A Backlight Power Management Framework for the Battery-Operated Multi-Media Systems

Hojun Shim
School of CSE
Seoul National University
Korea 151-010

Naehyuck Chang*
School of CSE
Seoul National University
Korea 151-010

Massoud Pedram
Department of EE
University of Southern California
USA CA 90089

Abstract

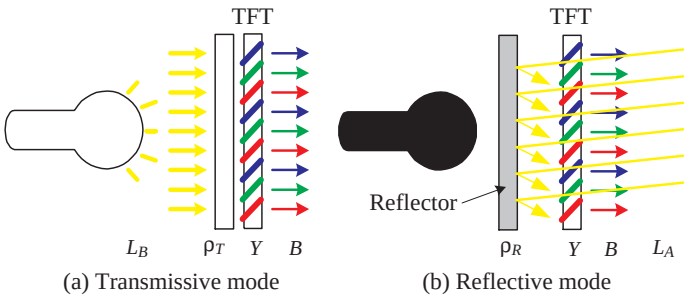
Dynamic Luminance Scaling (DLS) reduces the backlight power consumption of color TFT LCD while maintaining the image brightness or contrast. This paper presents extended DLS (EDLS) as a framework for the backlight power management of the transflective LCD panels. The EDLS framework switches between DLS and Dynamic Contrast Enhancement (DCE) depending on the panel mode, which may be transmissive or reflective, and the remaining battery energy. This paper analyzes the energy gain from the backlight and the overhead of EDLS in view of application transparency and hardware-software trade-offs, which visualize the systematic approach for the efficient EDLS implementation. As a result, we demonstrate a compact EDLS system that relies on approximate histogram construction, thereby, achieving a significant reduction in the area and power overhead of the EDLS system with negligible loss of the energy savings in the backlight system. Overall, the compact EDLS system results in an average of 25% power saving in the backlight system for still and moving images, without any modification of the existing applications.

1 Introduction

Color TFT (Thin-Film Transistor) LCD (Liquid Crystal Display) panels enable the battery-operated hand-held embedded systems to support full-featured multi-media and have replaced monochrome STN (Super Twisted Nematic) LCD panels for their applications. Most importantly, a TFT LCD panel does not illuminate itself but simply filters a backlight source, and this backlight is a primary power consumers in most systems. Thus, reducing the backlight power consumption is one of the primary ways to extend the battery life. Most existing power reduction techniques are based on power management during idle or slack times, and thus difficult to apply to the display panels, which have no idle time as long as they are turned on. Dimming or turning off the backlight results in noticeable degradation of the legibility of the LCD panel.

Recently, a power reduction technique was introduced which maintains either the brightness or the contrast of the LCD panel when the backlight is dimmed [1]. Appropriate image compensation techniques preserve either the brightness or the contrast of the original image, at the expense of minor image distortion, which does not seriously affect the legibility of the display. Because the CCFL (Cold Cathode Fluorescent Lamp) backlight is usually considered to have a slow response, a feedback control circuit was proposed to enable us to change the backlight luminance fast enough to support movie streams [2]. This technique is known as dynamic luminance scaling (DLS) of a backlight. Since human understanding of a display is affected by both the brightness of the LCD panel and the ambient luminance, another approach that has been introduced is backlight autoregulation in the context of the ambient luminance [3].

*Corresponding author



(a) Transmissive mode (b) Reflective mode

Figure 1: The transmissive mode and the reflective mode of a transflective LCD panel.

In this paper, we introduce a backlight power management framework for color TFT LCD panels considering the battery-operated multi-media applications. Since the image quality is as important as power consumption in portable multi-media applications, a quality power management scheme is critical. We extend DLS to cope with the transflective LCD panels that operate both with and without a backlight, depending on the remaining battery energy and the ambient luminance. This technique is suitable for the transflective LCD panels, which dominate the battery-operated electronic systems. The reason for popularity of the transflective LCDs is that the panel image is still visible without the backlight, even though the quality may be poor. We name our new backlight power management framework extended DLS (EDLS). It combines brightness compensation and contrast enhancement, which were previously separate techniques [1]. The EDLS approach compensates for loss of the brightness when there is a rich or moderate power budget and it compensates for loss of the contrast when power budget is poor.

We pursue an application-transparent approach with EDLS, intercepting the frame buffer contents and performing image processing. The need for modification of the existing applications used to limit the scope for its use because the previous DLS implementation requires source code modification of the existing multi-media application [2, 4]. Thus, an application-transparent DLS implementation, which does not require any modification of the existing operating systems, APIs or application programs, would be highly desirable for the promotion and wide adoption of DLS.

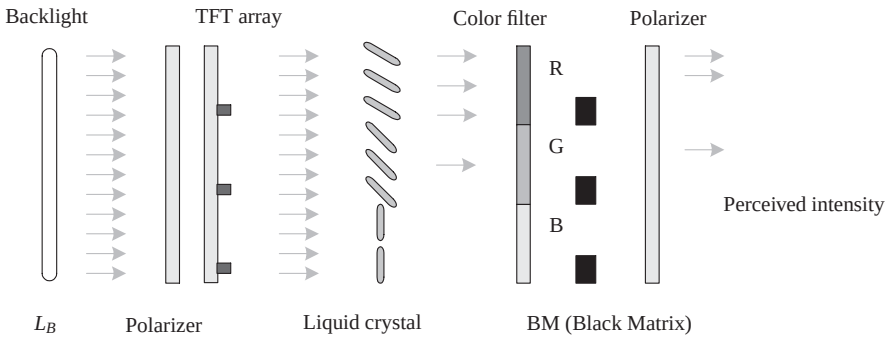
This paper provides a comprehensive analysis of hardware-software trade-offs for application-transparent EDLS. Finally, we demonstrate a perfectly application-transparent EDLS implementation. By considering all the trade-offs, we show that a pure hardware implementation of EDLS is beneficial with our system which has 16-bit color depth and 640×480 screen resolution. This application-transparent, pure hardware EDLS implementation exhibits a 25% power reduction while maintaining acceptable image quality and hardware overhead, in terms of both area and power.

2 Theory of operation

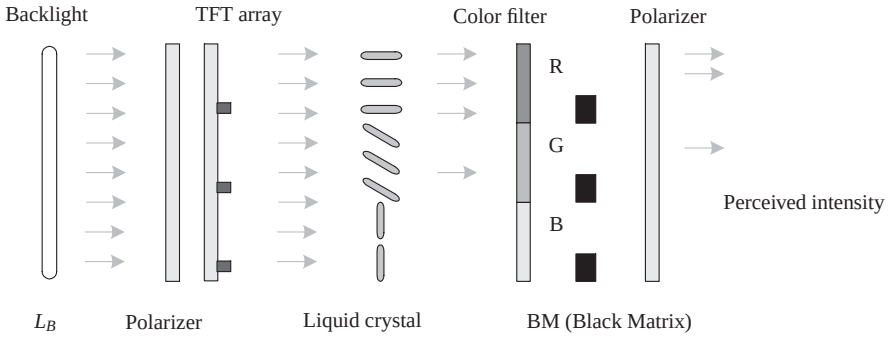
2.1 Transflective LCD

There are three kinds of TFT LCD panels. A transmissive LCD panel uses a backlight source; a reflective LCD panel uses the ambient light by means of a reflector without backlight; a transflective LCD panel is a hybrid of the transmissive and reflective LCD panels (Fig. 1). Transmissive and transflective LCD panels use very bright backlight sources, which emit more than 1000 cd/m^2. Unfortunately, the transmittance of the LCD, ρ_T, is relatively low and thus the resultant maximum luminance of the panel is usually less than 10% of that of the backlight source. This is why the backlight lamps have to be very powerful. When the backlight is turned off, a transflective LCD panel acts like a reflective LCD panel. The ambient light coming from the front side of the panel is reflected through the LCD like a backlight source. Theoretically, the backlight and the ambient light act in an additive manner. However, once the backlight is turned on, the transflective LCD panel effectively operates in the transmissive mode because the backlight source is generally much brighter than the ambient light.

The brightness in the transmissive mode is proportional to the product of the transmittance, ρ_T, and the backlight luminance, L_B. Similarly, the brightness in the reflective mode is proportional to the product of the reflectance, ρ_R, and the ambient luminance, L_A. The reflectance of the reflector is even lower than the transmittance of the LCD panel. In contrast with purely reflective LCD panels, transflective LCD panels cannot maintain high reflectance because their



(a) Original image with full backlight luminance



(b) DLS image with dimmed backlight luminance

Figure 2: Principle of DLS. Compensating the color values to have brighter image results in the similar perceived intensity with a dimmed backlight.

design favors transmissive operation. Moreover, the luminance of the ambient light is far weaker than that of the backlight source. Thus in general, the transmissive mode is significantly superior to the reflective mode in terms of both the brightness and the contrast. For example, the NEC6448BC33-50 LCD panel has a reflectance of 3.6% and a contrast ratio of 8:1 in the reflective mode. However, the same LCD panel exhibits a contrast ratio of 300:1 in the transmissive mode.

The principle of DLS and DCE is to reduce the luminance of the light source and allow more light to pass through the liquid crystal screen by enhancing the image luminance. This makes human eyes recognize the same intensity as shown in Fig. 2 [2]. While DLS preserves the original colors, DCE may severely change the original color in the pursuit of the higher contrast, and thus improve readability. Consequently, DCE must be considered as a very aggressive power management scheme for the transmissive LCD panels since it involves turning on the backlight, even if the power setting is low, to achieve a minimum level of readability. This separates the existing DLS and DCE strategies since they aim at improving the legibility of transmissive panels in very different ways.

The EDLS framework achieves a harmonious combination of DLS and DCE. Once the backlight is turned off due to low remaining battery level, the fidelity of the LCD image in terms of the brightness and the contrast is no longer a driving issue. Instead, readability becomes the most important criterion to be met by the display. DCE can be applied to transflective LCD panels, across the transmissive mode with extreme backlight dimming and the reflective mode when using ambient light.

2.2 The EDLS framework

The first DLS implementation was an application-embedded DLS system that was highly coupled with the application [2]. This DLS system can freely utilize information in its application, and thus the energy reduction from backlight dimming can be maximized with the minimum computational overhead. However, embedding a DLS technique in

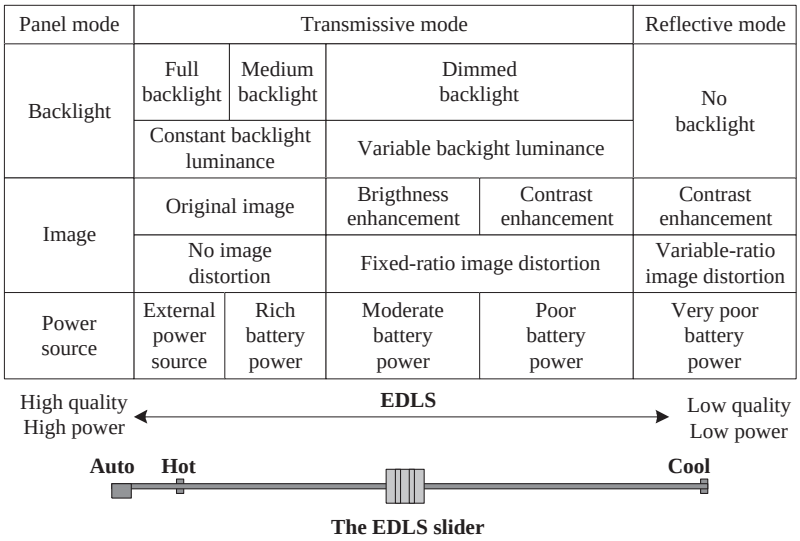


Figure 3: The EDLS framework.

the application requires a lot of *ad hoc* effort to port the DLS technique to all kinds of applications, one at a time. Furthermore, this approach may be completely infeasible when the application software is proprietary. The second DLS implementation introduced a standard API (Application Program Interface) at window management level [4]. One benefit of the standard API is a systematic approach to DLS porting, but this approach still requires same (although not as much as the first implementation) source code modification.

To build the EDLS framework, we borrow the principles of brightness compensation for DLS and contrast enhancement for DCE [4]. We determine the scaling factor for the backlight with upper and lower thresholds of T_H and T_L, which are in turn derived from the predefined saturation ratio, S^R. Let C denote the current color value. After brightness compensation, the new color value, C', is given by

$$C' = \min(2^n - 1, \frac{2^n - 1}{T_H} C), \tag{1}$$

where n is the color depth of each color component. And S_{BC} is the brightness compensation factor, which is equal to $\frac{2^n - 1}{T_H}$. Similarly, after contrast enhancement, the new color value, C', is given by

$$C' = \min(2^n - 1, \frac{2^n - 1}{T_H - T_L} \max(0, C - T_L)). \tag{2}$$

The contrast compensation factor, S_{CE}, is equal to $\frac{2^n - 1}{T_H - T_L}$. After the image compensation, we reduce the backlight luminance by $\frac{1}{S_{BC}}$ or by $\frac{1}{S_{CE}}$, depending on the EDLS mode.

Fig. 3 shows the EDLS framework. The EDLS interface is a simple slider knob similar to a brightness control knob on the front panel of a monitor. The EDLS knob controls the energy vs. the quality of the image trade-off rather than the luminance of the backlight. It provides users with an extensive backlight power management scheme which can be used to extend the battery life at the cost of whatever display degradation they are prepared to accept. There is also an automatic mode which changes the power management level depending on the remaining battery energy.

When the system is powered on by an external power source, the backlight will be fully turned on and exhibit its maximum luminance. In this case, no backlight power management is desirable, so that users may enjoy the best image quality. However, even when the remaining energy level of the battery source is high, users still may want to extend the battery life for future use, but they are generally not ready to sacrifice picture quality appreciably at that stage. As the remaining battery energy decreases, users may become more willing to compromise the image quality to improve the battery life. At this point, EDLS applies DLS.

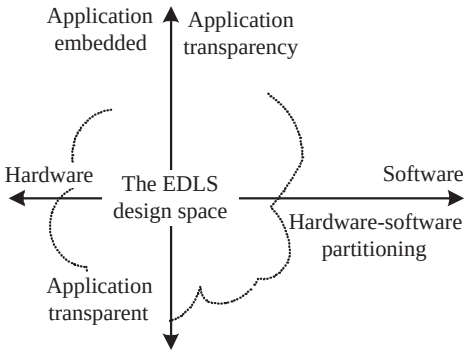


Figure 4: The EDLS design space.

With a poor power budget, users may simply want to complete their current task within the remaining battery energy, although the image quality becomes poor. This is the right time for EDLS to change from the DLS mode to the DCE mode. Although DCE may alter the original colors, a moderate degree of DCE can maintain a fixed distortion ratio. But if the battery energy is nearly exhausted, the only alternative is to turn off the backlight to continue with task execution. Without the backlight, EDLS applies the best DCE to achieve the maximum possible contrast. In this case, EDLS cannot guarantee a fixed amount of image distortion, but users may be provided with more readability.

3 The trade-offs in the EDLS implementation

3.1 The EDLS design space

We analyze EDLS capability in terms of the energy reduction, the energy overhead, the performance penalty, response time and the image quality. They are affected by the hardware-software trade-offs and the application-transparency with a given display specification.

The EDLS process starts by building a RGB histogram of the image to be displayed. The EDLS slider determines the panel mode (transmissive or reflective), the image processing algorithm (DLS or DCE), and the maximum allowed percentage of saturated pixels, S^R, after image processing. The EDLS process derives T_H and T_L from S^R and the histogram. Eqs. 1 and 2 show how to calculate the scaling factor that controls the amount of backlight dimming. The command for a new luminance is forwarded to the PID controller rather than the backlight inverter to improve response time.

As there are many ways to add EDLS to an application, we illustrate the EDLS design space as a plane, with two axes corresponding to the application transparency and the hardware-software partitioning (cf. Fig 4). The optimization objectives are the energy reduction of the backlight, the energy overhead and the performance penalty of the EDLS process itself, and the resultant image quality. We consider these optimization goals in the four quadrants of the design space. However, application-embedded hardware EDLS, the left-upper quadrant in Fig. 4, is not a reasonable solution, and thus we exclude it from further discussion.

Fig. 5 summarizes how to add the EDLS capability to typical multi-media applications. These applications draw images in the frame buffer, and the backlight luminance is fixed by user preference (Fig. 5 (a)). The gray boxes represent functional blocks to be added to the existing implementation. Our first approach is to embed EDLS in an application program (Fig. 5 (b)). The advantage of this approach is that we will have many opportunities to reduce the EDLS overhead. For example, we may construct an approximate histogram in a compressed domain for an MPEG decoder. Sometimes, we may have the histogram before rasterization and thus avoid additional frame buffer accesses. However, such optimizations are *ad hoc* and should be explored only case by case. Modification of an application to support the EDLS discourages developers from getting the benefit of EDLS due to the heavy porting burden. A standard EDLS API makes the porting process systematic [4], but this approach still involves source code modification and, in many cases, EDLS developers cannot access the source code of an existing application. Even though this approach is limited by portability, it achieves the best energy reduction and image quality because it can

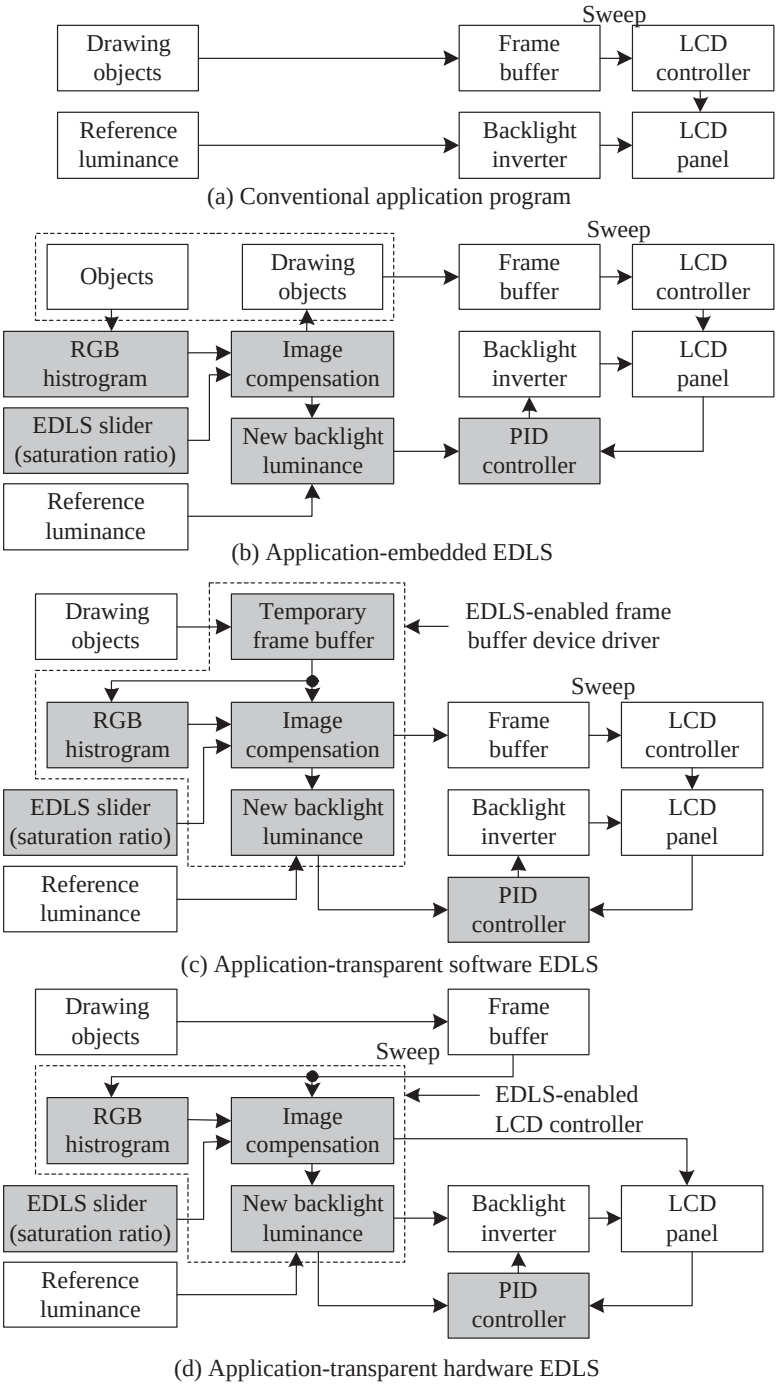


Figure 5: Porting the EDLS capability.

use the application context [4].

The alternative approach is to implement EDLS functionality outside the applications, as shown in Fig. 5 (c) and (d). This offers an application-transparent EDLS implementation because we do not have to modify the existing application. Instead, we simply redirect the frame buffer address pointer using a new device driver. The EDLS functional blocks are then periodically read the temporary frame buffer and rebuild the histogram, although visual artifacts may occur due to improper synchronization between an application and the EDLS functional blocks.

The frame buffer device driver is different from the general character or block device driver of the Linux operating system. There are no read and write routines in the device driver source code. Instead, application programs access the frame buffer directly without the aid of the driver. The role of the frame buffer device driver is to find a frame buffer device, create the page table entries corresponding to the physical address of the frame buffer, and report the virtual base address of the frame buffer to the Linux kernel. After initialization, there is nothing for the device driver to do because the application directly accesses the frame buffer memory to draw cursor, menus, pictures and so on. Thus, oversampling is the only way to synchronize an application with the EDLS functional blocks. The potential overhead for refreshing the histogram is much higher than that of application embedded EDLS because the EDLS functional blocks must read the entire frame buffer during in every refresh period. Since we cannot modify the application, no collaboration between the application and the EDLS functional blocks is possible. Fig. 5 (c) shows how all the EDLS functional blocks are implemented in a frame buffer device driver. Except for the PID control block, which is beyond the ability of Linux-based systems, all the EDLS functional blocks are implemented in the device driver. This approach is subject to the EDLS overhead, which increases with the screen resolution, the color depth and the refresh rate.

We can also embed the EDLS functional blocks in an LCD controller (Fig. 5 (d)). This offers a user-friendly EDLS implementation because the users do not have to make any modification of the existing application. Synchronization of the EDLS functional blocks with an application never causes visual artifacts since the LCD controller sweeps the LCD panel every 16.67ms, and application-transparent hardware EDLS updates the histogram whenever the sweep operation occurs. The hardware EDLS approach does not involve any additional frame buffer accesses. We add extra comparators and counters to a standard LCD controller for histogram construction. Image processing requires additional datapath resources such as multipliers, adders and comparators, as defined by Eqs. 1 and 2. Image processing is performed on-the-fly before issuing the RGB color data to the LCD panel. The frame buffer always contains the original image regardless of the EDLS system. Detailed quantitative analysis is presented in the next sections.

3.2 Compact EDLS

As mentioned in Section 3.1, the area complexity of application-transparent hardware EDLS explodes as the color depth increases. More precisely, the area overhead increases exponentially by the color depth, n, in order to include 2^n counters and comparators to construct the histogram. However, we can approximate the EDLS algorithms to reduce the area complexity. Our test platform requires 64 19-bit counters and 6-bit comparators for full-resolution histogram construction of 640×480 resolution and 16-bit color depth. The area explosion corresponding to the color depth affects both the cost and energy overhead. We forge a compromise between the energy saving from the backlight and the area complexity of the EDLS-enabled LCD controller as follows:

Definition 1 *We define compact EDLS-d, whereby d signifies that d-digit histogram, where $d \leq n$ are used. More precisely, we truncate the color values to d-digit numbers and compose a d-digit histogram.* ■

Using compact EDLS-d a 2^n-level histogram is approximated by a 2^d-level histogram. This imposes restrictions on the value of T_H and T_L, and reduces the area complexity of application-transparent hardware EDLS from 2^n to 2^d.

Compact EDLS-d may achieve lower power savings from the backlight compared to EDLS because it may result in smaller brightness compensation or contrast enhancement factor. Note that the energy reduction from the backlight is roughly equal to $\frac{1}{S_{BC}}$ or $\frac{1}{S_{CE}}$, depending on the EDLS mode.

The worst-case threshold, T'_H, and the brightness compensation factor, S'_{BC}, of compact EDLS-d are calculated as

$$T'_H = \lceil T_H, \frac{2^n}{2^d} \rceil = T_H + \frac{2^n}{2^d}, \tag{3}$$

Table 1: The trade-offs between hardware EDLS and software EDLS (*Prop.*: proportional, *Exp.*: exponential, *NA*: not affected, *Mit.*: mitigated from 2^n to 2^d).

Implementation		Screen resolution	Color depth	Refresh rate	Compact EDLS-d
Software	Power	*Prop.*	*Prop.*	*Prop.*	*NA*
	Delay	*Prop.*	*Prop.*	*Prop.*	*NA*
Hardware	Power	*NA*	*Exp.*	*NA*	*Mit.*
	Area	*NA*	*Exp.*	*NA*	*Mit.*

and

$$S'_{BC} = \frac{2^n - 1}{T'_H} = \frac{2^d(2^n-1)S_{BC}}{2^n S_{BC} + 2^d(2^n-1)} \approx \frac{2^d S_{BC}}{S_{BC} + 2^d}, \tag{4}$$

where T_H and S_{BC} are the threshold and the brightness compensation factor used by EDLS. The worst-case difference in the power reduction between EDLS and compact EDLS-d in the DLS mode is:

$$P_B \times (\frac{1}{S_{BC}} - \frac{1}{S'_{BC}}) = P_B \times \frac{2^n}{2^d(2^n-1)}, \tag{5}$$

where P_B is the original backlight power consumption without EDLS or compact EDLS-d.

In the same way, the worst-case upper and lower thresholds, T'_H and T'_L, and the worst-case contrast enhancement factor, S'_{CE}, of compact EDLS-d in the DCE mode are calculated as

$$T'_H = \lceil T_H, \frac{2^n}{2^d} \rceil = T_H + \frac{2^n}{2^d}, \tag{6}$$

$$T'_L = \lfloor T_L, \frac{2^n}{2^d} \rfloor = T_L - \frac{2^n}{2^d}, \tag{7}$$

and

$$S'_{CE} = \frac{2^n - 1}{T'_H - T'_L} = \frac{2^d(2^n-1)S_{CE}}{2^{n+1} S_{CE} + 2^d(2^n-1)} \approx \frac{2^d S_{CE}}{2S_{CE} + 2^d}, \tag{8}$$

where T_H, T_L are the upper and lower thresholds, and S_{CE} is the contrast enhancement factor used by EDLS. Thus, the worst-case difference in the power reduction between EDLS and compact EDLS-d in the DCE mode is calculated as

$$P_B \times (\frac{1}{S_{CE}} - \frac{1}{S'_{CE}}) = P_B \times \frac{2^{n+1}}{2^d(2^n-1)}. \tag{9}$$

Note that the backlight power consumption penalty due to the approximation made by compact EDLS-d is usually much less than the worst-case value, as calculated above.

3.3 The trade-offs between hardware EDLS and software EDLS

The area and energy overhead for application-transparent hardware EDLS-d grows exponentially as d increases. But hardware compact EDLS-d exhibits a constant area and energy overhead regardless of the screen resolution, the color depth and the refresh rate. Furthermore, the energy and delay overhead does not exhibit noticeable change as the screen resolution and the refresh rate increase, because application-transparent hardware EDLS is performed during the sweep operation.

As the hardware EDLS-d and software EDLS behave differently with each other as shown in Table 1, we must choose an appropriate implementation scheme considering the display specification. Software EDLS is superior to hardware compact EDLS-d for low screen resolution, low color depth and low refresh rate. On the other hand, hardware EDLS becomes more beneficial as the screen resolution, color depth and refresh rate increase.

Once hardware EDLS-d has been determined for a given display specification, we can further compromise the energy reduction from the backlight and the area overhead for the EDLS functional blocks. Hardware EDLS-d slightly

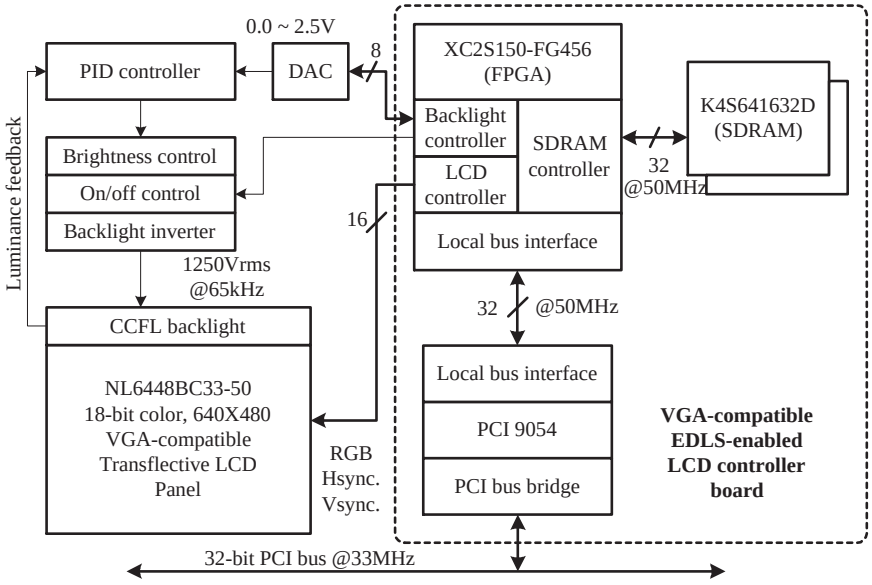


Figure 6: The block diagram of an application-transparent EDLS prototype.

reduces the energy saving achieved by backlight dimming, and results in minor inconsistency in the inter-frame saturation ratio in the video applications, as d decreases, *i.e.*, more compaction. Note that such compaction is also applicable to the software-oriented approach, but we would expect no enhancement in either the energy or time overhead because all the operations must be performed by the same datapath resources in the CPU.

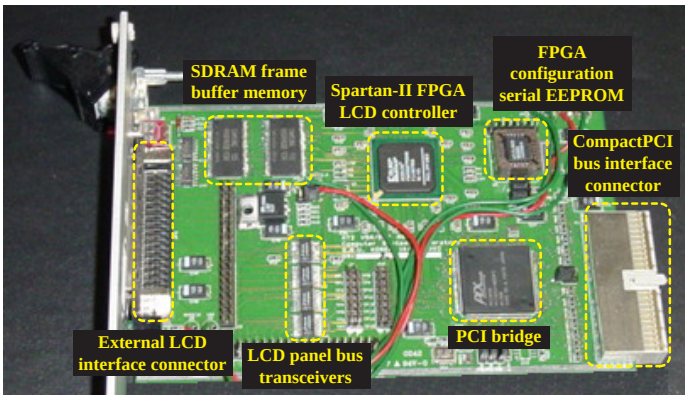
4 Experimental results

4.1 Implementation

Based on the design-space analysis, we have implemented a VGA (Video Graphics Adapter) compatible LCD controller with application-transparent hardware EDLS-4. It turned out that a pure hardware implementation is superior to a pure software implementation for 640×480 screen resolution with 16-bit color depth. But, we implemented both software EDLS and hardware EDLS-4 for performance evaluation purpose.

The architecture of the prototype is shown in Fig. 6. The implementation includes an FPGA (Field-Programmable Gate Array) EDLS-enabled LCD controller, a PCI bus interface, a frame buffer memory and a PID (Proportional, Integral and Differential)-controlled CCFL backlight inverter, in the form of a 3U CompactPCI board (Fig. 7 (a)). Since the EDLS-enabled LCD controller must be backward compatible with the conventional VGA-compatible LCD controllers, at first, we had to implement mandatory IP (Intellectual Property) cores including a VGA timing generator, an SDRAM controller, a local bus interface, etc. The LCD controller is equipped with two Samsung K4S641632D SDRAM devices for the frame buffer memory. The backlight system of an NEC6448BC33-50 10.4" TFT LCD panel consumes about 8.1W at its maximum luminance. Thanks to an effective compaction of the EDLS algorithms to reduce the hardware resource requirement, we could achieve the FPGA targeting of the EDLS capability, even with a small and low-cost Xilinx Spartan-II FPGA, XC2S-150FG456, without significantly compromising the power savings in the backlight system.

Our target platform is an Intel XScale-based Linux machine, whose backbone is a CompactPCI Mezzanine bus. The XScale platform has been built as a low-power embedded system platform known as the APOLLO (Adaptive Power Optimization and controL for the Land warriOr). The APOLLO platform was built by the University of Southern California and designed by Seoul National University as a joint project. In this paper, we focus on implementing EDLS on a custom LCD controller board. As shown in Fig. 8, the APOLLO platform is a fully integrated low-power test-bed with sophisticated power measurement features. The APOLLO platform is designed to interface with a UDAS



(a) The EDLS-enabled VGA-compatible LCD controller board

(b) The PID controller implemented with a PIC RISC microcontroller

Figure 7: The EDLS-enabled VGA-compatible LCD controller board and the PID controller .

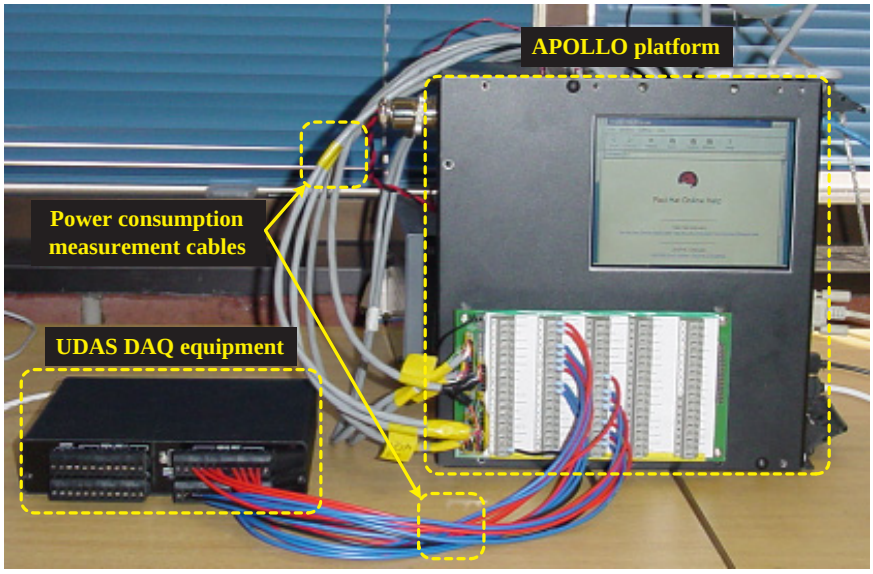


Figure 8: The APOLLO platform.

Table 2: The normalized backlight luminance with application-transparent EDLS and EDLS-4 for the still images ($S^R = 0.3$).

	EDLS-4 (hardware)		EDLS (software)	
	DLS	DCE	DLS	DCE
S1	0.80	0.67	0.76	0.65
S2	0.94	0.84	0.91	0.81
S3	0.80	0.80	0.78	0.78
S4	0.84	0.76	0.79	0.73

(USB Data Acquisition System) for logging power consumption profiles. Fig. 7 (b) shows the separate implementation of a PID controller, which is located near the CCFL inverter. Since the Linux operating system tends to have a slow response time due to its heavy locking mechanism, a 1ms timer interrupt for the PID control is not achievable with the XScale microprocessor of the APOLLO platform. Therefore, we used a simple RISC microcontroller, PIC16C74A from Microchip Technology instead.

The prototype is not only compatible with the APOLLO platform but also with any CompactPCI system, either for 3U or for 6U. The EDLS-enabled LCD controller is supported by a VGA-compatible Linux driver corresponding to the Linux kernel 2.4.19. As a result, the APOLLO platform is capable of utilizing the EDLS capability for all kinds of applications that use the LCD display, without any modification of the application programs.

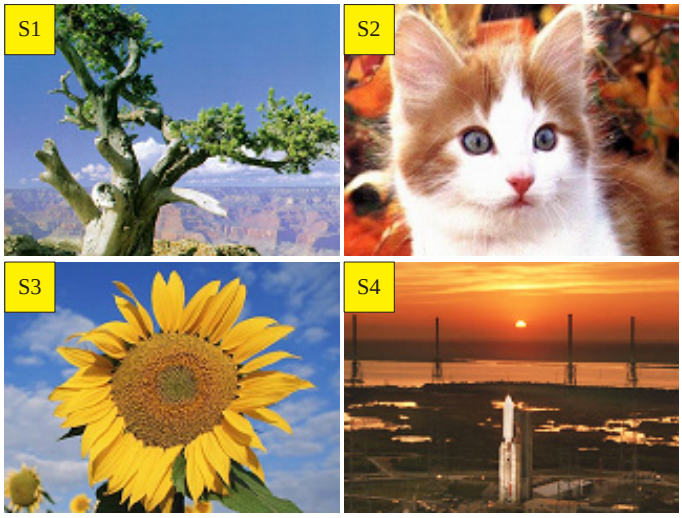
4.2 Energy reduction and image quality

We compared the energy reduction from the backlight of software EDLS and hardware EDLS-4. First, we enabled software EDLS with the still images (Fig. 9 (a)). As shown in Table 2, we achieved a 9% to 35% energy reduction from the backlight with $S^R = 0.3$. Next, we applied hardware EDLS-4 for the same still images. It exhibits slightly higher backlight luminance than software EDLS, but the differences are less than 5% with $S^R = 0.3$. As for the image quality, the final simulated images are shown in Fig. 9 (b) and Fig. 9 (c), which confirm that even EDLS-4 results in only minor quality degradation of the image.

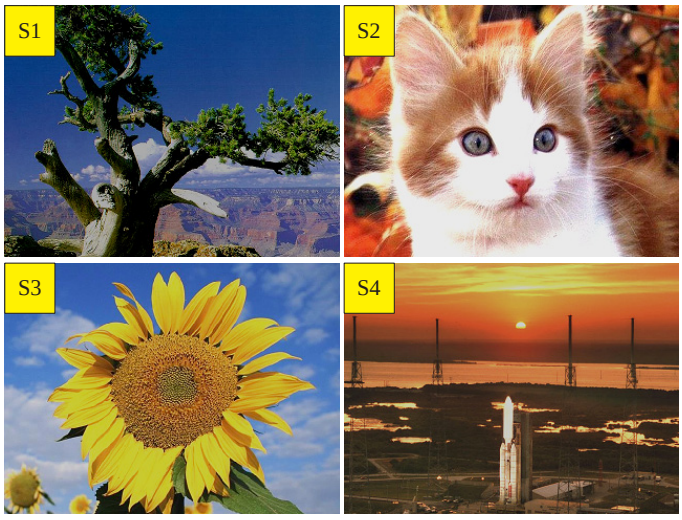
Finally, we applied both software EDLS and hardware EDLS-4 to a movie clip, namely a trailer for the movie 'Bad Boys 2'. Normalized backlight luminance for each picture frame is illustrated in Fig. 10 with $S^R = 0.2$. Backlight power consumption is reduced by 20% and 31% on average with software EDLS in the DLS and DCE modes, respectively. On the other hand, we achieve by 19% and 31% of the backlight power reduction using hardware EDLS-4, in the DLS and DCE modes, respectively. This shows that the reduction of backlight power with hardware EDLS-4 is almost same as that with software EDLS in this video application. Although similar results on many other movies confirm the same conclusions, they are not reported here due to space limitation. This shows that the compact EDLS with $d \geq 4$ does not cause severe performance degradation in general.

4.3 Power, delay and area overhead

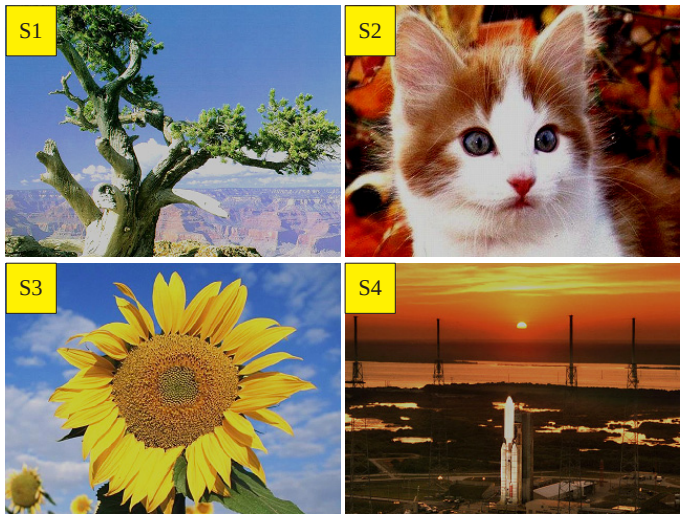
Fig. 11 summarizes the EDLS overhead using the application-transparent software approach in terms of the screen resolution, the refresh rate and the color depth. We measured the power and timing behavior of this approach on the APOLLO platform. The EDLS process is usually designed to respond to 30Hz screen refresh rate requirement for quality movie streams. The device driver needs a 36.9MB/s data bandwidth to refresh the histogram for 640×480 resolution with 16-bit color depth and 30Hz refresh rate. We measured the power overhead and the CPU utilization when the device driver occupies less than 100% of CPU resource by changing the screen resolution, the color depth and the refresh rate. Although more than 100% utilization is not justified, we extrapolate the data up to 640×480 screen resolution with 30Hz refresh rate for the comparison with the hardware approach to be discussed next. The energy and delay overhead linearly increases with the screen resolution, refresh rate and color depth increase. At 640×480 resolution with 16-bit color depth and 30Hz refresh rate, application-transparent software EDLS requires over 300% utilization of a 733MHz XScale processor by the EDLS-enabled device driver alone, and results in a 240mW power overhead. Most of all, the timing overhead that exceeds 300% of the CPU utilization is unacceptable. It is no wonder that we cannot implement application-transparent software EDLS at the 640×480 resolution LCD panel with 16-bit



(a) Test still images for application-transparent hardware EDLS-4



(b) Resultant images after EDLS-4 in DLS mode (SR=0.3)



(c) Resultant images after EDLS-4 in DCE mode (SR=0.3)

Figure 9: The still images before and after application-transparent hardware EDLS-4.

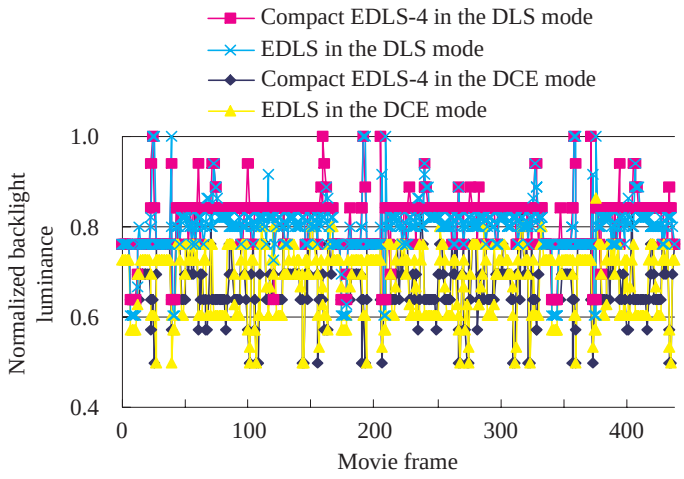


Figure 10: The backlight luminance adaptation with application-transparent EDLS and EDLS-4 for a movie clip ($S^R = 0.2$).

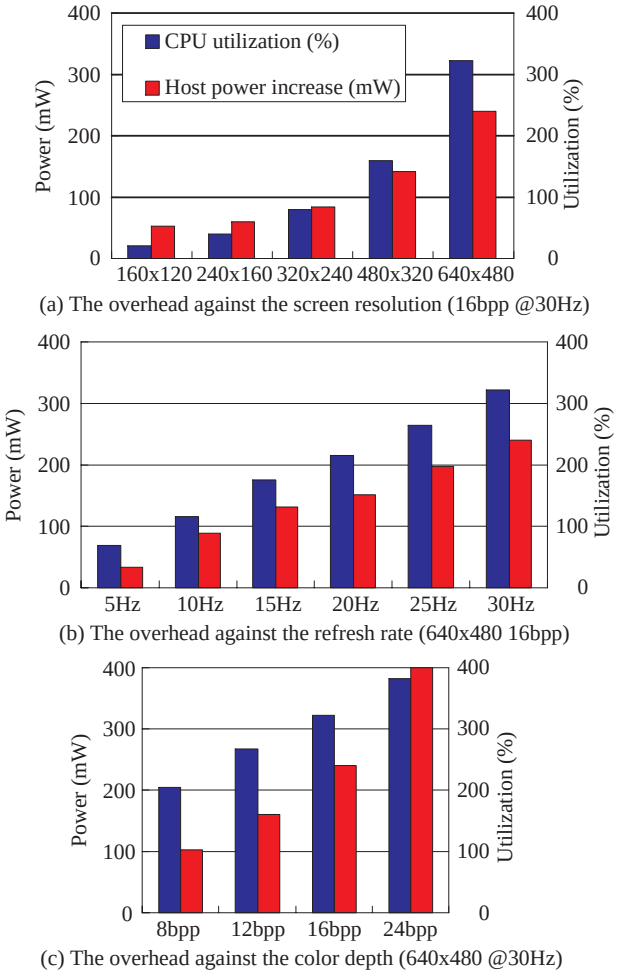


Figure 11: The energy overhead and the performance penalty of application-transparent software EDLS.

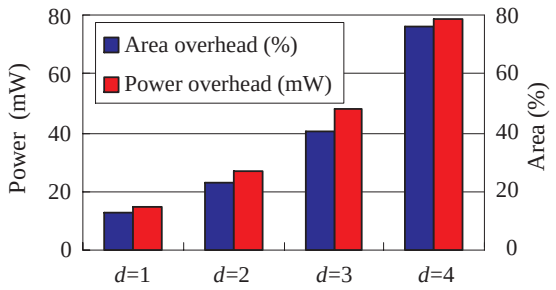


Figure 12: The energy-area overhead of application-transparent hardware EDLS-*d*.

color depth and 30Hz refresh rate on the APOLLO platform.

Next, we measured the power consumption of the LCD controller with and without the EDLS capability using the UDAS on the APOLLO platform. We allocated 849 slices, 750 slice flip-flops, 182 IOBs, 2 GCLKIOBs, 2 Block RAMs and 2 DLLs, all of which occupy only 63,246 equivalent gates for the LCD controller without EDLS. Without the EDLS capability, the whole LCD controller board consumes 1,307mW, of which 225mW is the power consumption of the FPGA core. After adding the EDLS capability into the LCD controller, it occupies 1,574 slices, 1,448 slice flip-flops, 182 IOBs, 2 GCLKIOBs, 2 Block RAMs and 2 DLLs, with an equivalent gate count of 77,578 gates. Our EDLS is a 16-level color histogram analysis and replaces a 64-level full-resolution analysis. The resultant total power consumption is 1,392mW, while the core power consumption is 284mW, which shows that the power consumption overhead of hardware EDLS-*d* is quite small. Fig. 12 summarizes EDLS-*d* area and its power overhead for constructing the histogram.

On the other hand, the image enhancement processing does not cause serious overhead in hardware EDLS. It requires additional datapath resources such as multipliers, adders and comparators, but only three 13-bit precision integer multipliers can manage the image processing for 16-bit color depth. The multipliers for image processing require just 77 slices, which correspond to an area overhead of 9%. Furthermore, the total power consumption of the LCD controller prototype increases by only 6mW due to the image processing.

5 Conclusions

This paper introduces extended DLS (EDLS) as a framework for backlight power management of the transflective LCD panels for quality multi-media applications powered by batteries. We have explored the EDLS design space where the application transparency and the hardware-software partitioning show trade-offs in terms of the energy reduction, the energy overhead, the performance penalty and the image quality. As there is a further trade-off between the energy reduction and the area overhead, we proposed compact EDLS, EDLS-*d*, which approximates the image processing algorithm of EDLS and dramatically mitigates the explosion of gate count in the LCD controller, achieving nearly the same backlight power savings as the exact EDLS system. Our hardware compact EDLS has perfect application transparency, and thus supports all kinds of multi-media applications running on Linux without any modification. We demonstrate that the EDLS compaction with $d \geq 4$ results in an average backlight power savings of 25% without any appreciable image quality degradation.

References

[1] I. Choi, H. Shim, and N. Chang, "Low-power color TFT LCD display for hand-held embedded systems," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 112 – 117, August 2002.

[2] I. Choi, H. Shim, N. Chang, I. K. Kim, J. Moon, and N. I. Cho, "Low-power liquid crystal display systems using dynamic backlight luminance scaling," in *Low Power Design Contest of International Symposium on Low Power Electronics and Design*, August 2002.

[3] F. Gatti, A. Acquaviva, L. Benini, and B. Ricco, "Low power control techniques for TFT LCD displays," in *Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pp. 218 – 224, October 2002.

[4] I. Choi, H. S. Kim, H. Shin, and N. Chang, "LPBP: low-power basis profile of the java 2 micro edition," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 36 – 39, August 2003.