Runtime Power Estimation of Mobile AMOLED Displays

Dongwon Kim, Wonwoo Jung, Hojung Cha
Department of Computer Science
Yonsei University, Korea
{dwkim,wwjung,hjcha}@cs.yonsei.ac.kr

***Abstract*— Modeling and estimating power consumption of OLED displays are necessary to understand the energy behavior of emerging mobile devices. Although previous study exists to model and estimate the power consumption of stationary display images, to the best of our knowledge, no prior work is found to deal with runtime power behavior of OLED display running real applications. This paper proposes a runtime power estimation scheme for OLED displays that involves monitoring kernel activities that capture the screen change events of running applications. The experiment results show that the proposed scheme estimates the display energy consumption of running applications with reasonable accuracy.**

***Index Terms*— Power, energy, modeling, estimation**

I. Introduction

Energy estimation techniques for mobile devices have recently been studied actively. Such techniques are often centered on hardware components that are associated with a significant amount of energy consumption of overall device energy [1-2]. Display components typically consume the largest portion of energy when users actively interact with devices [3]. Both liquid crystal displays (LCD) and organic light-emitting diodes (OLED) displays are widely used for the display components of mobile devices. Most of the previous work on display power models has been mainly focused on the LCD types [1-2]. However, to understand the energy behavior of modern devices, it is important to model OLED displays, especially the active-matrix OLED (AMOLED[1]) types, which are becoming increasingly popular in smartphones.

Modeling OLED energy consumption is more difficult than the LCD types, because the power consumption characteristics of OLED are complex due to its light emitting characteristics. The LCD power model is often represented as a simple linear form, just considering the backlight level [1-3]. However, the OLED power model should consider the RGB values of sub-pixels. This naturally leads to computation overhead to estimate the energy. In addition, tracing the runtime power behavior of OLED displays is not trivial. With LCD, the power changes are easily detected by monitoring system events that alter the backlight level [1-2,4]. In the case of OLED displays, the power changes should be dynamically detected whenever the screen contents are updated.

Previous work [5-7] presented power models and energy estimation schemes for OLED displays. The power models were obtained by experimental methods and showed convincing results for their target displays. However, prior estimation schemes have mainly dealt with *stationary* screen images and have been validated with simple applications. We believe that the previous methods are not suitable for estimating the *runtime* display power while arbitrary applications are running in the device.

In this paper, we present a lightweight and runtime screen monitoring scheme that traces changes in display power consumption of running application, by monitoring system activities at the kernel level.

II. Runtime Display Power Estimation

For the OLED displays, the power consumption depends on the color distributions in the frame buffer (i.e., the screen color images). Hence, in order to estimate the display power consumption for any running application, the color changes in frame buffer should be detected dynamically whenever the screen contents are updated.

Several methods exist to detect the change of contents in frame buffer at runtime. A simple scheme is to poll the contents of frame buffer periodically. The right selection of polling period in this method is, however, practically difficult because tradeoffs exist between polling overhead and detection accuracy in color changes. Another mechanism to estimate the color changes in frame buffer is to use hints from the user interactions, such as touch events, which often cause the changes in screen color images. However, this scheme is not complete in its own, since the touch events do not necessarily cause the changes of screen image in some applications, and also the screen image can be altered without the touch event in other cases.

In our work, we used the Android-specific system operations to detect the changes in frame buffer at runtime. The Android framework uses *SurfaceManager* to combine the frames, construct the buffer, and project the frames onto the screen. *SurfaceManager* uses Android binder [8] to manage the frame buffer in the Linux kernel. When an application requests a display switch, the application uses RPC via Android binder. To execute *SurfaceManager*, the BC_TRANSACTION command is sent to the binder driver with *binder_transaction()*. Thus, our scheme analyzes the data processed in

[1] In this paper, OLED refers to AMOLED.

978-3-9815370-0-0/DATE13/©2013 EDAA

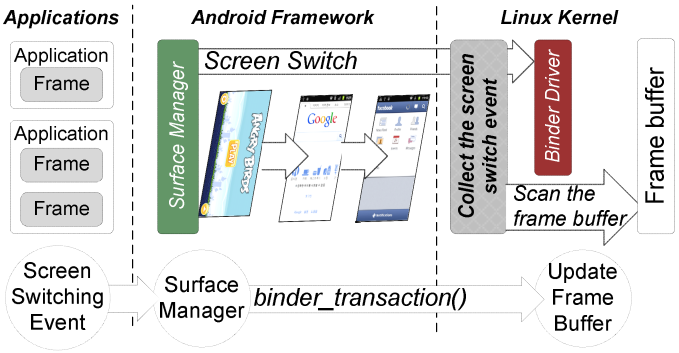


Fig. 1. Runtime estimation scheme

binder_transaction() and collects data on the screen transitions. We implemented the mechanism of detecting the binder transaction as a loadable module using the *Kprobes* facility [9]. Based on *Kprobes*, our scheme detects and analyzes the *SurfaceManager* operation. Fig. 1 illustrates the overview of the proposed mechanism. This runtime scheme is generic enough to be applied to any running application and any system situation, hence estimates the display power consumption adequately.

Efficient monitoring of frequent binder transaction is critical in our scheme. We certainly require a simple mechanism to access the frame buffer. The idea is to access the frame buffer only at the "last event." During a short sliding operation on a launcher application, *SurfaceManager* operates until the sliding operation is done, causing a heavy binder transaction call. For a series of binder transactions, we access the frame buffer and calculate the power consumption only at the last event. To distinguish the events, we delay each event with a time window. If no event occurs in a time window, the event is regarded as the last one. On the other hand, in some situations, the last event may not be detected. Applications, such as games, cause continuous changes of display contents, making the suggested method inapplicable. In such situations, we apply the periodic scan process instead.

Meanwhile, in order to estimate the power consumption of a display screen at stationary points, we need to obtain the RGB pixel data dynamically. The estimation cost depends on the display resolution; that is, the number of pixels that are stored in a frame buffer. The calculation cost and overhead for accessing the frame buffer should be included in the estimation cost. Instead of scanning every pixel in the frame buffer, we use a grid-based partial scanning method, where the center pixel of each grid represents the color characteristics of the segment. This way, we scan only a portion of the frame buffer, although the accuracy is degraded. Here, we need to decide the grid size by considering the tradeoff between computation overhead and estimation accuracy. The computation overhead increases linearly with grid size and scan frequency. The grid size should therefore be decided by considering accuracy, overhead, and scan frequency

III. Experiments

For the experiments, we used the Samsung Galaxy S3 (S3) smartphone [10]. S3 has a 4.8-inch Super AMOLED display

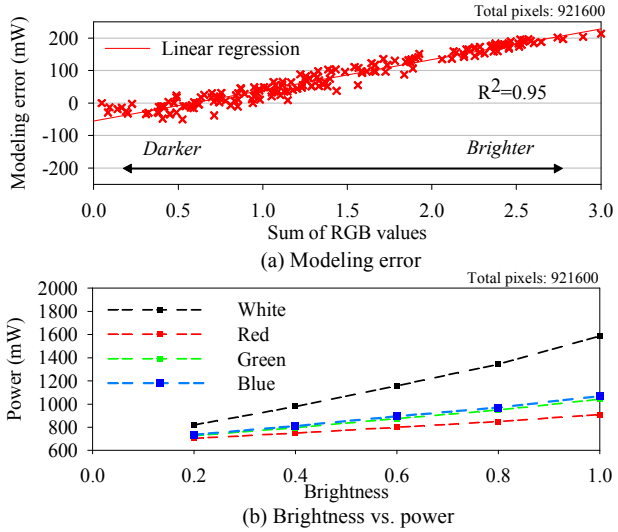


Fig. 2. Experiments on AMOLED display power consumption

with a display resolution of 720×1280 WXGA. The device uses 24-bits of sRGB color space and employs the PenTile matrix technology. Device power consumption was measured with a Monsoon Power Meter [11], and all experiments were conducted without activating the WiFi or cell network components. The power consumption of the OLED display was obtained by subtracting the system power consumption (including CPU) from the overall power consumption of the device. Refer [12] for details on the measurement method.

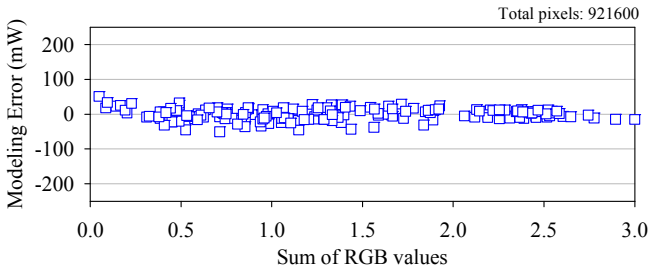
A. The AMOLED Power Model

Although prior work exists to model and estimate OLED power consumption, our preliminary experiments showed that the models are not accurate for the displays equipped in our smartphones. According to previous studies [5-7], the power model for OLED displays can be represented as follows:

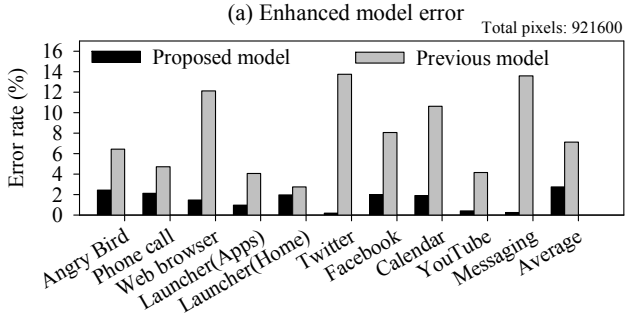
$$P_{oled} = C + \sum_{i=1}^{n}\{f(R_i) + h(G_i) + k(B_i)\}, \quad (1)$$

where C is the base power of the OLED component; that is, C is obtained with the screen being activated with the *black* color. The total number of pixels is represented by n, and i is the pixel index. The power consumptions of the R, G, and B sub-pixels are represented by $f(R_i), h(G_i)$, and $k(B_i)$, respectively, and the R_i, G_i, and B_i values are transformed into a linear RGB format.

To validate the power model on our devices, we measured the display power consumption with random combinations of RGB values and then compared it with the estimation obtained with Equation (1). Fig. 2 (a) shows the modeling error for S3. The X-axis of the graph represents the sum of RGB values, which ranges from 0 to 3, where each R/G/B ranges from 0 to 1. The Y-axis represents the modeling error (in mW), which is defined as the difference between the real measurement and the estimated value over the whole pixels. The graphs show that the error increases linearly as the screen color becomes brighter (i.e., as the sum of RGB increases). In the case of the color white (i.e., the RGB sum is 3), the modeling error rate reaches



(a) Enhanced model error



(b) Error rates for real applications

Fig. 3. Validating the modified power model

about 13.7% for S3. Note that the color white is frequently used in smartphone applications, and incorrect modeling would cause a large estimation error. We also experimented with the display power consumption under varying levels of brightness. Fig. 2 (b) shows that the power consumption of the OLED display is linearly proportional to the brightness level, as long as the RGB values are fixed.

Based on the preliminary experiments, we used a slightly modified power model by reflecting both the sum of RGB values and the brightness level in the original model. First, in addition to the individual R/G/B power characteristics, we also considered the summation characteristics of RGB values, which are obtained with a linear regression technique of $e(R, G, B) = a(R+G+B) + b$, where a and b are the constants that are experimentally acquired with the target device. That is, the linear regression lines drawn in Fig. 2 (a) determine the slope a and the y-intercept b of the function. The modified power model is now as follows:

$$P_{oled} = C + Br \cdot \Sigma_{i=1}^{n}\left((\beta_R R_i + \beta_G G_i + \beta_B B_i) + eRi,Gi,Bi,\right) \quad (2)$$

Here, Br is the brightness level ranging from 0 to 1. $f(R), h(G)$, and $k(B)$ used in the previous model are now represented as the β_R, β_G, and β_B coefficients.

We conducted various experiments to evaluate the accuracy of the modified model. First, we experimented with single-colored screens. We used the same RGB values to compare the performance of the new model with the previous one. Fig. 3(a) shows the error rate of the modified power model for S3. The error rate is defined as the ratio between the real measurement and the estimated one over the whole pixels. The average error rate is approximately 1.4%, which shows an enhancement over the previous model of 6.9%. In particular, the white-colored screen is now reduced to 0.9% from the previous model of 13.7%.

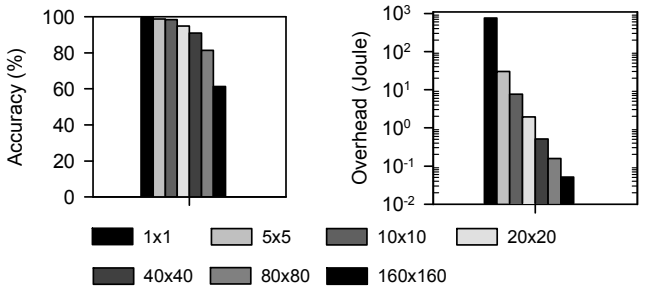


Fig. 4. Grid-based partial scanning scheme: accuracy and overhead

Next, we evaluated the scheme with more realistic scenarios. We benchmarked a series of screen images captured from real applications, including the launcher (home and application list), web browser, Angry Birds, phone call, Twitter, Facebook, calendar, YouTube, and messaging. Fig. 3(b) shows the error rates. In all cases of running real applications, the error rate of the modified model was less than that of the previous one. The average error rates of the new one were 1.4%, whereas the error rate of the previous model was 8.0%.

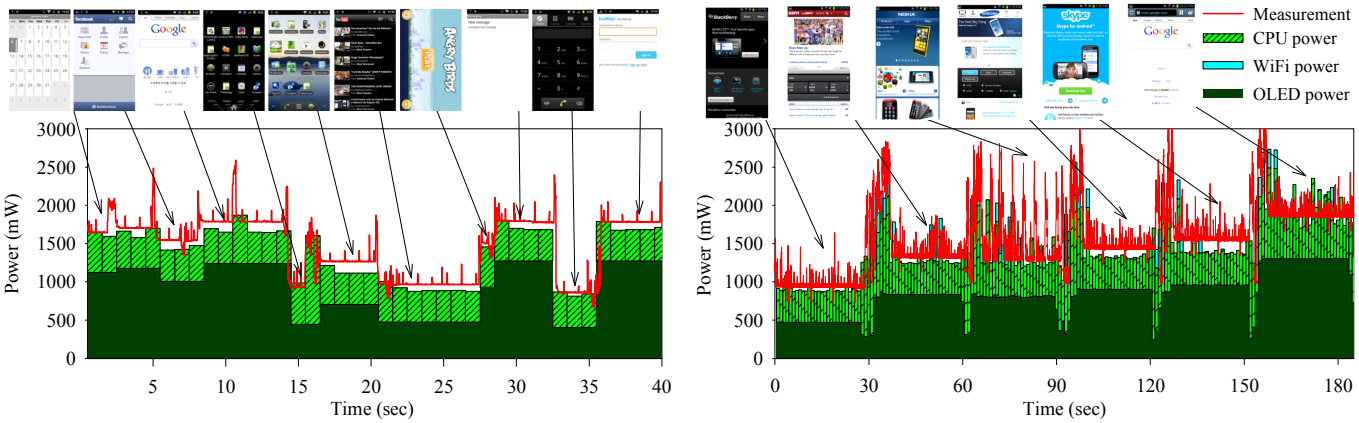
B. Accuracy and Overhead

We now conducted experiments to validate the feasibility of the proposed runtime energy estimation scheme. First, to evaluate the performance of the grid-based partial scanning scheme, we experimented with real applications varying in grid size. Fig. 4 shows the accuracy and overhead of the proposed scheme. The accuracy is estimated by color difference, and the overhead is measured by the Monsoon power meter. Fig. 4 shows that an increased grid size generally decreases accuracy, but in some cases, the error is small even with a larger grid size. This is caused by the over-fitting characteristics of the grid-based partial scanning scheme. Fig. 4 shows the grid-scanning overhead distribution for each grid size. Overall, the performance results experimentally validate the tradeoff between computation overhead and estimation accuracy.

Next, we prepared a test scenario, which sequentially switches the static screen images captured from a series of real application. Fig. 5(a) shows the runtime results with 40x40 grids. The arrows point to the timeline where each image is displayed on the screen. Note that the CPU power consumption is estimated by its utilization [1]. The display power consumption is shown to be static for each image, because the power state of smartphone is basically idle just displaying images, without doing anything in the system. Our results show that the estimated display power closely matches with the actual device power consumption measured with Monsoon power meter. The sharp edge in the power trace is caused by the CPU when the screen changes. The overall error rate is 5.6%, or 2.4 J/40 seconds.

C. Monitoring the real applications

We applied the proposed model and estimation technique for OLED display to AppScope [1], which is an energy metering tool to collect fine-grained process-specific energy information for Android smartphones. AppScope monitors



(a) Test scenario and runtime power behavior

(b) Monitoring the real applications

Fig. 5. Runtime power estimation on S3

application's hardware usage, such as CPU, WiFi, 3G, and GPS at the kernel level, and accurately estimates energy consumption. With the AppScope integration, we are now able to trace the runtime power behavior of OLED display, as well as the power traces for other hardware components, while running actual Android applications.

Fig. 5(b) shows the detailed traces of device power consumption while visiting the home pages of six websites using the Google browser. The bar segments in each timeline represent the estimated power consumption of OLED, WiFi, and CPU while running the browser. The red lines represent the actual power trace obtained with the Monsoon power meter. For the Internet connection, we activated WiFi, instead of 3G. We stayed at each website for about 30 seconds, and moved on to the next site. The AppScope traces in Fig. 5(b) show that the estimated power consumption of device is closely matched to the actual measurement. In particular, for each website change, the power consumption of the hardware components exhibits similar patterns. That is, the display power is temporarily reduced, due to the brief change of screen into the window which lists the favorite sites (i.e., white texts on black background). The WiFi component actively retrieves new web contents during this change. The processor is also engaged in processing the new contents, hence spending additional CPU cycles.

Overall, the experiment results show that our scheme accurately observes the screen changes at runtime, and adequately estimates the OLED power consumption for real applications. The estimated energy consumption for the period of 180 seconds is about 7J, which exhibits approximately 2.2% of error rate.

IV. Conclusions and Future Work

We presented a runtime power estimation scheme for mobile OLED displays. Based on kernel activity monitoring, the proposed scheme provided accurate estimation results for running applications with low overhead.

Based on our current results, we are planning the following work. First, the grid-based scanning technique needs to be enhanced. Although the proposed method showed good accuracy, the scheme should be further enhanced to cover the uneven distribution of color in a grid. Second, the *SurfaceManager*-based monitoring scheme is currently unable to detect the change of continuous video images. The monitoring scheme should be supplemented for video application.

Acknowledgements

This work was supported by a grant from the National Research Foundation of Korea (NRF), funded by the Korean government, Ministry of Education, Science and Technology under Grant (No.2012-0005522).

References

[1] C. Yoon, D. Kim, W. Jung, C. Kang and H. Cha. AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity. In *USENIX ATC*, 2012.

[2] L. Zhang et al. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *CODES+ISSS*, 2010.

[3] A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *USENIX ATC*, 2010.

[4] W. Jung, C. Kang, C. Yoon, D. Kim, H. Cha. DevScope: a nonintrusive and online power analysis tool for smartphone hardware components. In *CODES+ISSS*, 2012

[5] M. Dong, Y. K. Choi and L. Zhong. Power Modeling of Graphical User Interfaces on OLED Displays. In *DAC*, 2009.

[6] M. Dong and L. Zhong. Chameleon: A Color-Adaptive Web Browser for Mobile OLED Displays. In *MobiSys*, 2011.

[7] X. Chen et al. Quality-retaining OLED Dynamic Voltage Scaling for Video Streaming Applications on Mobile Devices. In *DAC*, 2012.

[8] Android Binder, https://www.nds.rub.de/media/attachments/files/2011/10/main.pdf retrieved on 7/20/2012

[9] Kprobes, http://www.kernel.org/doc/Documentation/kprobes.txt retrieved on 7/20/2012

[10] Samsung I9300 Galaxy S III, http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php retrieved on 7/20/2012

[11] Monsoon Solutions, Inc., http://www.msoon.com/LabEquipment/PowerMonitor/ retrieved on 6/17/2012