# A Backlight Power Management Framework for Battery-Operated Multimedia Systems

**Hojun Shim and Naehyuck Chang**
Seoul National University

**Massoud Pedram**
University of Southern California

*Editors' note:*
Thin-film transistor liquid-crystal displays are systems widely used to support full-featured multimedia. For such systems, backlight is a major source of power dissipation. This article introduces a backlight power management framework and explores trade-offs in the extended dynamic-luminance-scaling design space in terms of energy reduction, performance penalty, and image quality.

*—Radu Marculescu, Carnegie Mellon University; and Petru Eles, Linkoping University*

**COLOR THIN-FILM TRANSISTOR** (TFT) liquid-crystal-display (LCD) panels enable battery-operated, handheld embedded systems to support full-featured multimedia, and have replaced monochrome supertwisted nematic LCD panels in most applications. Most importantly, a TFT LCD panel does not illuminate itself, but filters a backlight source, and this backlight is a primary power consumer in most systems. Thus, reducing backlight power consumption is one of the primary ways to extend battery life in battery-powered electronic devices. Most existing power reduction techniques are based on power management during idle or slack times, and are therefore difficult to apply to display panels, which have no idle time as long as they are turned on. Simply dimming or turning off the backlight results in appreciable degradation of the display's legibility.

Recently, Choi et al. introduced a power reduction technique[1] that maintains either the brightness or contrast of an LCD panel when the backlight is dimmed down. Appropriate image compensation techniques preserve either the brightness or contrast of the original image at the expense of minor image distortion, which does not seriously affect the legibility of the display. Because the cold cathode fluorescent lamp (CCFL) backlight usually responds slowly to the input changes for luminance control, Choi et al. have proposed a feedback control circuit[2] that enables the backlight luminance to change fast enough to support movie streams. This technique is known as dynamic luminance scaling (DLS) of a backlight. Both the brightness of the LCD panel and the ambient luminance affect a person's ability to read a display, and this has motivated another approach involving backlight autoregulation in the context of ambient luminance.[3] Simultaneous brightness and contrast scaling[4] enhances image fidelity with a dim backlight, and thus permits an additional reduction in backlight power. In an MPEG-1 video streaming application, Pasricha et al. have implemented this approach (including the necessary image processing) using adaptive middleware to avoid any extra burden on the streaming clients.[5]

In this article, we introduce a new backlight power management framework, *extended DLS* (EDLS), for the color TFT LCD panels used in battery-operated multimedia applications. We extend DLS to cope with transflective LCD panels, which operate both with and without a backlight, depending on the remaining battery energy and ambient luminance. These popular transflective LCD panels are the dominant choice for battery-operated electronic systems because they allow an image to remain visible without a backlight, even though the quality can be poor. Our technique compensates for loss of brightness with a rich or moderate power budget and for loss of contrast with a low power budget.

The need to modify existing applications limited the scope of use for DLS because previous implementations required the modification of source code in existing multimedia applications.[2] In developing EDLS, we have pursued an application-transparent approach that intercepts the frame buffer contents and performs image compensation. We expect that this will contribute to the wide adoption of DLS. Our system is an application-transparent, pure-hardware EDLS implementation with a 16-bit color depth and a 640 × 480-pixel screen resolution. It exhibits a 25% power reduction while maintaining acceptable image quality. The hardware overhead, in terms of both area and power, is moderate.

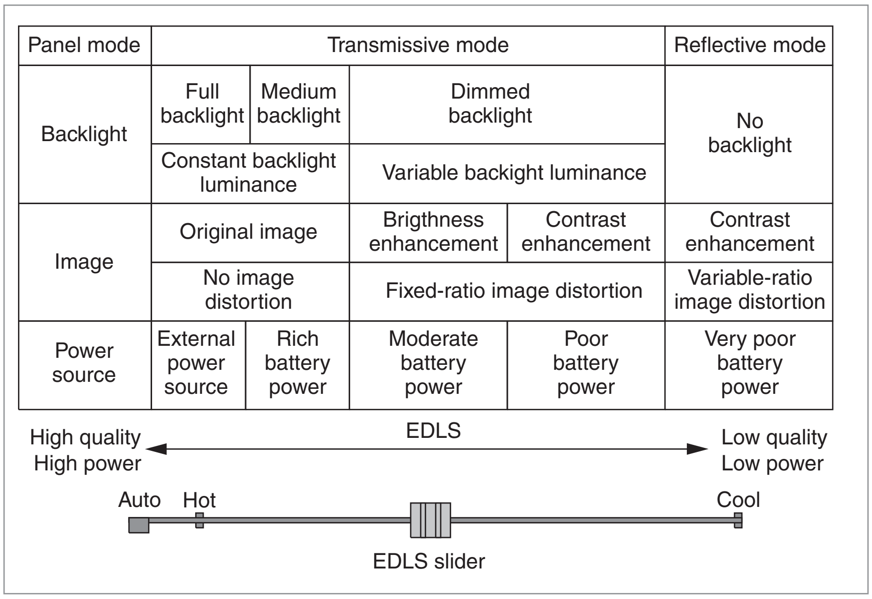

**Figure 1. EDLS framework.**

## Transflective LCD

There are three types of TFT LCD panels.[6] In *transmissive* LCDs, a backlight illuminates the pixels from behind (that is, opposite the viewer). Transmissive LCDs offer a wide color range and high contrast, and are typically used in laptops. They perform best under lighting conditions ranging from complete darkness to an office environment. *Reflective* LCDs are illuminated from the front (that is, the same side as the viewer). Reflective LCD pixels reflect incident light originating from the ambient environment or a frontlight. Reflective LCDs can offer very low power consumption (especially without a frontlight) and are often used in small portable devices such as handheld games, PDAs, or instrumentation. They perform best in a typical office environment or in brighter lighting. Under dim lighting, reflective LCDs require a frontlight.

*Transflective* LCDs are partially transmissive and partially reflective, so they can make use of environmental light or a backlight. Transflective LCDs are common in devices used under a wide variety of lighting conditions, from complete darkness to sunlight.

Transmissive and transflective LCD panels use very bright backlight sources that emit more than 1,000 cd/m$^2$. However, the transmittance of the LCD, $\rho_T$, is relatively low, and thus the resultant maximum luminance of the panel is usually less than 10% of the backlight luminance. Theoretically, the backlight and the ambient light are additive. However, once the backlight is turned on, a transflective LCD panel effectively operates in the transmissive mode because the backlight source is generally much brighter than the ambient light.

The brightness in transmissive mode is proportional to the product of transmittance $\rho_T$ and backlight luminance $L_B$.[7] Similarly, the brightness in the reflective mode is proportional to the product of reflectance $\rho_R$ and ambient luminance $L_A$. The reflectance of an LCD panel is even lower than its transmittance. The transmissive mode is significantly superior to the reflective mode in terms of both brightness and contrast. For example, the NEC6448BC33-50 LCD panel exhibits a contrast ratio of 300:1 in transmissive mode versus 8:1 in reflective mode.

## The EDLS framework

The principle of DLS is to reduce the light source's luminance but compensate for the loss in brightness by allowing more light to pass through the screen, enhancing the image luminance.[1,2] The viewer should perceive little change. Dynamic contrast enhancement (DCE) also enhances image quality under a dimmed backlight, but does so by increasing the image's contrast. So although DLS preserves the original colors, DCE can result in a noticeable change to the original colors in pursuit of higher contrast and improved legibility. Thus, DCE is a very aggressive power management scheme for transmissive LCD panels, which differentiates it from DLS.

The EDLS framework, as illustrated in Figure 1, achieves a harmonious combination of DLS and DCE.

The EDLS interface is a simple slider knob, similar to a brightness control knob on a monitor. The EDLS knob controls the trade-off between energy consumption and image quality, and not simply the backlight's brightness. It provides users with a power management scheme that can extend battery life at the cost of whatever display degradation the user will accept. There is also an automatic mode that changes the power management setting, depending on the remaining battery energy.

When connected to an external power source, the backlight is fully on and exhibits its maximum luminance. There should be no backlight power management so that users can enjoy the best image quality. When the system is battery powered, however, users might want to extend the battery life for future use, even if the battery is already fully charged. But users generally aren't ready to sacrifice appreciable picture quality at that stage. As the remaining battery energy decreases, users might become increasingly willing to compromise image quality to extend battery life. This is the point at which EDLS applies DLS.

With a poor power budget, the user's prime concern might well be to complete their current task within the remaining battery energy budget, even if the image quality decreases. This is the optimum time for EDLS to change from DLS to DCE mode. Although DCE might alter the original colors, a moderate degree of DCE does at least maintain a fixed distortion ratio. However, if the battery energy is nearly exhausted, the only remaining option is to turn off the backlight. Without the backlight, EDLS applies DCE to achieve the maximum possible contrast. In this case, EDLS cannot guarantee a fixed amount of image distortion, but the user should still be able to read the display and finish the task.

### Formulation of DLS and DCE

To build the EDLS framework, we borrowed the DLS principles of brightness compensation and DCE principles of contrast enhancement.[2] The EDLS process starts by building a red-green-blue (RGB) histogram of the image for display. 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. ($S^R$ is a given input parameter determined by user preference.) Then, the EDLS process derives upper and lower thresholds $T_H$ and $T_L$ from $S^R$ and the histogram, and calculates a scaling factor that controls the amount of backlight dimming, as Choi et al. have shown.[2] Let $C$ denote the current color value. After brightness compensation, new color value $C'$ is

$$C' = \min(2^n - 1, S_{BC}C), \qquad (1)$$

where $n$ is the color depth of each color component, and $S_{BC}$ is the brightness compensation factor, which equals $(2^n - 1)/T_H$. Similarly, after contrast enhancement, new color value $C'$ is

$$C' = \min[2^n - 1, S_{CE}\max(0, C - T_L)], \qquad (2)$$

where contrast compensation factor $S_{CE}$ equals $(2^n - 1)/(T_H - T_L)$. After image compensation, we reduce backlight luminance $L_B$ so that $L_B' = L_B/S_{BC}$ or $L_B' = L_B/S_{CE}$, depending on the EDLS mode. The power reduction ratio of the backlight is $1 - L_B'/L_B$.[1]

## Trade-offs in EDLS implementation

Figure 2 summarizes how to add EDLS capability to typical multimedia applications. Such applications draw images in the frame buffer, and user preference sets the backlight luminance (Figure 2a). Because there are many ways to add EDLS to an application, we must consider application transparency and hardware-software partitioning. In addition, we must optimize the backlight energy, energy overhead, and the performance penalty of the EDLS process itself, balancing those costs against the resulting image quality.

Our first approach is to embed EDLS in an application (Figure 2b). The advantage of this approach is that it gives us many opportunities to reduce the EDLS overhead. For example, we can construct an approximate histogram in a compressed domain for an MPEG decoder. Sometimes, we can obtain the histogram before rasterization and thus avoid additional frame buffer accesses. Our first DLS implementation falls into this category; it was highly coupled with the application. (Choi et al. demonstrated the first DLS implementation at the SIGDA university booth at the 2000 Design Automation Conference and at the design contest during the 2002 International Symposium on Low Power Electronics Design.) However, such optimizations are ad hoc, and thus the necessary changes to the application discourage developers from using EDLS because of the heavy porting burden.

Our second DLS implementation introduced a standard application programming interface (API) at the window management level.[2] A standard EDLS API makes porting systematic,[2] but this approach still involves source code modification. However, in many cases, EDLS developers simply cannot access an existing application's source code. Even though this approach has limited portability, it maximizes energy reduction and image quality because it can use the application context.[2]

The alternative approach is to implement EDLS functionality outside the applications, as Figures 2c and 2d illustrate. This approach offers an application-transparent EDLS implementation because it doesn't require modification of the existing application. Instead, we simply redirect the frame buffer address pointer using a new device driver. The EDLS functional blocks then periodically read the temporary frame buffer and rebuild the histogram. However, visual artifacts, such as a flicker on the LCD panel, can occur because of improper synchronization between the application and the EDLS functional blocks. Figure 2c demonstrates the implementation of all the EDLS functional blocks in a frame buffer device driver. Because the application directly accesses the frame buffer memory to draw the cursor, menus, pictures, and so forth, oversampling is the only way to synchronize an

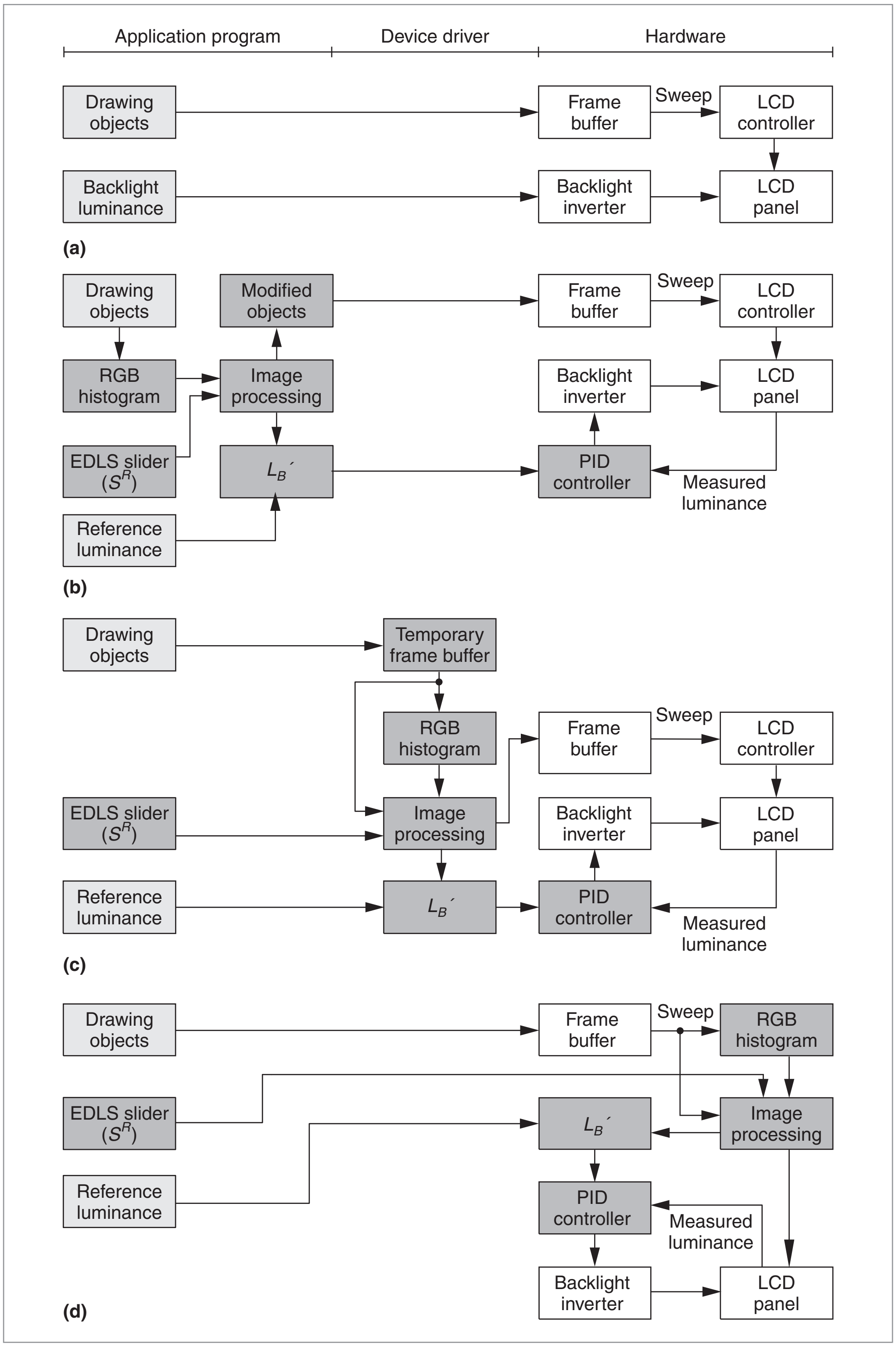

**Figure 2. Porting the EDLS capability. We can apply EDLS to a conventional application (a), through EDLS-embedded (b) or application-transparent (c) software, or through application-transparent hardware (d). White boxes represent hardware blocks in the LCD display system. Light gray boxes represent software blocks in the original application program. Dark gray boxes represent functional blocks added to an existing implementation.**

We have forged a compromise between energy savings from the backlight and the area complexity of the EDLS-enabled LCD controller.

application with the EDLS functional blocks. The overhead for refreshing the histogram is potentially much higher than that of an application-embedded implementation because the EDLS functional blocks must read the entire frame buffer in every refresh period.

An appropriate way to partition hardware and software might be to embed the EDLS functional blocks in the LCD controller (see Figure 2d). In that case, synchronizing the EDLS functional blocks with an application does not cause visual artifacts because the LCD controller sweeps the LCD panel every 16.67 ms (LCD panels commonly have a 60-Hz refresh rate), and application-transparent hardware EDLS updates the histogram whenever the sweep operation occurs. We added extra comparators and counters to a standard LCD controller to construct the histogram. Image processing requires additional data path resources such as multipliers, adders, and comparators to perform the manipulations defined in equations 1 and 2, but the hardware EDLS approach does not involve any additional frame buffer accesses. We performed image processing on the fly before issuing the RGB color data to the LCD panel; the frame buffer always contains the original image.

## Compact EDLS

The area overhead of application-transparent hardware EDLS increases exponentially with color depth *d* because $2^n$ counters and comparators are necessary to construct the histogram. However, we can approximate the EDLS algorithms to reduce this area overhead. Our test platform requires 64 counters (19 bits each) and 64 comparators (6 bits each) to construct a full-resolution histogram at a $640 \times 480$-pixel resolution display and 16-bit color depth. The area explosion corresponding to the color depth affects both the cost and energy overhead. We have forged a compromise between energy savings from the backlight and the area complexity of the EDLS-enabled LCD controller, called *compact EDLS*. We use the acronym EDLS-*d*, in which *d* signifies a *d*-digit histogram, where $d \le n$. More precisely, we truncate the color values to *d* numbers and compose a *d*-digit histogram. Using EDLS-*d*, we approximate a $2^n$-level histogram with a $2^d$-level histogram. This restricts the values of $T_H$ and $T_L$, and reduces the area complexity of application-transparent hardware EDLS from $2^n$ to $2^d$.

EDLS-*d* might achieve less power savings from the backlight than full EDLS because it can result in reduced brightness compensation or a smaller contrast enhancement factor. The energy reduction that the backlight dimming achieves roughly equals $1/S_{BC}$ or $1/S_{CE}$, depending on the EDLS mode. We calculate worst-case threshold $T_H'$ for EDLS-*d* as

$$T_H' = \lceil T_H, 2^n/2^d \rceil = T_H + 2^n/2^d,$$

where notation of the form $\lceil A, B \rceil$ denotes a ceiling function of number *A*. The ceiling function rounds *A* up to the nearest multiple of significance, *B*.

We also calculate brightness compensation factor $S_{BC}'$ as

$$S_{BC}' = (2^n - 1)/T_H' = [2^d(2^n - 1)S_{BC}]/[2^n S_{BC} + 2^d(2^n - 1)] < 2^d S_{BC}/(S_{BC} + 2^d),$$

In these equations, $T_H$ is the threshold, and $S_{BC}$ is the brightness compensation factor used in EDLS. Thus, $T_H'$ determines actual saturation ratio $S^R$. Because $T_H' \le T_H$, and thus $S'^R \le S^R$, EDLS-*d* achieves a smaller power reduction than EDLS. The difference in the power reduction between EDLS and EDLS-*d* in the DLS mode is equal to $P_B/S_{BC} - P_B/S'_{BC}$, and thus it is bounded by $2^n P_B/[2^d(2^n - 1)]$, where $P_B$ is the original backlight power consumption.

In the same way, we calculate worst-case upper and lower thresholds $T_H'$ and $T_L'$, and the worst-case contrast enhancement factor, $S_{CE}'$, of EDLS-*d* in the DCE mode as

$$T_H' = \lceil T_H, 2^n/2^d \rceil = T_H + 2^n/2^d,$$

$$T_L' = \lfloor T_L, 2^n/2^d \rfloor = T_L - 2^n/2^d,$$

and

$$S_{CE}' = (2^n - 1)/(T_H' - T_L') = [2^d(2^n - 1)S_{CE}]/[2^{n+1}S_{CE} + 2^d(2^n - 1)] < 2^d S_{CE}/(S_{CE} + 2^d),$$

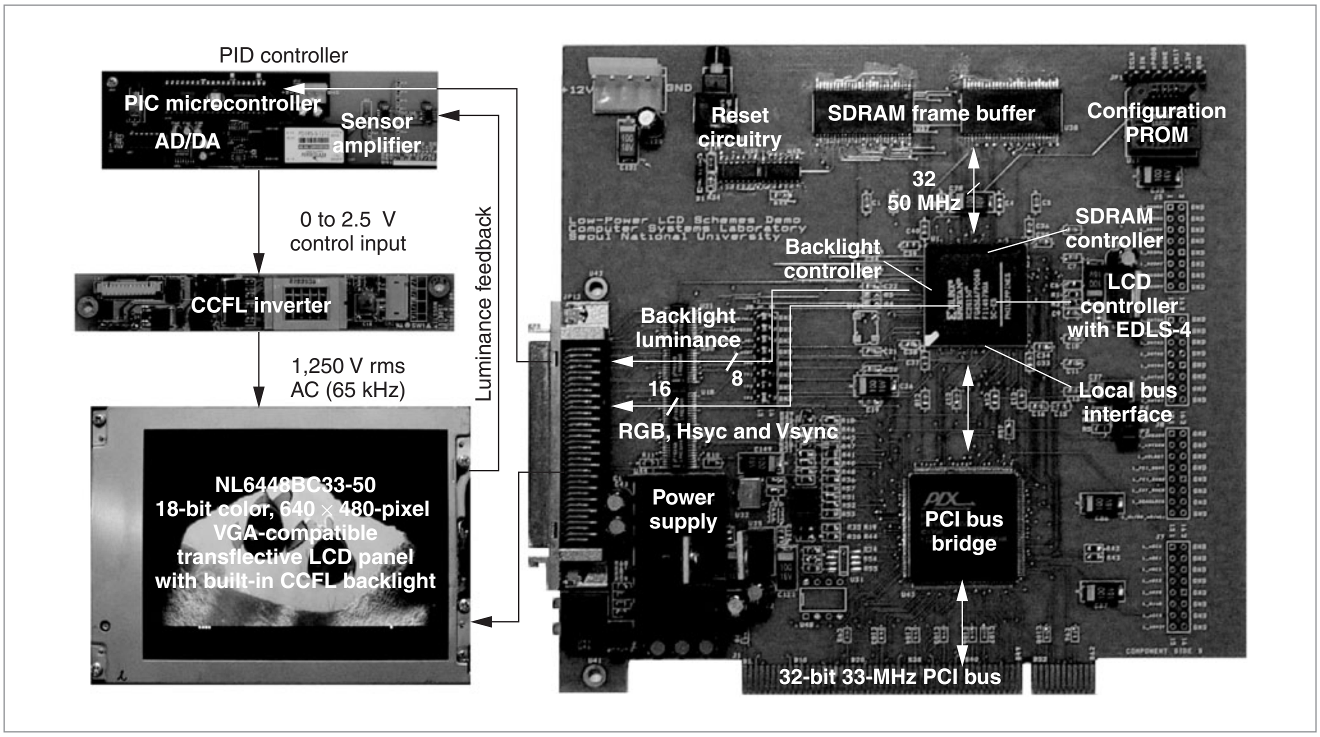

**Figure 3. Block diagram of an application-transparent EDLS prototype.**

where $T_H$ and $T_L$ are the upper and lower thresholds, and $S_{CE}$ is the contrast enhancement factor used in EDLS. Note that $\lfloor A, B \rfloor$ is a floor function of number $A$, which rounds $A$ down to the nearest multiple of significance, $B$. The difference in the power reduction EDLS and EDLS-$d$ achieve in DCE mode equals $P_B/S_{CE} - P_B/S'_{CE}$, and thus it is bounded by $2^{(n+1)}P_B/[2^d(2^n - 1)]$. The backlight power consumption penalty resulting from the EDLS-$d$ approximation is usually much less than the worst-case value we have just calculated.

After determining the parameters of hardware EDLS-$d$ for a given display specification, we can make a further compromise between the energy reduction from the backlight and the area overhead for the EDLS functional blocks. Hardware EDLS-$d$ slightly reduces the energy savings that the backlight dimming achieves, and results in a minor inconsistency in the interframe saturation ratio in video applications, where $d$ is small and the area savings is large. Note that EDLS-$d$ is also applicable to the software-oriented approach, but we would not expect improvement in either the energy or time overhead because the same data path resources in the CPU must perform all the operations.

## Experimental results

We implemented a VGA-compatible LCD controller with application-transparent hardware EDLS-4 at a 640 × 480-pixel screen resolution and 16-bit color depth. It outperformed a software EDLS that we also implemented for comparison.

Figure 3 shows the prototype's architecture. The implementation includes an FPGA EDLS-enabled LCD controller, a peripheral component interconnect (PCI) bus interface, a frame-buffer memory, and a CCFL backlight inverter using a proportional-integral-differential (PID) controller. The LCD controller contains two Samsung K4S641632D SDRAM devices for the frame-buffer memory. The backlight system of an NEC6448BC33-50 10.4-inch TFT LCD panel consumes about 8.1 W at its maximum luminance. Thanks to an effective compaction of the EDLS algorithms, it was possible to mount the EDLS-4 on a small, low-cost Xilinx Spartan-II FPGA, the XC2S-150FG456. The Linux operating system tends to have a slow response time because of its heavy locking mechanism, so a 1-ms timer interrupt to activate the PID controller is not feasible. Instead, we used a simple reduced-instruction-set computing (RISC) microcontroller, the PIC16C74A from Microchip Technology. A VGA-compatible Linux driver (which corresponds to the Linux kernel 2.4.19) supports

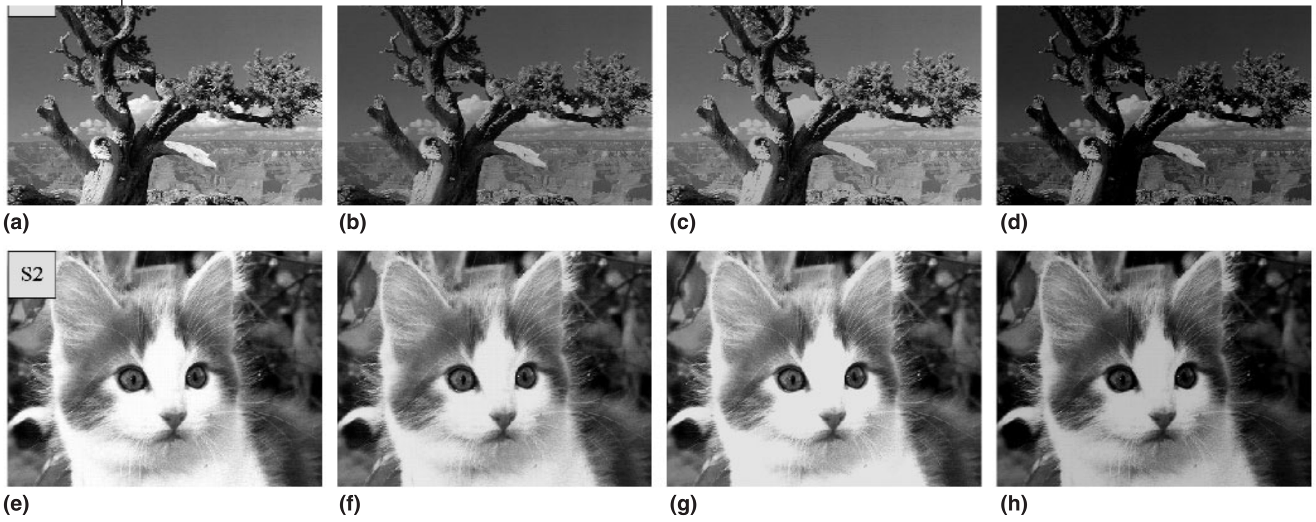

**Figure 4. Two still images before and after application-transparent hardware EDLS-4 ($S^R = 0.3$): in their original state (a, e); with a dimmed backlight, where $L_B' = 0.80L_B$ (b) and $L_B' = 0.94L_B$ (f); produced by EDLS-4 in DLS mode, where $S'^R = 0.12$ and $L_B' = 0.80L_B$ (c), and $S'^R = 0.24$ and $L_B' = 0.94L_B$ (g); and produced by EDLS-4 in DCE mode, where $S'^R = 0.27$ and $L_B' = 0.67L_B$ (d), and $S'^R = 0.27$ and $L_B' = 0.84L_B$ (h).**

**Table 1. Average and variance of backlight power savings for a movie stream (%).**

| Technique | Average power savings (percentage) | Variance (percentage) |
|---|---|---|
| EDLS (DLS) | 20.6 | 38.3 |
| EDLS-4 (DLS) | 18.9 | 43.2 |
| EDLS (DCE) | 32.3 | 58.6 |
| EDLS-4 (DCE) | 32.0 | 58.7 |

the EDLS-enabled LCD controller. The resulting platform can use the EDLS capability for all types of applications that use the LCD display, without any modification.

## Energy reduction and image quality

EDLS reduces the backlight's energy consumption. We will now compare the power reduction achieved by software EDLS and hardware EDLS-4. There is no reason to use software EDLS-*d* where *d* is smaller than the original color depth. Hardware EDLS-*d*, where $d = 4$, is a reasonable configuration when considering hardware complexity. We expect more power reduction by software EDLS under fixed $S^R$ because $S^R > S'^R$; thus, we can use a dimmer backlight with software EDLS. In other words, EDLS-*d* produces an image quality no worse than EDLS, but saves less power.

Figure 4 illustrates the image quality of EDLS-4. Figures 4a and 4e are the original images and Figures 4b and 4f are unprocessed images with a dimmed backlight. We can see that the dimmed backlight reduces both brightness and contrast. We produced Figures 4c and 4g using EDLS-4 in DLS mode with the same amount of backlight dimming as in Figures 4b and 4f. EDLS-4 restores both brightness and contrast to their original values. We can hardly see any image distortion, although it is present. Finally, Figures 4d and 4h are the results of using EDLS-4 in the DCE mode with more backlight dimming and hence reduced power consumption. Although their brightness is less than that of the original, the contrast has been recovered.

Finally, we applied both software EDLS and hardware EDLS-4 to a movie clip, namely a trailer for the movie *Bad Boys 2*. Table 1 summarizes the average power reduction and variance, where $S^R = 0.2$. This example shows that EDLS-4 produces significant results in a real situation.

## Power, delay, and area overhead

Although EDLS significantly reduces backlight power consumption, it involves power, delay, and area overheads that take place in other components. These overheads are primarily determined by the screen resolution, refresh rate, and color depth. Typically, EDLS must cope with a 30-Hz refresh rate for quality movie streams.

Thus, application-transparent software EDLS occupies a 36.9-Mbps data bandwidth to refresh the histogram at a 640 × 480-pixel resolution and 16-bit color depth. Even though it is not an expensive setting in modern applications, application-transparent software EDLS requires an over 300% usage of a 733-MHz XScale processor. That would imply a 240-mW power overhead if it were feasible. This shows that application-transparent software EDLS is only applicable to low screen resolutions.

On the other hand, power and area overheads for hardware EDLS-*d* are not sensitive to screen resolution; they are only sensitive to *d*. Table 2 summarizes the overhead for histogram construction, which is a primary concern for hardware EDLS-*d*. Image enhancement is not a serious overhead in hardware EDLS-*d*, but it takes most of the CPU and memory resources in software EDLS. Image processing requires additional data path resources such as multipliers, adders, and comparators; however, just three 13-bit precision integer multipliers can manage the image processing for 16-bit color. The multipliers for image processing require just 77 slices, which corresponds to an area overhead of 9%. Furthermore, the LCD controller prototype's power consumption increases by only 6 mW because of image processing.

**Table 2. Power and area overheads of EDLS-*d*.**

| EDLS-*d* | No. of slices | Equivalent no. of gates | FPGA core power (mW) | LCD controller's total power (mW) |
|---|---|---|---|---|
| Without EDLS | 926 | 64,656 | 229 | 1,313 |
| EDLS-1 | 1,033 | 66,596 | 239 | 1,328 |
| EDLS-2 | 1,121 | 68,356 | 248 | 1,340 |
| EDLS-3 | 1,266 | 71,372 | 260 | 1,361 |
| EDLS-4 | 1,574 | 77,578 | 284 | 1,392 |

**THE EDLS FRAMEWORK** is applicable to most battery-operated mobile multimedia terminal devices. The MPEG-21 multimedia framework initiative aims to support a wide range of networks and devices in the delivery and consumption chain of their multimedia resources. MPEG-21 digital item adaptation (DIA) can also help save power in terminal devices, though the framework is not primarily designed for power reduction, and DIA defines only limited power awareness. As future work, the EDLS framework under MPEG-21 DIA will offer several promising opportunities for power savings in terminal devices as well. ■

## Acknowledgments

The Research Institute of Computer Technology at Seoul National University provided research facilities for this study. The Brain Korea 21 Project also supported this work.

## References

1. I. Choi, H. Shim, and N. Chang, "Low-Power Color TFT LCD Display for Hand-Held Embedded Systems," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 02), ACM Press, 2002, pp. 112-117.
2. I. Choi et al., "LPBP: Low-Power Basis Profile of the Java 2 Micro Edition," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 03), ACM Press, 2003, pp. 36-39.
3. F. Gatti et al., "Low Power Control Techniques for TFT LCD Displays," *Proc. Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Systems* (CASES 02), ACM Press, 2002, pp. 218-224.
4. W.-C. Cheng, Y. Hou, and M. Pedram, "Power Minimization in a Backlit TFT-LCD Display by Concurrent Brightness and Contrast Scaling," *Proc. Design, Automation and Test in Europe* (DATE 04), IEEE CS Press, 2004, pp. 252-257.
5. S. Pasricha et al., "Reducing Backlight Power Consumption for Streaming Video Applications on Mobile Handheld Devices," *Proc. First Workshop on Embedded Systems for Real-Time Multimedia* (ESTIMedia 03); http://www.cecs.uci.edu/conference_proceedings/esti_f.pdf.
6. *Display Modes (Transmissive/Reflective/Transflective)*, Sharp Microelectronics of the Americas, 2002; http://www.sharpsma.com/sma/Products/displays/AppRefGuide/DisplayModes.htm.
7. T. Tsukuda, *TFT/LCD: Liquid-Crystal Displays Addressed by Thin-Film Transistors*, Taylor & Francis, 1996.

**Hojun Shim** is a PhD candidate in the School of Computer Science and Engineering at Seoul National University, Korea. His research interests include low-power and embedded systems. Shim has a BS in computer engineering from Seoul National University. He is a student member of the IEEE and a SIGDA member of the ACM.

**Naehyuck Chang** is an associate professor in the School of Computer Science and Engineering at Seoul National University. His research interests include system-level low-power design and embedded systems design. Chang has a BS, an MS, and a PhD in control and instrumentation engineering from Seoul National University. He is a member of the IEEE and the ACM.

**Massoud Pedram** is a professor in the Department of Electrical Engineering Systems at the University of Southern California. His research interests include computer-aided-design methodologies and techniques for low-power design, synthesis, and physical design. Pedram has a BS in electrical engineering from the California Institute of Technology, and an MS and a PhD in electrical engineering and computer science from the University of California, Berkeley. He is a Fellow of the IEEE, an associate editor of the *IEEE Transactions on Computer-Aided Design*, and the IEEE Circuits and Systems Society Distinguished Lecturer Program Chair.

■ Direct questions and comments about this article to Naehyuck Chang, School of Computer Science and Engineering, Seoul National University, Shilim, Kwanak, Seoul, 151-010, Korea; naehyuck@snu.ac.kr.