A High-Efficient and Cost-Effective LCD Signal Processor

Tze-Yun Sung 1  Chun-Wang Yu 1  Yaw-Shih Shieh 1  Hsi-Chin Hsin 2

1 Department of Microelectronics Engineering, Chung Hua University, Hsinchu, Taiwan 300-12, R.O.C.  
2 Department of Computer Science and Information Engineering, Formosa University, Hu-Wei, Taiwan 632-08

Abstract

This paper investigates the system of LCD monitor and TV signal processor. It is shown that the processor provides scalar, brightness and contrast adjuster, Gamma correction and dithering. The algorithms are discussed and implemented by Verilog hardware description language. The key hard codes are displayed. The signal processor for monitor and TV achieves real-time and high performance. The system is evaluated and implemented on FPGA (Xilinx-XC2V6000).

Keywords: LCD, scalar, brightness and contrast adjuster, Gamma correction, dithering.

1. Introduction

In this paper, we discuss the algorithms of image scalar, brightness, contrast adjuster, Gamma correction and dithering. The system is integrated and the performance of the signal processor achieves real-time, area and power consumption under FPGA implementation are also optimized [1], [2].

The system diagram of control circuits of LCD monitor and TV is shown in Figure 1, the system is composed of analog to digit converter, video decoder and signal processor, which includes controller and microcontroller [3]. In the initial state, the microcontroller reads data from flash memory and writes commands and tables into register file of controller [4].

So far, we rarely find papers and documents to describe and discuss the algorithms and system architectures of LCD and TV signal processor. But the industry of LCD has been growing rapidly, and has become the main stream of high technology in Taiwan. This paper present the key components of the monitor and TV signal processor and propose the algorithms and the circuits implemented by Verilog codes [2].

The remainder of this paper is organized as follows. Section II presents the image scalar algorithm, and displays the key Verilog code, which describes the circuits of the scalar algorithm. In Section III, the algorithm and implementation of the brightness and contrast adjuster are proposed. Section IV applies the algorithm of Gamma correction to correct the input signal, and display true color. Section V presents the algorithm of dithering and implementation of the algorithm. Finally, the conclusions are given.

2. The Algorithm and Implementation of Image Scalar

The purpose of image scalar is to enlarge and reduce the input image. The algorithm of image scalar avoids the distortion, when the scaling operation is performing. In this section, the algorithm will be described in detail.

In this scalar, we may not use the special purpose digital signal processor [3]. The hardware is implemented by simple multipliers, and memory is saved. Hence, the hardware is saved, and implemented easily. Figure 3 shows the data flow of LCD monitor and TV signal processor. In Figure 2, the INP module recognizes the input signals and transfer to YUV 4:2:2 mode, which presents in video, S-video and HDTV. The SC module performs the function of image scalar. The CSC (color space converter) module transfers video signals to RGB mode. The transformation equation of CSC module is implemented by simple multipliers, adders and shifters [2], [3], [4].

In SC system, sc_fifo and sc_fifo4 module provide the memories for Y-buffer and X-buffer. All of input signal are stored in Y-buffer. The Y-buffer requires the memory capacity of 1024 (pixels) · 3-bytes · 8 (Lines), it saves hardware cost and performs the functions [4].

2.1. The Data Retrieval and Computation

The new horizontal line (row) is created by four-upper and lower rows. The data are stored and computed in Y-buffer. The Y-buffer requires the memory capacity of 1024 (pixels) · 3-bytes · 8 (Lines), it saves hardware cost and performs the functions [4].
where $p_{m1}$ and $p_{p2}$ are presented as color level (0~255).

According to this algorithm, the hardware is saved and the digital signal processor is not required, the low hardware cost is achieved. The key codes for performing the algorithm are described as follows:

2.2. The Algorithm of Expanding and Reducing Lines

In Y-buffer, we perform expand and reduce operation by using data set of horizontal scan lines to stay or jump. For example, if the data of original image is 800 horizontal lines, we may expand to 1024 lines; the special algorithm is developed and explained as follows:

- #include <iostream>
- #include <fstream>
- #include <cmath>

```c
// C3
assign my_m1 = p_m1[23:16] * coeff[34:27]; // C3
assign my_z0 = p_z0[23:16] * coeff[25:18]; // C2
assign my_p1 = p_p1[23:16] * coeff[16:9]; // C1
assign my_p2 = p_p2[23:16] * coeff[7:0]; // C0
assign ry = myr + { 8'b0, myr[16:8] } + 8'h80;
```

When we reduce lines and get the inc_p as follows:

```c
inc_p = \{ 1, 1, 1, 2, 1, 1, 1, 2, 1, 1 \}...
```

The algorithm is described as follows:

2.3. The Parameters for Expanding Pixels and Lines

In order to adjust the color and smooth the color distribution, the parameters or weighing factors must be found [3]. The data for computing the parameters are presented as follows:

```c
Pixel value = \frac{C3 \times p_{m1}}{256} + \frac{C2 \times p_{z0}}{256} + \frac{C1 \times p_{p1}}{256} + \frac{C0 \times p_{p2}}{256} + 0.5
```

where $p_{m1}$ and $p_{p2}$ are presented as color level (0~255).

According to the algorithm, the hardware is saved and the digital signal processor is not required, the low hardware cost is achieved. The key codes for performing the algorithm are described as follows:
$$(-16 \times 160 + 144 \times 140 + 143 \times 120 - 16 \times 100) \div 256 = 130$$

New line 40:
$$(-6 \times 160 + 58 \times 140 + 221 \times 120 -16 \times 100) \div 256 = 125$$

New line 41:
$$(0 \times 140 + 255 \times 120 +0 \times 100 - 0 \times 80) \div 256 = 120$$

Hence, the line expansions are achieved, and feeling of observation is much better.

2.4. The Algorithm of Expanding and Reducing Pixels

In X-buffer, the data of pixel are computed for expanding or reducing pixels. The algorithm of expanding pixels is described as follows:

where L20’ new is a new horizontal line intermediately, and has not expanded pixels yet. New Line 20 has been expanded by pixels.

3. Implementations of Brightness and Contrast Adjuster

The input signal is RGB, three colors are performed by brightness and contrast adjusting separately. Finally, three colors are performed an image together.

The value of brightness is defined as -128 ~ +127. 0 is default value. The brightness presents relative level. The level of brightness is defined as 0~199%. The default level is 100% [5], [6]. The computation of brightness is defined as

$$p \times 2^{8} / 128 = p \times 255 / 128 \div p \times 199\%.$$  

The algorithm is described by Verilog code, which is displayed as follows:

First, the brightness process performs the input image; the key code is displayed as follows:

$$g = g_{\text{in}} \times \text{con}_{g_{\text{in}}}$$
$$b = b_{\text{in}} \times \text{con}_{b_{\text{in}}}$$
$$r = r_{\text{in}} \times \text{con}_{r_{\text{in}}}$$

$$g_{\text{con}} = g[15:7] + g[6]$$
$$b_{\text{con}} = b[15:7] + b[6]$$
$$r_{\text{con}} = r[15:7] + r[6]$$

where $$g_{\text{in}} = p[23:16]$$, $$b_{\text{in}} = p[15:8]$$ and $$r_{\text{in}} = p[7:0]$$

Finally, the output image is performed by the following Verilog codes:

$$\text{bright}_g_{\text{in}} = \{\text{reg}_\text{bright}_g[7], \text{reg}_\text{bright}_g[7], \text{reg}_\text{bright}_g\}$$
$$\text{bright}_b_{\text{in}} = \{\text{reg}_\text{bright}_b[7], \text{reg}_\text{bright}_b[7], \text{reg}_\text{bright}_b\}$$

basic computation equation

$$R_{\text{out}} = R_{\text{in}} \times \text{Contrast} + \text{Bright}$$

$$= 170 \times 160 / 128 + 30$$

$$= 243.5$$

4. The Algorithm and Hardware Implementation of Gamma Correction

The intensity of display related with input signal is nonlinear, which is presented function of $$x$$, and x is input signal [5]. The equation is presented as

$$\text{Viewing}_\text{gamma} = \frac{1}{\text{Image}_\text{gamma}} \times \text{LUT}_\text{gamma} \times \text{Display}_\text{gamma}$$

where Image_gamma is defined as

$$\gamma_{\text{NTSC}} = 2.2, \gamma_{\text{PAL}} = 2.8, \gamma_{\text{RGB}} = 1$$ and $$\gamma_{\text{MAC}} = 1.8.$$  

and Display_gamma is defined as

$$\gamma_{\text{CRT}} = 2.5, \gamma_{\text{LCD}} = 1.6.$$  

The LCD transfer function in physics is shown in below:

5. The Algorithm and Hardware Implementation of Dithering

The color level of input signal is 8-bit, but the Gamma correction upgrades color level to 10-bit. But the output signal requires the color level of 6-bit or 8-bit

<table>
<thead>
<tr>
<th>Input</th>
<th>Hardware Implementation</th>
<th>Theoretical Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C = 160</td>
<td>B = 30</td>
<td>Rout = 243</td>
</tr>
<tr>
<td>Rin = 170</td>
<td>Rout = 243</td>
<td>Rout = 244</td>
</tr>
</tbody>
</table>
and the color distortion may occur, such as contour and color block. In this section, we downgrade the color distortion using dithering method, which simulates high quality image and eliminates contour and color block. In this signal processor, we choose the Bayer matrix to perform the dithering [1], [5], [6]. The Bayer matrix is represented as

\[
D_{2n} = \begin{bmatrix}
4D_n & 4D_n+2U_n \\
4D_n+3U_n & 4D_n+U_n
\end{bmatrix}
\]

where \(U_n\) is a \(n\times n\) unit matrix, \(n \neq 0\).

The \(D_2\) of 2 \(\mathbb{Z}\) and \(D_4\) of 4 \(\mathbb{Z}\) Bayer matrix are represented as follows:

\[
D_2 = \begin{bmatrix}
0 & 2 \\
3 & 1
\end{bmatrix}
\]

\[
D_4 = \begin{bmatrix}
12 & 8 & 2 & 10 \\
14 & 4 & 14 & 6 \\
11 & 11 & 1 & 9 \\
7 & 7 & 13 & 5
\end{bmatrix}
\]

where the threshold values are shown in the matrices. Hence, the color space can simulate high quality image by using low color level.

In vision color prospect, the difference of color level for human eyes is 1%. When 10-bit (0.1%) color level downgrades to 8-bit (0.4%), the difference of color level is less than 1%, the dithering process is not required. But we downgrades the color level to 6-bit (1.6%), the dithering process is required [5], [6].

6. Conclusions

In this paper, we use Verilog hardware description language (HDL) to implement the system. The system is partitioned into some modules. The architecture is easily debugged, and the modules are acted as IP, which can be reused. The human interface performs user friendly and full functions [7], [8]. The LCD monitor/TV signal processor is implemented; the low power and small area are taken into considerations.

References