial product additions in multiplier hardware. Every \( N \)-bit two’s complement number can be uniquely expressed in \( N \)-bit CSD format. The encoding scheme uses a ternary digit set; each digit can be either +1, 0, or −1. No consecutive CSD digits are both nonzero, i.e., \( c_i, c_{i-1} = 0 \), where \( c_i \) is the \( i \)’th value in CSD digits. This property implies that any arbitrary \( N \)-bit binary number can be represented in CSD form with no more than \((N+1)/2\) nonzero digits, and often fewer; therefore, the number of adders can be reduced, since implementation of the multiplication requires as many adders as the number of nonzero bits. It was shown that the expected number of nonzero digits in an \( N \)-bit CSD number tends to \( N/3+1/9 \) asymptotically, while there can be \( N \) nonzero digits for a two’s complement number.

CSD encoding can be accomplished by analyzing pairs of adjacent bits from the LSB to the most significant bit (MSB). If the two’s complement number is negative, the MSB +1 is 1; otherwise, it is 0. Table 1 and Fig. 1 provide an explanation for two’s complement to CSD conversion, and Table 2 is an example using the CSD conversion method. The bits \( x_i \) and \( x_{i+1} \) are adjacent bits of the two’s complement number, and \( c_i \) is the CSD digit as used in general adders. Three inputs (two input bits and one carry-in bit) are entered into the adders and two outputs (one output bit and one carry-out bit) are generated in Table 1.

The flowchart in Fig. 1 describes the algorithm for conversion of two’s complement number \( X \) to CSD number \( C \), where \( X=x_n x_{n-1} x_{n-2} x_{n-3} \ldots x_3 x_2 x_1 x_0 \), and \( C=c_{n-1} c_{n-2} c_{n-3} \ldots c_0 \). Table 2 shows an example of a CSD conversion for \( N=4 \), where * represents −1. As can be noted from Table 2, the number of nonzero digits is less for the CSD representation than the two’s complement representation, 23 versus 32.

| Table 1 Conversion of two’s complement to CSD. |
|------------------|---------|---------|---------|---------|
| Carry-in | \( x_{n-1} \) | \( x_i \) | Carry-out | \( c_i \) |
| 0       | 0       | 0       | 0       | 0       |
| 0       | 0       | 1       | 0       | 1       |
| 0       | 1       | 0       | 1       | 0       |
| 1       | 0       | 0       | 0       | 1       |
| 1       | 0       | 1       | 1       | 0       |
| 1       | 1       | 0       | 1       | 0       |
| 1       | 1       | 1       | 1       | 0       |

<table>
<thead>
<tr>
<th>Table 2</th>
<th>CSD conversion example.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carry-in</td>
<td>( x_{n-1} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Journal of Electronic Imaging 033010-2  Jul–Sep 2009/Vol. 18(3)
3 Common Subexpression Sharing Method

Several methods have been proposed to further reduce the number of adders and subtracters in CSD. The arithmetic computation of CSD can be further reduced by common subexpression sharing, which shares the common subexpression among several multiplication accumulation operations so that the total number of operations is reduced.\textsuperscript{12,13} Hartley introduced a sharing method for common subexpression in CSD.\textsuperscript{5} This technique extracts common subexpressions in all CSD coefficients. Hartley shows that the savings achieved by identifying common subexpressions can be as much as 50% of the total number of operators, when several multipliers are present in a network of operators.

We illustrate the efficiency of CSD multiplication with RGB to UVW transformation by Pei and Ding’s method. Table\textsuperscript{3} shows four coefficients of the transformation coefficients represented by CSD. Each multiplication in the color transformation can be written as 
\[ g_i \cdot x, \] where \( g_i \) is a binary one-row matrix coefficient \( i=1, \ldots, 8 \), and \( x \) is binary input that can be either 1 or 0. If each coefficient is expressed in CSD format as 
\[ g_i = g_{i0} g_{i1} \cdots g_{iM-1}, \] where \( M \) is the total bit length, then the multiplication equation becomes

\[ g_i \cdot x = \sum_{j=0}^{M-1} g_{ij} \cdot (x \gg j). \] (7)

The circled groups of bits have the same subexpression. The arithmetic operation is

\[ g_5 \cdot x = -x + x \ll 2, \]
\[ g_6 \cdot x = -x \ll 3 + x \ll 5, \]

\[ g_7 \cdot x = x + x \ll 3 + x \ll 6, \]
\[ g_8 \cdot x = x + x \ll 2 + x \ll 5, \] (8)

where \( x \ll a \) denotes the left shifts of \( x \) by \( a \) bits. The common subexpressions for the four coefficients are \( 10^* \) (solid circle) and \( 100^* \) (dotted circle). If we denote them as 
\[ w_1 = -x + x \ll 2, \]
\[ w_2 = -x + x \ll 3, \] (9)

the arithmetic operations can be rewritten as

\[ g_5 \cdot x = w_1, \]

\[ g_6 \cdot x = -x \ll 3 + x \ll 5, \]

\[ g_7 \cdot x = x + x \ll 3 + x \ll 6, \]

\[ g_8 \cdot x = x + x \ll 2 + x \ll 5, \]
digits and then converted to CSD numbers. Coefficients are represented in two’s complement binary as shown in Table 5 and there are two subexpressions, $10^* x = x$ and $10^* w = w$. Thus, by sharing the common subexpression structure $w_i$, the number of additions is reduced from nine to five. Note that we are not sharing the intermediate result $w_i$; we are sharing the multiplying structure, and the output of the multiplication can be reused for other multiplications using the same structure. An implementation of common subexpression sharing is shown in the following section. In this section, only four coefficients are described for succinct explanation of the common subexpression sharing method. As shown in Table 4, the number of adders is decreased from 33 to 16. This has a higher saving rate than the preceding four-coefficient example. The common subexpression sharing method is more effective for an application having longer bits and more multiplications operations.

4 Reversible CSD Color Transform Design

In this section, we present a CSD color integer transform from RGB to UVW conversion, since other conversions follow the same procedures. As can be seen in Table 5, the coefficients are represented in two’s complement binary digits and then converted to CSD numbers.

Next, we determine common subexpressions and find the subexpressions in each coefficient. The search result is shown in Table 5 and there are two subexpressions, $10^*$ (solid circles) and $10^*$ (dotted circles), as shown in Table 5.

Now, we explain efficient implementation of the color transform. The process of the forward integer color transform is defined as $z = Bx$, where $x$ is an input RGB vector, $z$ is an output vector, and $B$ is a color transform matrix as described in Eq. (6). We rearrange the equation so that only one addition and one multiplication are executed for each step, which results in efficient hardware implementation.

Then, $V_4V_3V_2V_1$ can be represented as

$$V_4V_3V_2V_1 = W_5W_4W_3W_2W_1,$$

$$W_2 = \begin{bmatrix} 1 & g_2 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad W_3 = \begin{bmatrix} 1 & 0 \\ g_3 & 1 \\ 0 & 0 \end{bmatrix}, \quad W_4 = \begin{bmatrix} 1 & g_7 \\ 0 & 1 \\ 0 & g_6 \end{bmatrix}, \quad W_5 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{11}$$

The process of the forward integer color transform designed by Pei and Ding is as follows in Eq. (12), where $g_m$ is a CSD coefficient, $Q$ is a quantizer, and $f_1 = [f_1 f_2 f_3]$:}


<table>
<thead>
<tr>
<th>Two’s complement</th>
<th># of 1s</th>
<th>CSD</th>
<th># of 1s</th>
<th>CSD with common subexpressions</th>
<th># of 1s</th>
</tr>
</thead>
<tbody>
<tr>
<td>11011010</td>
<td>5</td>
<td>00000010</td>
<td>3</td>
<td>00000010</td>
<td>3</td>
</tr>
<tr>
<td>01111111</td>
<td>7</td>
<td>10000000+</td>
<td>2</td>
<td>10000000+</td>
<td>2</td>
</tr>
<tr>
<td>01000110</td>
<td>3</td>
<td>010010+0</td>
<td>3</td>
<td>010010+0</td>
<td>2</td>
</tr>
<tr>
<td>11100111</td>
<td>6</td>
<td>000100+</td>
<td>3</td>
<td>000100+</td>
<td>3</td>
</tr>
<tr>
<td>00000011</td>
<td>2</td>
<td>0000010+</td>
<td>2</td>
<td>0000010+</td>
<td>1</td>
</tr>
<tr>
<td>00110000</td>
<td>2</td>
<td>01000000</td>
<td>2</td>
<td>01000000</td>
<td>1</td>
</tr>
<tr>
<td>00111001</td>
<td>4</td>
<td>0100001</td>
<td>3</td>
<td>0100001</td>
<td>2</td>
</tr>
<tr>
<td>00011101</td>
<td>4</td>
<td>0100001</td>
<td>3</td>
<td>0100001</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>33</td>
<td>Total 21</td>
<td>Total 16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5 Two’s complement to CSD in RGB to UVW (∗ denotes −1, and circles indicate common subexpressions).

<table>
<thead>
<tr>
<th></th>
<th>RGB to UVW</th>
<th>2’s complement</th>
<th>CSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>-0.2930</td>
<td>11011010</td>
<td>00*0001</td>
</tr>
<tr>
<td>g2</td>
<td>1.2227</td>
<td>01111111</td>
<td>100000*</td>
</tr>
<tr>
<td>g3</td>
<td>0.5508</td>
<td>01000011</td>
<td>01000010</td>
</tr>
<tr>
<td>g4</td>
<td>-0.1953</td>
<td>11100111</td>
<td>00<em>0000</em></td>
</tr>
<tr>
<td>g5</td>
<td>0.0273</td>
<td>00000001</td>
<td>00000010</td>
</tr>
<tr>
<td>g6</td>
<td>0.3789</td>
<td>00110000</td>
<td>01000000</td>
</tr>
<tr>
<td>g7</td>
<td>0.4453</td>
<td>00111001</td>
<td>00100001</td>
</tr>
<tr>
<td>g8</td>
<td>0.2266</td>
<td>00011101</td>
<td>00100001</td>
</tr>
</tbody>
</table>

4. \[ f_3[1] = f_4[1] + g_6 f_4[2], \]
\[ f_3[2] = f_7[2], \]
\[ f_3[3] = Q(f_3[3] + g_6 f_4[2]). \]

5. \[ f_3[1] = Q(f_3[1] + g_6 f_3[3]), \]
\[ f_3[2] = f_5[2], \]
\[ f_3[3] = f_5[3], \]
\[ z = P_1^T D_2 f_6. \]

Intermediate results \( f_i \) in Eq. (12) are a matrix multiplication \( f_{s+1} = W f_i \) for \( i = 1, \ldots, 5 \).

The quantizing and adding are omitted in the implementation of the forward transform structures shown in Fig. 2 for simplicity. Three implementations are described in Fig. 2 to demonstrate the efficiency of CSD and common subexpression sharing. Figure 2(a) is the structure with two’s complement representation, Fig. 2(b) is the structure with CSD representation, and Fig. 2(c) is the structure using common subexpression sharing with CSD representation. The common subexpressions are indicated with bold squares, S1 and S2, and their reuses are indicated by bold lines. S1 and S2 are substructures implementing a given multiplication to any input \( x \). The structures are reused to execute other multiplications in delayed timings. It is obvious that theadders in the structure are reduced from (a) to (c). Specifically, the number of adders for (a) is 25 and for (b) is 13, while it is 8 for (c).

We designed common subexpression sharing and applied them to the reversible-RGB to KLA, IV1V2, XYZ, UVW, YIQ, YC8C6, and DCT transforms listed by Pei and Ding.\(^4\) KLA, the abbreviation of Karhunen-Loeve average, has the highest ability for color decorrelation. IV1V2, also known as the HIS color coordinate system, has been used as a quantitative measure of specifying the hue, intensity, and saturation of a color. DCT, one of the color space conversions for discrete cosine transform, is used for image compression such as JPEG. YC8C6 is commonly used for JPEG, but DCT is considered as a compatible color transform for JPEG with YC8C6. The definitions for XYZ, UVW, YIQ, and YC8C6, are not provided, since they are well-known color transforms. Table 6 shows the results of our proposed method applied to reversible color transforms; it represents the CSD coefficients and best common subexpressions minimizing the number of operations based on exhaustive search.

5 Cost Analysis

Table 4 shows the hardware cost comparison of two’s complement, CSD, and CSD with subexpressions for the RGB to UVW transform, which is shown in Table 3. The number of adders is decreased from 33 to 21 after adopting CSD and further reduced to 16 after utilizing common subexpressions. Therefore, the total number of adders is decreased to 16 adders and achieves a 51.5% saving compared to the two’s complement representation. We obtain similar results from other reversible integer color transforms such as RGB to KLA, YUV, and DCT, as shown in Table 7. All color transforms do not have the same savings rates; the savings rate depends on the number of common subexpressions in the color transform coefficients. However, on average, about 50% savings are achieved. In addition, the inverse transform process is symmetric as a forward transform. Thus, the cost reduction of the inverse transform is the same as that of the forward transform.

6 Conclusion

CSD representation and common subexpressions play important roles in optimization algorithms of finite impulse response (FIR) or infinite impulse response (IIR) filters.\(^5\) In this paper, we propose an efficient reversible color integer transform by adopting CSD and common
Fig. 2 Color transformation structure: RGB to UVW. (a) Two’s complement. (b) Canonical signed digit representation. (c) Common subexpression sharing. Bold squares S1 and S2 denote common subexpressions, and bold lines show reuses. We illustrate the signal flow for $g_1f_1[2]$, which is the leftmost output signal. The coefficient $g_1=00+0+010$ is from Table 4. We observe that the signal at nodes $a = 100 \cdot f_1[2]$, $b = 01 \cdot f_1[2]$, and $c = 010 \cdot f_1[2] + 00000 \cdot f_1[2] = 00+0+010 \cdot f_1[2] = g_1f_1[2]$, as shown in the diagram. Other signal flows can be analyzed using the same procedure. (Color online only.)
subexpressions. We employ CSD expression instead of two’s complement and propose a common subexpression sharing technique for further hardware saving. We show a significant amount of savings in reversible integer color transform by utilizing the proposed CSD representation and common subexpression sharing technique.

The savings of total number of operators in color transforms are around 50%. The proposed method reduces the expense of full multipliers for reversible color integer transform implementation, while maintaining the same accuracy of the results of color transform by Pei and Ding’s method as well as the same accuracy of transform matrix representation. The hardware cost of image processing today has become very low; however, decreasing of adders can reduce chip size and power consumption and also increase calculation speed. Therefore, it is especially useful for real-time applications in mobile devices.

Acknowledgments

The authors would like to thank Professor Young-Beom Jang for valuable discussions regarding the filter structure, as well as the reviewers for their helpful suggestions. This research was supported in part by the Ministry of Knowledge Economy (MKE) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Strategic Technology, and by the Acceleration Research Program of the Ministry of Education, Science and Technology of the Republic of Korea and the Korea Science and Engineering Foundation.

References


Table 7 Cost comparisons for various color transforms.

<table>
<thead>
<tr>
<th></th>
<th>RGB to KLA</th>
<th>RGB to IV/V2</th>
<th>RGB to YCbCr</th>
<th>RGB to DCT</th>
<th>RGB to YIQ</th>
<th>RGB to XYZ</th>
<th>RGB to UVW</th>
</tr>
</thead>
<tbody>
<tr>
<td># of adders in 2's</td>
<td>50</td>
<td>42</td>
<td>31</td>
<td>34</td>
<td>34</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td># of adders in CSD</td>
<td>18</td>
<td>19</td>
<td>17</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Savings</td>
<td>64.0%</td>
<td>54.8%</td>
<td>45.2%</td>
<td>41.2%</td>
<td>47.1%</td>
<td>39.4%</td>
<td>51.5%</td>
</tr>
</tbody>
</table>

Sejung Yang received an MS degree in electrical engineering from the University of Southern California in 2005. She is currently in the PhD program in electronics engineering at Ewha W. University. Her research interests include image processing and medical imaging.

Byung-Uk Lee received a BS degree in electronic engineering from Seoul National University in 1979, an MS degree in electrical science from the Korea Advanced Institute of Science and Technology in 1981, and a PhD in electrical engineering from Stanford University in 1991. He worked for Daewoo Electronics Co. from 1981 to 1985 and from 1991 to 1995. He joined the Department of Electronics Engineering, Ewha W. University, in 1995, where he is currently a professor. His research interests include image processing and computer vision.