Low-Energy Pixel Approximation for DVI-Based LCD Interfaces

Andi Nurrachmat     Enrico Macii     Massimo Poncino
Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, ITALY

Abstract—Several options are available for the approximating of color images in hardware, both in terms of the type of transformation (e.g., quantization, dithering) as well as in terms of where the approximation takes place (e.g., graphics controller, frame buffer, LCD controller). In this work, we propose a color approximation approach, orthogonal to other color simplification schemes, which is done during the digital transmission of the color data to the LCD.

The proposed technique targets a specific digital standard (namely, DVI) and its serial protocols TMDS, to provide a 45–66% savings in the number of bit transitions on the LCD bus (which translate to a corresponding saving of energy), depending on the allowed degradation of image quality.

I. INTRODUCTION

High-definition color LCD displays are known to consume a significant amount of power: measured data show that components related to the LCD (frame buffer, buses, LCD controller, and LCD panel) may consume more than 70% of the total power [1].

LCD buses, which are the subject of our work, have very large capacitances, due to the length of the connections, and account today for about 10% of total LCD subsystem power, a figure that is projected to increase with the introduction of display technologies such as organic displays (OLEDs) that do not require a backlight to function. Modern LCD interfaces are digital; therefore, techniques for minimizing the switching activity caused by the data sent on the bus [2] appear as a feasible solution for reducing the energy consumption. However, such techniques, devised for parallel buses, are not applicable to LCD buses, which are based on serial protocols.

The issue of energy-efficient transmission on digital LCD interfaces has been first analyzed in [10], [11], where, based on the correlation between adjacent pixel values, proper encoding of pixel differences are transmitted. The solution proposed by [12] generalizes the notion of encoding by adding an extra degree of freedom: It builds energy-efficient codes by allowing pixel values to be approximated. This lossy encoding was devised for a standard digital interface (namely, OpenLDI [3]) that transmits plain pixel values without additional encoding. In this work, we complement the solution of [12] by proposing a lossy encoding for LCD buses suitable for digital interfaces that explicitly encode pixel data, such as the DVI standard [4].

Our approximation scheme is applicable in conjunction with other color approximation schemes that may take place within the graphic controller, the frame buffer or the LCD itself: Whatever image is transmitted on the bus, our color approximation is always possible.

Experimental results show average energy savings in the range 10-30%, depending on the tolerated quality level.

II. BACKGROUND AND RELATED WORK

A. Digital Display Interfaces: DVI and TMDS

Several interface standards have been proposed by various organizations, such as Plug and Display, Digital Flat Panel (DFP), OpenLDI [3], DVI [4], and GVIF [6]. They define in a modular manner the various levels of the communications, i.e., the electrical and physical details as well as the organization of the transmitted data (i.e., the pixels). All the data-link protocols used by the various standards are serial, to allow low interference and low pin count. Although no standard is universally accepted, DVI [4] has become the most widespread digital LCD interface. DVI is based on the serial protocol TMDS (Transition-Minimized Differential Signaling).

TMDS encodes each 8-bit pixel data using transition encoding, (i.e., by encoding bit transitions rather than actual values), adding two control bits, and eventually serializes it into a 10-bit value. Details about the encoding are given in the next section. What matters here is the emphasis on the serial nature of the communication. Figure 1 shows the conceptual timing of a Single-Link TMDS transmission. \( T_{Y_i} \) denotes the \( i \)-th bit of the encoded value, for each of the three RGB channels \( Y = \{ R, G, B \} \), while \( C_{Y_i} \)'s denotes the control bits for the three channels. The pixel clock ranges from 25 to 165 MHz.

\[
\begin{align*}
\text{Clk} & \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \\
C_0 & \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \\
C_1 & \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \\
C_2 & \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0) \quad (t_0)
\end{align*}
\]

Fig. 1. 24-bit, Single-Link, XOR-based TMDS.

B. Energy-Efficient LCD Bus Transmission

LCD protocols do not provide options for energy-efficient transmission. Most techniques for low-power LCDs have focused on circuit-level solutions either in the design of individual components of the display systems ([7], [8]), to control mechanisms inside the LCD controller ([1], [9]).

Recent works have studied the energy-efficient data transmission on the digital interface ([10], [11]), by properly encoding
III. TERMINOLOGY AND NOTATION

Let $X = (x_0, \ldots, x_{N-1})$ the $N = 8$ bit pattern representing the generic pixel value, and $T = TMDS(X) = (t_0, \ldots, t_{M-1})$ the $M = 10$ bit TMDS code corresponding to pixel $X$. According to the TMDS standard [4], the $t_i$'s are defined as:

$$
t_0 = x_0
$$

$$
t_1 = t_0 \oplus x_1 = x_0 \oplus x_1
$$

$$
\vdots
$$

$$
t_i = t_{i-1} \oplus x_i = x_0 \oplus \ldots \oplus x_i
$$

$$
(1)
$$

$$
t_M = 1 \text{ if } \odot = \text{XOR}; 0 \text{ if } \odot = \text{XNOR}
$$

$$
t_9 = \text{DC balancing bit}
$$

The choice of XOR vs. XNOR is dictated by the analysis of the 1’s count in $X$, so that the encoding yields a naturally 0-1 balanced codeword [4].

We will also denote with $IWT(X)$ (Intra-Word Transition) the function that returns the number of bit toggles of a binary pattern $X$, and by $Ones(X)$ the 1’s count of pattern $X$.

IV. COLOR APPROXIMATION FOR TMDS

Our objective is to obtain a 10-bit TMDS codeword that is a minimal-IWT approximation of the original 8-bit pixel value, where the approximation is specified as a maximum tolerated error for each pixel value $X$. The approximation can be done following two different approaches, according to when the approximation is done:

1) Directly manipulate TMDS codes while doing the encoding so as to generate low-IWT patterns;

2) Decouple approximation and TMDS encoding.

Notice that this issue is irrelevant in approximation schemes that consider standards that do not encode pixel values (e.g., as in [12] for OpenLDI/LVDS).

Technical issues, however, make in practice only the second scheme viable. First, the TMDS standard provides for a set of reserved patterns that cannot be used as regular codewords: only 460 10-bit patterns are allowed out of the 1024 possible [4]. Another reason that prevents the first solution is the need for TMDS decoding at the receiver’s end. Generating an artificially modified TMDS code may result in an unpredictable decoded pixel value.

Therefore, we resort to the operation flow of Figure 2: The pixel value $X$ is first approximated into a new value $X^*$, using the maximum error as a constraint; $X^*$, when TMDS-encoded, will then result in a low-IWT 10-bit pattern. Notice that we approximate pixel values, but are the TMDS-encoded words that are sent on the bus; therefore, our approximation requires some form of “prediction” of the IWT of a TMDS code, given the knowledge of the pixel value $X$.

In principle, the use of this prediction is not strictly necessary, since we could (i) apply TMDS to exactly compute the IWT, (ii) use this value to drive the approximation, and (iii) encode the so-obtained value. However, the scheme of Figure 2, based on an estimate of $IWT(TMDS(X))$ has two main advantages: (i) it is modular, that is, the approximation and TMDS steps are separated, and (ii) lends itself to a scheme where $\delta$ is not fixed.

A. Estimating the IWT of a TMDS Code

The IWT for a given TMDS pattern $T$ is defined as

$$
IWT(T) = \sum_{i=0}^{9} (t_i + t_{i-1}) = (t_0 + t_1) + \ldots + (t_8 + t_9)
$$

where $\Sigma$ denotes arithmetic sum, and each term is considered as an integer quantity.

The IWT can be expressed in terms of the $x_i$ variables by using Equation 1. In this way, however, it is possible to express the IWT only for the 8 LSBs (the “payload”); the DC balancing bit is not defined in terms of the $x_i$’s, while it is easier to treat the XOR/XNOR bit separately.

Assuming first XOR as TMDS operator, the IWT for the “payload” portion of the TMDS code $(t_0, \ldots, t_7)$ is:

$$
IWT^{XOR}(T) = 0 \oplus (x_0 \oplus x_1) + \ldots + (x_0 \oplus x_1) \oplus (x_0 \oplus x_1) \oplus x_2) + \ldots
$$

The above formula, using the associative property of the XOR operator, and the identity $X \oplus X = 0$, becomes:

$$
IWT^{XOR}(T) = (x_1 + x_2 + \ldots + x_7) = \sum_{i=1}^{7} x_i
$$

where ‘+’ denotes arithmetic sum, and each $x_i$ is considered as an integer quantity. The expression says that the IWT of (the payload of) a TMDS codeword is equal to the 1’s count of the original value $X$, starting from its second LSB ($x_1$).

When XNOR is used, with similar simplification, we obtain:

$$
IWT^{XNOR}(T) = \sum_{i=1}^{7} x_i'
$$

4338
In this case, the IWT of a TMDS codeword is equal to the 0's count of the pixel value, starting from the second LSB ($x_1$). The two formulas can be then modified to include the IWT due to the XOR/XNOR bit as follows:

$$IWT^{\text{XOR}}(T) = \sum_{i=1}^{7} (x_i) + \text{even}(\text{Ones}(X))$$
$$IWT^{\text{XNOR}}(T) = \sum_{i=1}^{7} (x_i)' + \text{even}(\text{Ones}(X))$$

(2)

where \text{even}() evaluates to '1' if its argument is an even value. The IWT due to the DC balancing bit cannot be accounted for because it is context-dependent; it is determined by the history of the previous patterns and is available only after the TMDS codeword is obtained. Therefore, we will neglect the IWT due to the ($t_s,t_0$) bits. The following example demonstrates how the above formulas can be used to predict the IWT of a TMDS code from the pixel value.

**Example 1:** Consider the following 8-bit value $X = 01000001$. $X$ has two 1's, thus TMDS uses XOR, to obtain $T = \text{TMDS}(X) = 0111000011$ (MSB first). The MSB (DC balancing bit) is assumed to be '0', whereas the second MSB at '1' signals the use of XOR.

The IWT($T$) of is 2 (not counting the IWT due to the DC balancing bit), equivalent to the number of '1's in $X$ starting from bit 1 (the underlined bits).

**B. Approximation Algorithm**

Now that we have a way to express the IWT of TMDS code from a pixel value, we can formulate the pixel approximation algorithm as follows: *Given a maximum pixel error $\delta$, approximate a value $X$ into $X^*$, such that $|X - X^*| \leq \delta$ and $IWT(T^*) < IWT(T)$ is the smallest possible, where $T = \text{TMDS}(X)$ and $T^* = \text{TMDS}(X^*)$.*

A straightforward implementation of such algorithms works as follows. In the following, $X$ denotes the generic pixel to be approximated, $T$ its corresponding TMDS code, and $X^*$ the approximate pixel value.

1. Sort all $N$-bit patterns in non-decreasing order of IWT of the corresponding TMDS codeword;
2. Based on the inspection of $X$, determine whether XOR or XNOR is used;
3. Choose $X^*$ such that $|X - X^*| \leq \delta$ and $IWT(T^*) < IWT(T)$ is the smallest possible.

According to Equation 2, such code is the one with minimal zero (XNOR) or one (XOR) count and whose distance does not exceed $\delta$.

In terms of hardware implementation, the approximation values can be stored in a table, indexed by the value of $X$.

We call this direct implementation static, because the error is (i) defined upfront and (ii) always applied to every pixel value without exceptions. For this reason, we can build a table whose content does not change over time. Under this static scenario, even the IWT estimation formulation is immaterial: Since errors are fixed, $IWT(\text{TMDS}(X))$ for a given $X$ can be computed from the definition without resorting to estimates.

**C. Dynamic Color Approximation**

When $\delta$ is not fixed upfront, but it is allowed to change as a result of different external conditions (e.g., low battery status), a dynamic implementation, i.e., where the approximation function explicitly depends on $\delta$, is desirable.

The estimation of the IWT of a TMDS code is now essential (in particular, the dependence of the IWT on the 1's (or 0's) count of the original pixel value) since it allows to devise a parametric implementation of the pixel approximation algorithm. However, due to strict performance and energy constraints for the TMDS codec chain, we need to minimize the hardware overhead of the implementation, and we will discuss a solution that is sub-optimal with respect to the scheme based on a lookup table.

Since the IWT of the TMDS code sent on the bus is related to the 1's/0's count of the original pixel value, an approximation that is tunable with respect to $\delta$ can be based on the reduction of the 1's or 0's count.

Let consider the XOR case, for simplicity. Here, IWT is proportional to the 1's count in the pixel value. Therefore, approximation can be easily done by setting some of the LSBs to 0, as a way to reduce 1's count. How many bits to reset depends on $\delta$. Notice that this scheme is suboptimal with respect to the formulation of Section IV-A because it does not account for the additional term even() which would require too much logic for implementation.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>Bits that can be set/reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2</td>
<td></td>
</tr>
<tr>
<td>3,...,6</td>
<td>$x_1$</td>
</tr>
<tr>
<td>7,...,14</td>
<td>$x_1,x_2$</td>
</tr>
<tr>
<td>15,...,30</td>
<td>$x_1,x_2,x_3$</td>
</tr>
<tr>
<td>31,...,64</td>
<td>$x_1,x_2,x_3,x_4$</td>
</tr>
<tr>
<td>65,...,128</td>
<td>$x_1,x_2,x_3,x_4,x_5$</td>
</tr>
<tr>
<td>129,...,254</td>
<td>$x_1,x_2,x_3,x_4,x_5,x_6$</td>
</tr>
<tr>
<td>255,256</td>
<td>$x_1,x_2,x_3,x_4,x_5,x_6,x_7$</td>
</tr>
</tbody>
</table>

**TABLE I** APPROXIMATION OVER THE RANGE OF $\delta$.

A simplified pixel approximation algorithm is one that, given a value of $\delta$, determines how many bits can be reset (XOR case) or set (XNOR case). Since we do not consider the LSB, which does not contribute to the IWT, setting or resetting $k$ bits (from bit 1 to bit $k$) in a binary pattern causes a worst case error of $\pm \sum_{i=2}^{k} = \pm (2^k - 1)$. We want this error to be smaller than $\delta$, i.e., $|2^k - 2| \leq \delta$, resulting in $k \leq \log_2(\delta + 2) - 1$, i.e., $k = \lfloor \log_2(\delta + 2) - 1 \rfloor$ (since only integer amounts of bits are possible). The number of bits can be set/reset for various values of $\delta$ are shown in Table I. Notice that $x_0$ is not considered since it does not contribute to the IWT.

Typical values of $\delta$ are in the range 15–50, corresponding to an image quality between 90% and 80% [12]. Therefore, only the first five rows of the table are used ($x_3$ is modified only for $\delta \geq 62$). From the table we can derive the equations of the LSBs of $X^*$ as a function of $\delta$. Only bits $x_1,x_3,x_4$ are modified (i.e., set or reset), as shown in Table II. The LSB $x_0$ is again left untouched since it does not contribute to the IWT.
TABLE II  APPROXIMATE PIXEL BITS FUNCTIONS.

<table>
<thead>
<tr>
<th>Function</th>
<th>XOR</th>
<th>XNOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0 &amp; x1 &gt; 3</td>
<td>1 &amp; x1 &gt; 3</td>
</tr>
<tr>
<td>x2</td>
<td>0 &amp; x2 &gt; 7</td>
<td>1 &amp; x2 &gt; 7</td>
</tr>
<tr>
<td>x3</td>
<td>0 &amp; x3 &gt; 15</td>
<td>1 &amp; x3 &gt; 15</td>
</tr>
<tr>
<td>x4</td>
<td>0 &amp; x4 &gt; 31</td>
<td>1 &amp; x4 &gt; 31</td>
</tr>
<tr>
<td>x5</td>
<td>0 &amp; x5 &gt; 63</td>
<td>1 &amp; x5 &gt; 63</td>
</tr>
<tr>
<td>x6</td>
<td>0 &amp; x6 &gt; 95</td>
<td>1 &amp; x6 &gt; 95</td>
</tr>
<tr>
<td>x7</td>
<td>0 &amp; x7 &gt; 127</td>
<td>1 &amp; x7 &gt; 127</td>
</tr>
</tbody>
</table>

The implementation of this dynamic scheme is quite simple and requires 29 library cells, with a delay of 0.6 ns and a power consumption of 67 uW, at the maximum DVI frequency of 165 MHz, for a 0.13 \( \mu \)m technology. Notice that the power consumption of a typical TMDS encoder is around a few hundred of mW [11]. Further details are omitted due to the limited space.

V. EXPERIMENTAL RESULTS

A. Energy and Quality Metrics

Concerning energy, the number of total transitions on each of the encoded RGB channels C0, C1, C2 (Figure 1) has been used as a metric for the total dissipated energy [2]. For the quality metric, we use the simple root-mean-square error (RMSE) metric in the 3-D RGB space. Although more sophisticated metrics are available, thanks to the type of approximation we consider (small perturbation of individual pixel values), RMSE exhibits an extremely good correlation with the widely used Universal Quality Index [13], which also accounts for the spatial and tonal features of images. As a quantitative measure, quality values around 80-85% represent the maximum tolerable approximation.

B. Experimental Data

We have applied the proposed color approximation schemes on all the color images contained in the SII database [5]. Figure 3 shows the transition savings (average over the three TMDS channels) for different values of quality, for the static TMDS approximation. The savings account for the transitions due to the DC balancing bits as well as those across consecutive pixels, which cannot be controlled by our encoding (the intrinsic efficiency of the code is therefore even higher).

The savings are quite significant even for high quality values (average 45% saving for 95% quality). The best compromise appears to be at 90%, for which a 53% average savings can be achieved. Figure 4 shows the same data, yet referred to the dynamic scheme. The loss in efficiency is non-negligible, (approximately a 30% difference with respect to the static scheme, independent of the quality). Also in this case, the best tradeoff appears to be for 90% quality (27% average saving).

VI. CONCLUSIONS

We have shown how approximating color values during their transmission on a digital LCD bus can be used to tradeoff image quality for a reduction of the energy consumed for the transmission. Our scheme, explicitly meant for LCD interfaces that encode pixel values, introduces two algorithms (for fixed and variable distortion levels, respectively) that allow to save from 45% to 66%, depending on level of tolerated image quality.

REFERENCES