Abstract—This paper proposes nominal decomposition for calculating color space transformation in single assignment language, to speed up execution time. Chromaticity coordinate transformation is discussed within the framework of nominal decomposition. The gamma curve look-up table for function mapping is also discussed with reference to Taylor expansion with 20-point and 100-point LUT types. The numerical value is decomposed into nominal value and base value, respectively. K-Table matrices for different bases are also proposed, in order to provide the base change for the associated derivation. Such an algorithm is suitable for the programming of single assignment C. The hardware description language VHDL can be generated by using Data Flow Graph in a single assignment C compiler. The results will show that the proposed algorithm can shorten the execution time to a certain extent. It is evident that the proposed algorithm can be very helpful in implementing the hardware-interface or software-driver for the processor display and printer system.

Index Terms—Color Space, Chromaticity Coordinate, Data Flow Graph (DFG), Look-up Table (LUT), Nominal Decomposition, Parallel Computing, Reconfigurable Computing System (RCS), Single Assignment C (SA-C), VHDL.

I. INTRODUCTION

A. Single Assignment C

Recently, systems on a chip, such as FPGA/CPLD, have drawn considerable attention in consumer electronics IC design as they represent a powerful IC design and development platform. However, the design language, such as VHDL or Verilog, still has a tedious implementation procedure for a complicated high-level signal-processing algorithm when the embedded system is considered. It is not easy to realize a high-level signal-processing algorithm directly by using VHDL or Verilog and a powerful language is required to realize complicated signal processing algorithms or embedded systems. A variety of C language becomes more and more popular due to easy programming via SystemC or HandleC.

Recently, the SA-C language was developed due to the VHDL design requirement [1]-[5]. Research on the Cameron Project, originally running at Colorado State University, has rapidly spread worldwide. The purpose of the Cameron project is to make FPGAs and other adaptive computing systems available to more applications’ designers.

Figure 1. SA-C compiler procedure for the reconfigurable computing system

B. Color Space Transformation Using Nominal Decomposition

To this end, SA-C has been developed into an alternative of C programming language and an optimizing compiler for the parallel operation [6]. The language and compiler can transfer high-level programs directly into FPGA design, especially on complicated signal processing or image processing, as well as many other system integration applications. As shown in Fig. 1, the SA-C source programs were initially translated into DFG, which is a token-driven semantics [7]. SA-C can be further translated into a type of graph called the data dependence and control flow (DDCF) graph, which is convenient for advanced optimization.

A conventional DFG, however, does not have nodes with internal state, for example, registers. Therefore, yet another round of optimization and translation transforms the data flow graph into an abstract hardware architecture graph, which is a data flow graph with state full nodes (mostly registers) and hand-shaking signals. The abstract hardware graph is then optimized one last time before VHDL is generated.
requirements.

This paper proposes an algorithm called nominal decomposition for color space transformation. An actual floating-point value can be decomposed into two independent parts: the nominal part and the base part. The nominal part can be declared as the fixed-point data type in SA-C. The base part can be declared as the integer data type. Such an algorithm has the advantage of a data-flow independent arithmetic operation that is suitable for parallel SA-C computation.

In most processor systems, such as the operation system and peripheral video interface, the sRGB follows the CIE standard for every information product. The printer color CMYK system follows the ICC standard to keep the true color for printed documents. Among the variety of chromaticity coordinates, such as XYZ and YUV, sRGB is the standard color space as the intermediate of color space transformation in the processor system [9].

As shown in Fig. 2, the proposed algorithm can convert the color space for the video signal, photos or an image. The algorithm is suitable for parallel operation due to nominal decomposition. The video information product has the interface to follow the sRGB standard. In addition, there is a gamma curve in company with the color space to modify the color mapping. The gamma curve can make an image clearer due to the gamma curve regulation.

The PC windows system has the standard gamma 2.2 for its OS, while the MAC computer has the gamma 1.8. The proposed algorithm can be implemented by either the peripheral-hardware or the driver-software approach. The color space transformation defined in the CIE chromaticity chart has many matrix conversion types. Since the video interface requires high-speed operation in many products, high-speed parallel computation suitable for speedup will be developed in this paper.

The gamma characteristic for the color space might have an effect on the image shown in the processor display system. The tristimulus xyz value can be derived from the standard sRGB coordinate by the appropriate matrix operation. The transformation formula can be further decomposed into two parts for arithmetic operation, the nominal value and the base value. Such decomposition can have the advantage of accelerating the execution time.

The gamma characteristic for the color space might have an effect on the image shown in the processor display system. The tristimulus xyz value can be derived from the standard sRGB coordinate by the appropriate matrix operation. The transformation formula can be further decomposed into two parts for arithmetic operation, the nominal value and the base value. Such decomposition can have the advantage of accelerating the execution time. Only a simple data type, such as integer and fixed point, are required. Floating-point and double data type are not required in this proposed algorithm. By using SA-C, the nominal value is declared as the fixed-point data type, such as fix or ufix. The base value is declared as integer value, such as int or uint.

Therefore, the proposed algorithm can have the potential to accelerate the execution time. Only integer and fixed-point arithmetic operation are required for the proposed algorithm. Such a data-flow independent algorithm is especially suitable for SA-C programming with parallel operation. As shown in Fig. 3, the data-flow graph can be converted into a reconfigurable computing system. The compiler can implement the proposed algorithm into circuit-level models by VHDL programming without too much data-dependency in RCS instead of the host, as shown in Fig.1. As shown in Fig. 4, the nominal value operation and the base value operation are independent of each other.

II. COLOR SPACE TRANSFORMATION

A. Representation of Arrays in SA-C

In general, the arrays in SA-C can be represented by two vectors [10]. A data vector contains all array elements in row-major order, and a shape vector specifies the number of elements per axis. Let A be an n-dimensional array with shape vector and data vector respectively.

\[\mathbf{sv} = [sv_{0}, sv_{1},..., sv_{n-1}] \quad \mathbf{dv} = [dv_{0}, dv_{1},..., dv_{n-1}]\]

Then the length l of the data vector should be

\[l = \prod_{k=0}^{n-1} sv_{k}\]

Sub-arrays or elements of the array may be addressed by index vectors of the set:

\[
\{[iv_{0}, iv_{1},..., iv_{m-1}] | (0 \leq m \leq n) \land (\forall k \in [0,1,...,m-1]: 0 \leq iv_{k} \leq sv_{k})\}
\]
An index vector \( iv = [iv_0, iv_1, ..., iv_{m-1}] \) selects the sub-array with shape vector \([sv_0, sv_1, ..., sv_{n-1}]\) and data vector: \([dv_p, dv_{p+1}, ..., dv_{p+q-1}]\), where \( p \) and \( q \) satisfy:

\[
p = \sum_{i=0}^{n-1} (iv_i) \prod_{j=0}^{i-1} sv_j \quad q = \sum_{i=n}^{l-1} sv_i
\]

The special cases \((m=0)\) and \((m=n)\) specify the selection of the whole array and the selection of the single array element \(dv_p\) respectively.

B. Matrix Operation Algorithm for SA-C

SA-C has modified data type signed or unsigned integers and fixed-point numbers with user-defined bit-widths. Nevertheless, SA-C has multi-dimensional rectangular arrays so that its extents are determined dynamically or statically. For example, the type declaration \( \text{int14 M}[.,6] \) is a declaration of a matrix \( M \) of 14-bit signed integers. The left dimension is determined dynamically, while the right dimension is specified by the user.

What is most important is the loop structure. The loop has three parts: a generator, a body, and a return expression. By using the for loop in SA-C, many parallel matrix operations can be realized very easily, e.g.,

```c
for _ in [n] return (tile (B))
for _ in [m,n] return (tile (B))
```

will produce \( m \) and \( n \) iterations without having to declare an iteration variable. This will be helpful to the matrix expansion or flattening discussed in this paper.

Type I: Flattening a matrix to a vector:

```c
1:   int16[:] main (int16 A[;,:]) return{
2:   for V (~,:) in A return (tile (V))
}
```

Type II: Restructuring a matrix to a vector:

```c
1:   int16[;:] main (int16 A[:], int16 n)
2:   {int16 s=extents(A);
3:   assert (s%n==0, "not rectangular",s,n);
4:   int16 res[;,:];for window V[n] in A step 5:
   (n) return (array(V)); } return(res);
```

C. Chromaticity Coordinate Transformation from sRGB to YUV Transformation

The YUV model defines a color space in terms of one luminance and two chromaticity components. YUV is used in the PAL and NTSC systems of video signal for the processor system and it is the standard in much of the world [11]. The YUV models human perception of color more closely than does the standard RGB model used in processor graphics hardware, but not as closely as HSL color space and HSV color space. Y stands for the luminance component and \( U \) and \( V \) are the chromaticity components. The YCbCr or YPbPr color space used in video interface, are derived from YUV. Cb/Pb and Cr/Pr are simply scaled versions of \( U \) and \( V \), and are sometimes inaccurately called YUV.

YUV signals are created from an original RGB source. The weighted values of \( R \), \( G \) and \( B \) are added together to produce a single \( Y \) signal, representing the overall brightness, or luminance, of the pixel. The \( U \) signal is then created by subtracting \( Y \) from the blue signal of the original RGB, and then scaling. Also, \( V \) is created by subtracting \( Y \) from the red color, and then scaling by another factor. The following equations can be used to derive \( Y \), \( U \) and \( V \) from \( R \), \( G \) and \( B \) in compact form:

\[
C_{YUV} = M_{VR} C_{sRGB}
\]  (4)

The standard elements of the above matrix can be described in detail as follows:

\[
Y=(0.299)\ R+(0.587)\ G+(0.114)\ B
U=(0.492)(B-Y)=(-0.147)\ R-(0.289)\ G+(0.436)\ B
V=(0.877)(R-Y)=(0.615)\ R-(0.515)\ G-(0.100)\ B
\]  (5)

\( M_{VR} \) is defined with respect to the different color temperature conditions.

D. Chromaticity Coordinate Transformation from XYZ to sRGB

In a processor system, the quantitative three-color gray scale is clearly defined by the CIE 1931 standard [12]. The coordinate transformation between the chromaticity coordinate XYZ and standard RGB (sRGB) is the basic operation for the processor display system [13].

\[
C_{XYZ} = M_{XR} C_{sRGB}
\]  (6)

In complete form, the matrix can be written as [14]:

\[
X=(0.4124)\ R+(0.3576)\ G+(0.1805)\ B
Y=(0.2126)\ R+(0.7152)\ G+(0.0722)\ B,
Z=(0.0193)\ R+(0.1192)\ G+(0.9505)\ B
\]  (7)

Note that \( M_{XR} \) is defined with respect to different color temperature conditions. To implement the coordinate transformation into hardware or software, nominal decomposition is proposed for speeding up the numerical operation.

In some interface cards, nominal numerical expression is especially suitable for the arithmetic operation. Some processor interfaces often adapt the ITU-RBT. 709 color system [15]. In such a color system, the white point is defined as D65 with a color temperature of 6500K. When the digital camera receives the image, the processor software processes the image and then saves it to the specific image format. D65 is often used as the associated format. However, the printer system usually prints the image under the format of D50. Therefore, the concept of color space transformation has to be considered here to keep the true color of the images or documents. The related white-point calculation is performed in the verification.

E. Generalized Derivation for the Color Space Transformation

To describe the above two transformations of Eqs (4) and (5) in detail, the generalized equation can be written in the following form, with three scaling factors:
\[ X = m_1 R + m_{12} G + m_3 B, \quad Y = m_{21} R + m_{22} G + m_{23} B \]
\[ Z = m_3 R + m_{12} G + m_3 B \]

(8)

Let the three factors be equal to -2, then we have:
\[ X_{\text{norm}} X_{\text{base}} = ((m_{11})_{\text{norm}} (R)_{\text{norm}} 2^{-2} + (m_{12})_{\text{norm}} (G)_{\text{norm}} 2^{-2} + (m_{13})_{\text{norm}} (B)_{\text{norm}} 2^{-2}) 2^{k_{\text{ext}}} \]

(15)

With nominal decomposition, the above expression can be decomposed as the following two independent equations. Note that \( 2^{-2} = 0.25 \) is the simple right-shift 2 operation in nominal operation. The base \( X_{\text{base}} = 2^k_{\text{ext}} \):
\[ X_{\text{norm}} X_{\text{base}} = ((m_{11})_{\text{norm}} (R)_{\text{norm}} 2^{-2} + (m_{12})_{\text{norm}} (G)_{\text{norm}} 2^{-2} + (m_{13})_{\text{norm}} (B)_{\text{norm}} 2^{-2}) \]

(16)

Since there are three terms in the linear combination, the scaling factor 0.25 can be specified further under the proposed algorithm and numerical saturation can be avoided. Any software-based driver or hardware-based processor can implement the algorithm very easily by using only fixed-point and integer data type in SA-C. The algorithm is based on the fixed-point operation.

The fixed-point processor or the FPGA/CPLD chip in the interface card can adapt the algorithm very easily by using the SA-C compiler. Since the operation is decomposed into nominal value and base value, the calculation can be further implemented by parallel operation without any coupling terms.

F. K-Table Matrix for Base Change

The above derivation has described detailed nominal decomposition for the \( X \) variable. For simplification, the \( Y \) and \( Z \) variables have the same derivation which can be summarized as K-Table for the base change. The color space transformation for YUV is also the same and can be defined as another K-Table, as shown in Fig. 4. K-Table can be realized by using manifold array operations provided in SA-C.
\[ K_{\text{comYUV}} = \begin{bmatrix} k_{\text{comY}} & k_{\text{comU}} & k_{\text{comV}} \end{bmatrix} \quad (17-2) \]

Then for \( \gamma \) in [3,1] return (tile \((k\text{omyuv})\)) will create a matrix with extents in SA-C with parallel operation. To tell the Y of YUV from the Y of XYZ in notation, the Y’ is defined for the purpose of identification.

III. GAMMA CURVE MAPPING FUNCTION

A. Color Modification with LUT

In processor system, the digital camera with CCD/CMOS devices can capture the images outside of the processor system. Digital sensors such as CCD/CMOS are linear. That means the voltage generated in each pixel and the pixel level emerging from ADC are proportional to exposure. Gamma correction is required to create better image quality.

To provide the unified coordinate for the color space of Windows and peripheral systems, sRGB is the color space standard defined by Hewlett-Packard and the Microsoft Corporation. The digital encoding for sRGB is 8-bit standard defined by the International Journal of Information and Electronics Engineering, Vol. 1, No. 1, July 2011.

The proposed algorithm can be helpful in keeping the maximum resolution of numerical operation. In general, gamma 2.2 is widely adapted as the following calculation formula [16]:

\[ \gamma \text{spec} = \frac{\gamma}{2.2} \]

The maximum digital resolution for the numerical operation: \( \gamma \) is the gamma curve specification of the \( j \)-th color for the processor display system. The LUT unit is required to regulate the gamma curve for the color signal. However, the LUT unit usually requires a larger memory to

\[ \gamma(2.2) \text{spec}^j = f_j(R_{\text{RGB}}, G_{\text{RGB}}, B_{\text{RGB}}) \quad (20) \]

The above three functions are nonlinear. LUT is required to obtain the associated function mapping. In order to compute the functions easily, piecewise linear approximation is required. Taylor expansion combined with LUT is used to expand the approximation locally, which is then integrated using the proposed global Taylor approximation. The local limitation is usually the weak point of the Taylor expansion. By dividing \( N \) sections to keep the validity of the local approximation, the global approximation will be integrated by using the summation of the local approximation in the individual sections. In the local sectional region, Taylor expansion will guarantee the validity of the global approximation to be effective. The accuracy range for the division can be further determined by considering the maximum acceptable error.

The case which is smaller than 0.00304 can be decomposed as follows:

\[ R_{\text{RGB}} = 12.92 R_{\text{RGB}} \]

For the 8-bit operation, the proposed method can get the maximum digital resolution for the numerical operation:

\[ R_{\text{RGB}} = (R_{\text{RGB}})_{\text{norm}}(R_{\text{RGB}})_{\text{base}} \]

B. Gamma Curve Implementation for Color Space Transformation

This paper proposes a nominal decomposition algorithm suitable for the digital implementation of color space transformation in the processor display and printer system. This algorithm can be used in either the hardware or the software. The nominal decomposition will convert the CIE chromaticity coordinate from the sRGB signal.

Gamma correction is also discussed as a way to regulate the appropriate image quality in different processor display systems. The gamma curve for the PC Windows system is gamma value 2.2. The gamma value for the MAC computer system is 1.8. Each processor system exhibits its own display texture. The gamma characteristic curve can be generally defined as:

\[ Y = X^{(\gamma_{\text{spec}})} \]

where \((\gamma_{\text{spec}})^j\) is the gamma curve specification of the \( j \)-th color for the processor display system. The LUT unit is required to regulate the gamma curve for the color signal. However, the LUT unit usually requires a larger memory to
map the nonlinear function.

Conventional point-to-point mapping may be required to obtain the function value. Such an approach will occupy a large amount of memory in order to be precisely accurate. To save memory space for the gamma correction, a Taylor expansion with parallel structure is proposed in this paper.

As shown in Fig. 4, the proposed algorithm can accelerate the execution of color space transformation due to the parallel structure. SA-C based programming can implement the proposed algorithm very conveniently. SA-C can compile the algorithm into DFG and translate the DFG codes into VHDL source codes. The algorithm for the gamma correction can be implemented in the software-based system driver or in the hardware-based interface card.

C. Global Taylor Expansion Combined with LUT

Usually, the look-up table needs considerable memory space to implement the look-up table for function mapping [17]. Taylor expansion combined with LUT can be used to save memory space. Since the gamma curve is a function with a slow change rate, Taylor expansion for the look-up table should be enough for approximation.

As shown in Fig. 5, N sections of the input variable were divided to reasonably approximate the gamma curve in a global region. N value is illustrated as 20 and 100 in this paper. The 20-point and 100-point cases will be observed upon verification. By dividing the global region into N sections, the central points of the N sections are \(X_i\sim X_N\). To specify the individual gamma value for each color, the \(j\) index is used to identify the color:

\[
Y = X^{(\gamma_{20})} = f_j(X_i) + f_j'(X_i)(X - X_i), \quad i = 0 - N, j = R, G, and B.
\]

where the \(X_i \sim X_N\) are the specified points defined in the look-up table. \(h = X_i - X_{i-1}\) is the neighborhood distance between the two adjacent points. The function approximation for the neighborhood of the \(X_i\) can be:

\[
\text{if } X_i - h/2 \leq X < X_i + h/2, \text{then } Y = f + f'(X - X_i)
\]

where \(X_0 \sim X_N\) are the central points defined in the look-up table for the \(N\) sections. The look-up table for the three-color can be built-up as shown in Fig. 5. With LUT, the gamma curve can be calculated in the form of a linear function. The linear function is a linear combination of the related terms. The addition and multiplication arithmetic operations can be used for further SA-C programming which is easily translated into VHDL for FPGAs.

As shown in Fig. 5, we can expand the single-color LUT into the three-color LUT required for the gamma curve. The individual gamma value can be further regulated independently due to the color temperature effect. The white balance function can be modified by regulating the individual gamma values to compensate the color temperature for the image in the processor system.

To speed up the proposed algorithm, nominal decomposition was used again. To express for clarity, the equation can be represented as the linear function with slope \(m_j\) and y-intercept \(b_j\) parameters below:

\[
Y = X^{(\gamma_{20})} = f_j(X_i) + f_j'(X_i)(X - X_i) = m_jX_i + b_j
\]

To decompose the above equation, the nominal decomposition can be further expressed as:

\[
Y = (m_j)_{base}(X_i)_{base} + (b_j)_{base}
\]

where \(m_j\) and \(b_j\) are the bases for the three variables with power of 2:

\[
(m_j)_{base} = 2^{k_{mj}}(X_i)_{base} = 2^{k_{mj}}(b_j)_{base} = 2^{k_{bj}}
\]

The nominal variables are defined as the following form:

\[
\begin{align*}
(k_{mj})_{norm} &= \frac{m_j}{(m_j)_{base}} \leq 1, \quad (X_i)_{norm} = \frac{X_i}{(X_i)_{base}} \leq 1 \\
(b_j)_{norm} &= \frac{b_j}{(b_j)_{base}} \leq 1
\end{align*}
\]

\[
\begin{align*}
(k_{mj})_{norm} &= \frac{m_j}{(m_j)_{base}} \leq 1, \quad (X_i)_{norm} = \frac{X_i}{(X_i)_{base}} \leq 1 \\
(b_j)_{norm} &= \frac{b_j}{(b_j)_{base}} \leq 1
\end{align*}
\]

\[
\begin{align*}
(k_{mj})_{norm} &= \frac{m_j}{(m_j)_{base}} \leq 1, \quad (X_i)_{norm} = \frac{X_i}{(X_i)_{base}} \leq 1 \\
(b_j)_{norm} &= \frac{b_j}{(b_j)_{base}} \leq 1
\end{align*}
\]

\[
\begin{align*}
(k_{mj})_{norm} &= \frac{m_j}{(m_j)_{base}} \leq 1, \quad (X_i)_{norm} = \frac{X_i}{(X_i)_{base}} \leq 1 \\
(b_j)_{norm} &= \frac{b_j}{(b_j)_{base}} \leq 1
\end{align*}
\]

The K-Table matrix can be formed by the definition \(K_\text{combi} = \text{[K}_{\text{combi}]}\). Then the semantic expression for \_ in [3,1] return (tile (kcommb))will create a matrix by using extents in SA-C with parallel operation. If we want to make the following condition bounded, then \(|Y_j|_{norm} < 1.0\). We should select the scaling factor as follows to avoid from numerical saturation. Therefore, the following parameters can be determined by the above
inequality: $k_{ij} = -1 \cdot k_{ij} + k_{ij} = -1$. This means we can select the following as the reasonable factors to perform the right-shift operation:

$$k'_{ij} = -1, \text{let } k'_{ij} = 0, k'_{ij} = -1 \text{ or } k'_{ij} = -1, k'_{ij} = 0.$$  \hfill (33)

D. Bounded Proof for Taylor Expansion

This section discusses the accuracy of the remainder function $R(X)$ of the Taylor series approximation:

$$Y = X^{(\gamma_{\text{spec}})} = f_j(X_i) + f'_j(X_i)(X - X_i) + R_j(X)$$ \hfill (34)

where $X_i - h/2 < \xi < X_i + h/2$,

$$|R_j(X)| = \left| \frac{f(\xi)}{2!} (X - X_i)^2 \right|$$

\begin{align*}
&< \left| \frac{(\gamma_{\text{spec}})_j}{2!} (X - X_i)^2 \right| \\
&< \frac{\left| (\gamma_{\text{spec}})_j \right| h^2}{2!} \cdot 4 \\
&= c_i
\end{align*} \hfill (35)

The accuracy function can be derived from the following compact form. Since the $(\gamma_{\text{spec}})_j - 1 < 0$ is a negative value for this $(\gamma_{\text{spec}})_j = 1/2, 4$ case, the $c_i$ can be the form

$$c_i = \frac{(\gamma_{\text{spec}})_j h^2}{8 X_i (\gamma_{\text{spec}})_i^3}$$ \hfill (36)

The above shows that the larger the $X_{i+1}$, the less accuracy there is. In the next verification section, the accuracy functions for the 20-point LUT and 100-point LUT are provided.

If you are using Word, use either the Microsoft Equation Editor or the MathType add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation or MathType Equation). “Float over text” should not be selected.

IV. VERIFICATIONS

A. Comparison with Execution Speed

Color space transformations are often applied in many processor peripheral and interface cards. Therefore, the proposed algorithm requires rapid computing to fulfill the real-time image-processing requirement.

The SA-C compiler can generate the parallel computing VHDL for the circuit-level models of the associated algorithm. Some speedup tests are illustrated to show the capability of fast computing.

As shown in Table 1, the verification illustrates three cases for study: the ANSI-C VC program running on PC and SA-C running on the LINUX system for PC with fixed-point and floating-point. Two of the realization approaches were compared: the fixed-point realization (nominal decomposition used) and the floating-point realization (nominal decomposition not used). The following cases were verified on LINUX system on the 1GHz Pentium PC. The VHDL were verified on Xilinx FPGA platform under 40MHz clock.

Case 1: Algorithm run using SA-C with fix data type; SA-C compiler compiles fixed-point operation for proposed algorithm

Case 2: Algorithm run using SA-C with float data type; SA-C compiler also compiles direct floating-point operation

Case 3: Algorithm run using ANSI-C with float data type; ANSI-C VC compiler compiles direct floating-point operation

<table>
<thead>
<tr>
<th>Case</th>
<th>Pentium PC time (ms)</th>
<th>Pentium PC cycles</th>
<th>FPGA time (ms)</th>
<th>FPGA cycles</th>
<th>Flip-flop ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>4</td>
<td>20%</td>
<td>0.2</td>
<td>20%</td>
<td>20</td>
</tr>
<tr>
<td>Case 2</td>
<td>30</td>
<td>46%</td>
<td>0.19</td>
<td>46%</td>
<td>157</td>
</tr>
<tr>
<td>Case 3</td>
<td>60</td>
<td>49%</td>
<td>0.48</td>
<td>49%</td>
<td>125</td>
</tr>
</tbody>
</table>

As shown in Table 1, the comparison shows that Case 1 spends less time to calculate the algorithm. Case 3 spends more time to calculate the algorithm. Results show that the proposed algorithm combined with the SA-C programming will be faster than the conventional programming using SA-C only. Conventional SA-C programming will declare the numerical operation by using float data type directly. By verification, the proposed algorithm is faster than the conventional technique. SA-C programming also proved to be faster than the conventional ANSI-C VC language that is sequentially programmed. The sequential C programming technique might have critical compatibility problems with circuit-level models by VHDL description. VHDL can execute the computing based on the logic design which is considered in the circuit level. While the sequential C programming has to execute the instruction within one instruction cycle, SA-C has the reconfigurable capability to perform the parallel computing.

B. Computing of Color Space Transformation

The tristimulus value, nominal tristimulus value, and chromaticity diagrams were calculated and plotted as shown in Figs. 6 and 7, and are close to the standard CIE 1931 results [18]. In order to verify the validity, a number of cases for white-points under different color temperature are illustrated in Table 2. Not that the axes without units mean normalized value.

The gamma curve effect [19] on the testing image can be observed in Fig. 8. Gamma 2.2 will make the original sRGB value higher and the image becomes brighter. Usually the gamma curve with gamma 2.2 is realized by gamma 2.4, due to the digital implementation. Gamma 2.4 is slightly modified by the linear scaling and offsetting [16]. In the end, they were
almost the same, as verified in Fig.9.

C. Calculation of Taylor expansion with LUT

The global Taylor approximation was plotted as shown in Fig. 10, to compare the nonlinear function. The Taylor approximation is quite match with the original function. Since the Taylor approximation is a linear function with first-order approximation, LUT can calculate the function very easily. By using the proposed Taylor expansion for LUT in SA-C, the required memory is very small. As shown in Table 3, there are two types for LUT: one is 100-point LUT, the other one is 20-point LUT. The 100-point LUT requires 100 words in memory space, while 20-point LUT only requires 20 words in memory space.

Of course, the required memory space can be very small for the two types. LUT methods can speed up the gamma curve mapping. The accuracy for both of them is acceptable. The look-up table often requires numerous memory space such as M points to build up a mapping function in the way of point-to-point. In the M points, we selected only fewer N points (N<<M) to reduce the memory requirement.

Results show that accuracy depends on the X_i value as shown in Eq. (36). The relation for accuracy versus X_i can be easily observed from Fig. 10. For the X_i interval, the accuracy is proved to be a constant c_i and bounded by this constant. c_i is associated with the given section points X_i, section distance h and gamma value γ_{ape}. Also, c_i is the accuracy for the linear approximation. For the adjacent X_i, interval, the accuracy was also proved to be a constant c_{i+1} and bounded. Therefore, the totally accuracy for the global Taylor approximation can be the linear combination of the c_i and c_{i+1}.

VI. DISCUSSION

1) The proposed algorithm is not the same as floating point operation. Data-dependency decomposition is used to speedup the calculation in the pre-processing stage. Only right-shift and left-shift operations are required. Unlike the floating-point operation that processes in SAC compiler, the proposed algorithm for the color space transformation is actually a pre-processing stage before SAC compiler. No floating-type data is declared in the algorithm. Only integer-type data is used. The processing for the float-point data is different from the one for the integer operation.

2) The definition of the proposed algorithm is clear. The proposed algorithm is a pre-processing before compiled by SAC compiler. The pre-processing is helpful for speedup the execution time. At the pre-processing stage, the nominal value and base value are separated independently to be suitable for parallel operation.

3) The K-table in section 2.6 is clearly defined in Table 1. From the precious section 2.5, the expressions for the X variable are derived in detail. Y and Z variables can be deduced in the same way.

4) Equation 10 defines the required base for the associated derivation. The base has to be larger than the maximum value of the three variable m_{11}, m_{12} and m_{13}.

5) The flow chart in Fig. 4 describes the data-dependency can be separately considered by a calculation machine. Left and right parts of the flow chart can execute independently. The four matrices are defined as shown in Fig. 4.

6) Each column of Table 1 compares the execution time under the same percent cycle and percent flip-flops. Since the color space transformation is modeled by matrix operation with real number. Floating point operation is required to perform the calculation. Before using SAC compiler, this paper develops the formulation to be suitable for fixed-point operation. The fixed-data type implementation is required in the pre-processing stage. Fixed-point data without proposed algorithm can not speedup the color space transformation.

VI. CONCLUSIONS

This paper has successfully proposed nominal decomposition algorithms for the color space transformation. The gamma curve mapping function was also decomposed into the structure suitable for the SA-C circuit-level parallel operation. With the DFG embedded in SA-C, the required VHDL codes can be further obtained. Results show that the objectives of the speedup can be achieved. The color space transformation for chromaticity was successfully implemented.

Taylor expansion combined with LUT was able to approximate the required gamma curve very well by using the 20-point and 100-point types. K-Table matrices were built up for the base change for the nominal decomposition. It is evident that the proposed algorithm can be very helpful to the color space transformation for the hardware-based interface card and software-based system driver.
Table 2: The XYZ and RGB value for the white-points for different standards calculated from the proposed algorithm

<table>
<thead>
<tr>
<th>Standard</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>R_8bit</th>
<th>G_8bit</th>
<th>B_8bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>D65</td>
<td>0.3127</td>
<td>0.329</td>
<td>0.3583</td>
<td>72</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>Gamma 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D65</td>
<td>0.3127</td>
<td>0.329</td>
<td>0.3583</td>
<td>148</td>
<td>155</td>
<td>154</td>
</tr>
<tr>
<td>Gamma 2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D50</td>
<td>0.3457</td>
<td>0.3505</td>
<td>0.3038</td>
<td>97</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>Gamma 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D50</td>
<td>0.3457</td>
<td>0.3505</td>
<td>0.3038</td>
<td>167</td>
<td>148</td>
<td>133</td>
</tr>
<tr>
<td>Gamma 2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D93</td>
<td>0.283</td>
<td>0.297</td>
<td>0.42</td>
<td>52</td>
<td>63</td>
<td>87</td>
</tr>
<tr>
<td>Gamma 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D93</td>
<td>0.283</td>
<td>0.297</td>
<td>0.42</td>
<td>131</td>
<td>141</td>
<td>160</td>
</tr>
<tr>
<td>Gamma 2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sRGB</td>
<td>0.313</td>
<td>0.329</td>
<td>0.358</td>
<td>72</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td>Gamma 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sRGB</td>
<td>0.313</td>
<td>0.329</td>
<td>0.358</td>
<td>148</td>
<td>147</td>
<td>146</td>
</tr>
<tr>
<td>Gamma 2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Gamma curve function mapping LUT using Taylor expansion

<table>
<thead>
<tr>
<th>Standard</th>
<th>20 point LUT</th>
<th>100 point LUT</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma 2.2</td>
<td>0.015</td>
<td>0.005</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7. The chromaticity diagram for Y-X and Z-X

Figure 8. Gamma curve modification under full-color YUV122 color system

Figure 9. (a) Gamma curve 2.2 and modified gamma 2.4 comparison (b) generated DFG representation

Figure 10. LUT illustrations using Taylor expansion for 20-point and 100-point LUT

(a) Gamma 1.0  
(b) Gamma 2.2

(a) Gamma curve with 20-point LUT

(b) Gamma curve with 100-point LUT

(c) Accuracy under 20-point LUT

(d) Accuracy under 100-point LUT

Figure 10. LUT illustrations using Taylor expansion for 20-point and 100-point LUT
ACKNOWLEDGMENT

Financial support from the National Science Council project is appreciated. The authors wish to thank Chi-Lin Tech. Inc. for providing the testing equipment. Thanks to Tzeng Tseng for typing this article enthusiastically.

REFERENCES


Jian-Long Kuo received a B.S. degree in Electrical Engineering from National Sun Yat-Sen University at Kaoh-Siung in 1991, and a Ph.D. degree at the same institute in 1995. He is now an associate professor of the Institute of System Information and Control, Mechanical and Automation Engineering, National Kaohsiung First University of Science and Technology, NKFUST, Nan-Tze, Kaohsiung, TAIWAN. He was a project leader and research scientist at Electro-technology Division, Energy & Resources Labs. of ITRI, Hsin-Chu, TAIWAN. His research interests include FPGA/CPLD analog and digital circuits design and 3C consumer electronic system integration technology in audio and video display systems. The Chinese Institute of Engineers awarded him the Prize for Excellent Engineering Student in June 1991. Three national patents and two excellent scholastic prizes were awarded to him. He is now a member of IEICE, IEEE, IEE, and PHI-TOU-PHI.