Presented at: International display research conference (IDRC '08) ; November 3–6, 2008
(Monday – Thursday) University of Central Florida Orlando, Florida, USA

15.3: A Clipper-Free Algorithm for Efficient HW-Implementation of Local Dimming LED-Backlight

Marc Albrecht[1], Andreas Karrenbauer[2], Chihao Xu[1]
[1]Institute of Microelectronics, [2]Max-Planck-Institut für Informatik,
Saarland University, D-66123 Saarbrücken

Abstract: *In this paper, we present algorithms for the calculation of local dimming LED-Backlight. They assure clipper-free results and can be implemented at low HW-costs. The power saving is in average 38.7% compared to global dimming. For a XGA display 72K (RGB) logic gates are needed to implement this algorithm on chip.*

Keywords: LED-Backlight Algorithm, Local Dimming, Clipper-Free, Power Saving

1. Introduction

The advantages of LED-Backlight with local dimming in comparison to conventional CCFL backlight are well known by now. A detailed description of the state of the art of LED-Backlight is given in 1. Different approaches for calculating the backlight have been presented in publications, such as 2, 3, 4 and 5.

Our intention is to find an algorithmic solution for the backlight which is clipper-free and easy to execute, thus of low logics complexity.

First, we show the mathematical description of LED-Backlight. Then we present the algorithms that we have developed for this purpose. The hardware implementation is shown in chapter four. Chapter five includes the statistical results of our algorithm. Finally an outlook is given.

2. Description of LED-Backlight

The backlight luminance B observed at pixel (i,j) is determined by the sum of the attenuated luminance intensities I of the LEDs. Let L be the number of LEDs, then we have

$$B(i,j) = \sum_{k=1}^{L} \left[d_{ij}(k) \cdot I(k) \right],$$

where the coefficients $d_{ij}(k)$ model the attenuation of the light emitted from the k-th LED on its way to pixel (i,j), which mainly depends on the distance traveled. Since we drive the LEDs with pulse-width modulation (PWM) in a range in which the intensity scales linear with the duty cycle, we may write

$$B(i,j) = \sum_{k=1}^{L} \left[a_{ij}(k) \cdot x(k) \right].$$

$a_{ij}(k)$ is analog to $d_{ij}(k)$ and includes current amplitude of the LEDs which is proportional to the display luminance set by user. $x(k)$ *for* $k = 1, \ldots, L$ lies between 0 and 1 and determines the fraction of time in which the k-th LED shines with a fixed luminance. Clearly, the power consumption is proportional to the sum of duty cycles, that is

$$\sum_{k=1}^{L} x(k).$$

This is the objective we want to minimize. Furthermore, we only allow timings for the LEDs that permit a clipper-free solution, i.e. the luminance observed at each pixel is at least as required by the RGB values of the actual image. This yields a *Linear Program* 6 of the form

$$\min \left\{ \sum_{k=1}^{L} x(k) : \quad Ax \geq r, \quad 0 \leq x \leq 1 \right\},$$

where the matrix A is made of $a_{ij}(k)$ and captures the attenuation model of the backlights and r represents the RGB values of the given image. Hence, we have a constraint for each pixel and a variable bounded between 0 and 1 for each LED. A key observation that permits efficient algorithms to approximate the optimal solution in real-time is that the matrix A is *sparse*. That is, in a real setting, the number of LEDs that have an effect on a pixel is small, e.g. four in our case. We will exploit this fact in the next section to come up with efficient algorithms that minimize the power consumption by local dimming.

3. Algorithms for Local Dimming

Since the backlight LEDs are placed in a lattice structure behind the pixel plane, we have pixels that are affected only by one, two, or four LEDs. It is easy to see that the RGB value of a pixel, which only depends on one LED, determines a lower bound for the corresponding duty cycle. In this case, we can rewrite the respective constraint as

$$x(k) \geq \frac{r_{ij}}{a_{ij}(k)} \ .$$

This also applies to the pixels that depend on two or four LEDs when we set their respective duty cycles to 1, i.e. the lower bound for the k-th LED is determined by the maximum of

$$\frac{r_{ij} - \sum_{l \neq k} a_{ij}(l)}{a_{ij}(k)},$$

where the maximum ranges over all pixels *(i,j)* for which $a_{ij}(k) > 0$. Hence, in a first phase, we scan over the pixels to compute the lower bounds on the duty cycles.

Since the partial solution that we have after the first phase is usually not clipper-free, we have to perform a second pass to fulfill this constraint. For this task, we propose two methods. The first one produces nearly optimal results, but has a large memory overhead. The second one avoids this issue at the expense of slightly worse solutions.

The basic idea of the first method is to select in every iteration the pixel (i', j') with the greatest deficit, i.e. with the largest difference between required and actually observed luminance. Then we choose the LED with the highest effect on this pixel, i.e. the one with the greatest coefficient $a_{i'j'}(k)$. Let this LED be k'. Hence, if we increase $x(k')$ by

$$\Delta := \frac{r_{i'j'} - \sum_{k=1}^{L} a_{i'j'}(k) \cdot x(k)}{a_{i'j'}(k')},$$

the pixel (i', j') would receive enough light and would be satisfied. To make this algorithm a bit more sensitive, we actually do not increase by the entire amount but only by a fraction $\lambda \cdot \Delta$ for a fixed $\lambda \in (0,1)$. The algorithm terminates if no unsatisfied pixel remains and hence yields a clipper-free solution by design. Termination is guaranteed since we discretize all quantities and thereby increase in each iteration the duty cycle of one LED by at least one least significant bit.

However, selecting the most unsatisfied pixel does not come for free. We may use a well-known data structure called *priority queue* 6 which is of linear size. Clocks needed for insertion are logarithmic to the number of pixels, and querying the most unsatisfied pixel is possible in constant time. This method yields nearly optimal results in practice.

The main issue of this approach is the large memory overhead for the pixel data in the priority queue, even when simple structure like bucket queues are used. If local dimming is needed in a setting where power consumption has a much higher priority than the cost of RAM, then the algorithm described above is recommended.

If lower RAM size is preferred over few percents lower consumption is, we propose a second method to fit that need. Here, we scan over the pixels once in a fixed fashion and we increase the nearest LED such that the considered pixel gets satisfied (illustrated in Figure 1).

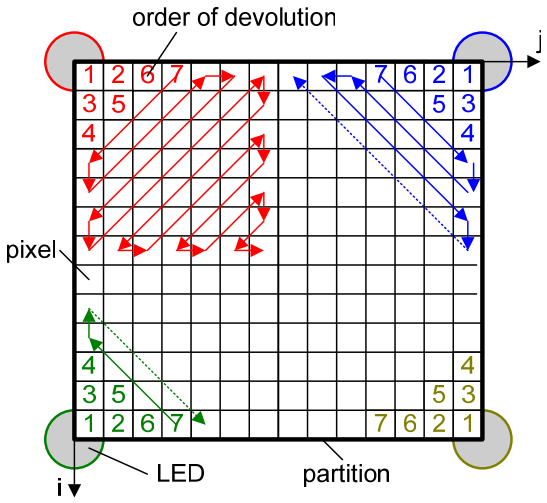

Figure 1: Order of devolution in a partition

The brightness of the pixels in each partition only depends on the four LEDs at the respective corners. Clearly, the pixels close to the corner are mainly influenced by that LED. Since at the beginning of the second phase, the intensities of the LEDs are at their lower bounds, we potentially underestimate their final contribution. However, since we start in the corners, the effect of this under-estimation is marginal since the influence decays with the distance. In each iteration, the power of the nearest LED of the considered pixel is increased such that the pixel gets satisfied. Hence, the under-estimation effect gets smaller over time. Thus, the simultaneously growing influence of the remote LEDs on the considered pixel remains uncritical.

But still, this method can be fooled to some extent by special instances because of the fixed scheme in which we consider the pixels. To get the best out of both worlds, it is of course possible to consider a hybrid approach. There, we determine a constant number of the most unsatisfied pixels during the first phase, use the same strategy as for the first method, and then continue in a fixed fashion as described before. Thus, we bound the memory overhead as we only store a constant fraction of the pixels for the intermediate step. We leave this idea for future work.

4. Hardware Implementation

The implementation result is shown in this section. Most of the operations are additions, subtractions and comparisons, while only few multiplications are used. Our main design goal for the implementation was to achieve low hardware costs. The following hardware configuration (Figure 2) was chosen.

Process	0.18 CMOS
Ram-Type	Single-Port
Data-Bus-Width	32 bit
Pixel-Resolution	8 bit
LED-Resolution	8 bit
Point-Spread-Function	11 bit
Image-Size	1024RGB x768
Number of LEDs	64RGB

Figure 2: HW configuration

The operating sequence of our HW design is shown in Figure 3. In the following the different phases of the implementation are described in detail.

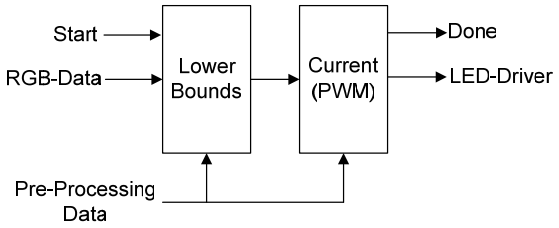

Figure 3: Operation sequence of the implementation

4.1 Pre-Processing

The values of the point spread function are determined by measurement. They are condensed and stored in the ROM (66Byte RGB). In the pre-processing phase these points are expanded by interpolation. The results of the pre-processing phase are stored in the RAM. The difference between the interpolated and the measured point-spread-function is negligible.

4.2 Calculation of the lower bounds and the final currents

For the calculation of the lower bounds as well as the calculation of the final currents, we have to look at each pixel once. To speed up the calculation for the lower bounds, we store additional information of 35.9kB in the ROM.

4.3 Hardware Complexity

The algorithm was coded in HDL. The logics complexity for RGB backlight is three times 72K gates. To process an image of size 1024x768x8bits, 2.49M clocks are needed. So in case of a processor frequency of 100 MHz, a frame rate of 40 frames per second can be achieved. This is sufficient for video application.

Additional memory is necessary to process the local LED luminance. A ROM of 3x66 Bytes is needed to store the condensed point-spread-function. The computation of the lower bounds requires 3x35.9kB SRAM.

The total chip area of this processor unit needs just a small fraction of the GDRAM size of 2360kB.

5. Statistical Results

Since the result depends on the image, a test-set of 100 RGB images (see Figure 4) was used to evaluate the effectiveness of our design.

Figure 4: Extraction of the test-set images

The important topic is the power saving due to the local dimming. For an image with maximal brightness in all pixels, the optimal solution for the point-spread-function we measured needs 42.93 for the sum of duty cycles (compared to 64, with a duty cycle of 1 for each LED). In average, we save 38.7% compared to this optimal global dimming. The power saving for our test-set is plotted in Figure 5 (blue).

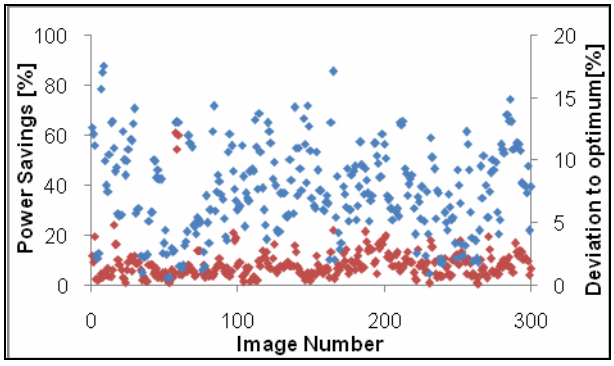

Figure 5: Power savings compared to global dimming in % (blue); Deviation of the implemented algorithm normalized to the optimal solution (red)

To prove that these results are close to the optimum, we used the linear programming software CPLEX 8 to compute the optimal solutions as benchmarks for the

results of our algorithm (red in Figure 5). The average deviation to the optimal solution is 1.6%.

The image in Figure 4 in the lower right corner marked by a red frame is the only outlier with a deviation of 12% from the optimal solution. This image demonstrates the limitation of our implemented algorithm. The relative deviation of the brightness after the lower bounds calculation is high compared to the final current calculation and there are at the same time strong transitions in the brightness. Even so, we achieve a power saving of 65% for this case.

Moreover, with the method proposed first, which we only implemented in C++ due to its large memory overhead, we achieve a solution that is only 0.3% off the optimum.

6. Outlook

In this paper we have presented two clipper-free algorithms for the image-dependent setting of LED backlight. In the hardware version we achieved a solution which is in average only 1.6% worse than the optimum. The average saving of power is 38.7%. On the implementation side we achieve a frame rate of 40 frames per second with three times 72K gates.

Still, there are some opportunities for future work. We will incorporate the intermediate step of the hybrid approach. We are working on an extension for HDR images and also on a solution that allows an efficient calculation of the TFT signals. Furthermore, we want to extend the algorithm and the implementation to sequential color.

A patent for the developed algorithms will be filed within the grace period.

7. References

1. Munisamy Anandan, “Progress of LED backlights for LCDs”, JSID2008, pp.287-310, 2008
2. Feng Li, et al., „Two approaches to derive LED driving signals for high-dynamic range LCD backlights“, JSID07, pp.989-996, 2007
3. Helge Seetzen, et al., “High Dynamic Range Display Systems”, ACM SIGGRAPH 2004
4. Hanfeng Chen, “Locally pixel-compensated backlight dimming on LED-backlit LCD TV”, JSID07, pp.981-988, 2007
5. Hanfeng Chen, “Backlight Local Dimming Algorithm for High Contrast LCD-TV”, Proc. of ASID06, pp.168-171, 2006
6. D.P. Bertsimas and J.H. Tsitsiklis. “An Introduction to Linear Optimization”. Athena Scientific, 1997.
7. Thomas H. Cormen, et al. “Introduction to Algorithms”, McGraw-Hill Higher Education, 2nd Edition, 2002
8. ILOG. CPLEX Homepage, July 2008, http://www.ilog.com.