

Using Checksum to Reduce Power Consumption of Display Systems for Low-Motion Content

Kyungtae Han, Zhen Fang, Paul Diefenbaugh, Richard Forand, Ravi R. Iyer, Donald Newell

Intel Labs, Hillsboro, OR 97124

{*kyungtae.han, zhen.fang, paul.s.diefenbaugh, richard.a.forand, ravishankar.iyer, donald.newell*}@intel.com

Abstract—Power consumption of the display subsystem has been a relatively less explored area compared to other components of a mobile device including computing, storage, and networking units, although the former often constitutes one of the most power-hungry portions of the system. Typical applications on a mobile device such as web browsing and text editing tend to have rather static image content; each frame hardly changes from the previous one. Efficiently detecting and handling no-motion scenarios is thus critical to extend the battery life. This paper focuses on image change detection.

We propose to use checksum to detect image changes. Specifically, CRC hardware is used to optimize the power consumption of 1) refresh of a local display and 2) data compression for wireless remote display. Compared with a traditional, pixel-by-pixel comparison approach, using checksum for image change detection is not only fast, but also reduces accesses to the frame buffer, resulting in significant power savings. We have built a FPGA prototype to verify that CRC can capture image changes well enough to ensure a “visually lossless” quality.

I. BACKGROUND

For battery-powered platforms, while much research effort of the computer architecture community has been focused on reducing the power consumption of the microprocessor, the caches, system memory, and the network, relatively less attention has been devoted to the display subsystem, although the latter actually constitutes one of the most power-hungry portions of the platform [1], [2]. In this paper, we propose a novel mechanism that can optimize the power consumption of two representative display subsystems: 1) a local display 2) a wireless remote display.

A. Local display refresh

Mobile platforms are typically used for web browsing, email, document viewing, etc. In these usage cases, the screen images are usually rather static [3]; most of the frames are identical to the preceding ones. However, in order to hold images stable on the screen, current display controllers still have to refresh the display at a fixed rate. Pixel data are fetched across the LCD interface link from the frame buffer which is typically part of system DRAM. Frame buffer accesses and LCD interface link transfers caused by display refresh constitute a big portion of the system power on a mobile platform [4].

Instead of fetching pixel data from the system frame buffer, the display controller could refresh a panel through a local frame buffer to save power consumption in the host

when the images to be displayed are same as the image which has already been displayed and stored in the local buffer [4], [5]. Fig. 1 shows the system organization of such a LCD display system. Dotted lines denote the components whose power could be saved at the system level if we can set the LCD to self-refresh mode. The controller can set the self-refresh signal only when the system is inactive for a long time, e.g., after no activity from the keyboard and mouse for 2 minutes. With self-refresh turned off, huge amount of data are still transferred to the LCD 60 frames/second, even though most of these frames are identical to each other.

Researchers have been trying to improve energy efficiency of the display system by dimming the LCD panel backlight [6], by dimming the screen portions that the user is not focusing on [7], by reducing power consumption of the inactive windows [8], and by adaptively changing the refresh rates and color depth [9], among many other innovations. Our work is orthogonal to these techniques.

B. Wireless Remote Display and Display Content Compression

To compensate for the small screen of mobile internet devices (MIDs), high-definition wireless remote display is emerging as a promising solution. In this usage model, the application runs on a user’s handheld device, while the picture frames are sent to a large intelligent display nearby through wireless links.

There are different mechanisms to intercept and transfer command/data from the host (i.e., sender) to the display (i.e., client, or receiver). *Screen scraping* is one of them. Instead of sending commands for the client to re-draw the graphics, it sends pre-rendered frames from the application server to the client display. Because of its advantage in rendering 3D frames on a thin client, the screen scraping approach has been used by a number of cross-platform remote display tools, such as VNC [10]. There are extensions to the basic “pixel scan” approach of VNC. For example, Microsoft RDP [11] and Citrix ICA [12] employ more complex protocols to reduce bit rate and enable special visual effects. Although these advanced protocols have obvious advantages for traditional PC-based remote terminals, we believe that they are less likely to become standards for thin display clients because they would require rather complex functionalities in the displays.

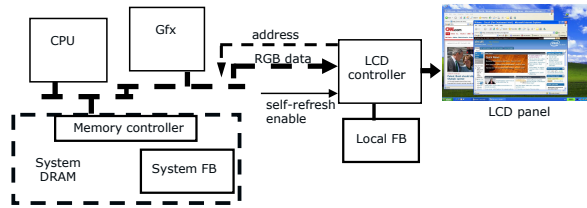


Fig. 1. Diagram of a Display System with a Local Frame Buffer

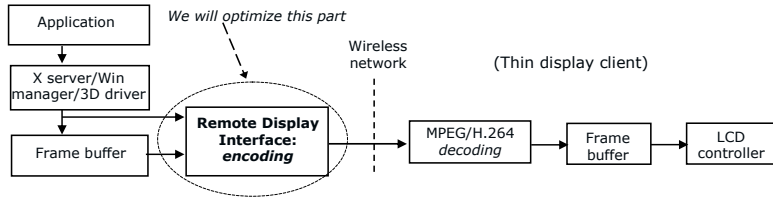


Fig. 2. Screen Scraping Based Wireless Remote Display

The transmission speed requirement for HD display is prohibitively high for current and near-future radio on a mobile device. For example, 1920x1080 pixels @ 24 frames/second, 24 bits/pixel would need over 1Gb/s of sustained wireless bandwidth from the MID to the display. Some form of data compression is necessary to reduce both network bandwidth requirement and transmission radio power. Most intelligent displays in the future are expected to have hardware MPEG/H.264 decoders for video playback. At the same time, camera, together with video encoding accelerators, will just be standard components in MIDs. Availability of video encoding hardware provides a handy data compression utility: motion estimation of MPEG and H.264 is designed to significantly reduce temporal redundancy between two frames, and can be used for wireless remote display data compression. Fig. 2 shows the baseline wireless remote display system.

Motion estimation (ME) is an inter-frame predictive coding technique used to eliminate the large amount of temporal redundancy that exists between video frames. Suppose a reference frame has been encoded and we are trying to encode the current frame. Instead of directly storing and transmitting every pixel, MPEG/H.264 only transmit the “delta” between the current macroblock (MB, 16x16 pixels) and an MB in an earlier frame [13], [14]. ME consists of two steps: 1) finding a MB in an earlier frame that is most similar to the current MB; 2) generate a pointer and a 16x16 matrix as the delta than the reference MB. The similarity test used in most encoders is Sum-of-Absolute-Differences, or SAD. The pointer, called motion vector (MV), gives the relative coordinates between the MBs. The 16x16 residual matrix gives the difference between the reference MB and current MB. The first step is the most time-consuming step in MPEG and H.264 encoding. The computational intensity of the similarity test is perhaps the biggest hurdle to wide deployment of real-time, low-cost H.264 hardware encoder.

For wireless remote display, a decent pixel search for ME alone can violate the delay target. As a result, we can only afford trying one macroblock in the reference frame. This candidate MB is the MB in the reference frame with the same coordinates as the current MB that is being encoded. If the current MB in the current frame is identical to the candidate MB in the reference frame, we encode the current MB as a P- or B-macroblock to exploit the temporal redundancy. Otherwise, encode the current MB as an I-macroblock, only exploiting intra-MB data compression. Except for usage cases like gaming and movie playback, most portions of the screen content is usually very static. Most MBs will end up being encoded as P- or B-macroblocks using just a pointer to the same MB in its reference frame and an all-zero residual matrix. This case can be encoded very efficiently in MPEG and H.264, i.e., achieving excellent compression ratios.

However, to decide that a current MB has no motion from the reference frame, pixel-by-pixel comparisons have to be performed in MPEG and H.264 search functions. This entails frequent accesses to the frame buffer, which is usually in DRAM. In an aggressively power-optimized DRAM system of a handheld device, maximum residency in DRAM standby and self-refresh modes is essential to save energy. Frequent frame buffer accesses will cause the DRAM ranks to constantly wake up from low-power modes.

C. Checksum/CRC

Cyclic redundancy check [15] is a type of data integrity checksum function that takes as input a data stream of any length and produces as output a value of a certain fixed size. Usually checksums, including CRC, are used to detect accidental alteration of data during transmission or storage. An n -bit CRC, applied to a data block of arbitrary length, can detect any single error burst that is not longer than n bits (i.e., any single alteration that spans no more than n bits of the data), and will detect a fraction $1 - 2^{-n}$ all longer error bursts. CRCs are very simple to implement in hardware. For

example, we implemented 8-bit CRC with 52 gates. Besides CRC, many other hash and checksum functions have similar properties in their ability to capture random bit changes. For simplicity, for the rest of the paper we assume that CRC is used as the checksum function.

II. USING CHECKSUM TO DETECT CHANGES BETWEEN FRAMES

Given two blocks of data A and B, generate the CRC code of them, CRC_A and CRC_B . If the exclusive-OR value between CRC_A and CRC_B is equal to zero, we know that there is high probability that A and B are identical. Otherwise, we know that the two data blocks are different.

Let A and B be two successive frames that are sent to the LCD panel, then this CRC code check can serve as an image change detector between frames. Section II-A discusses this embodiment of the idea.

Let A and B be two macroblocks in motion estimation, one in the current frame and the other in the reference frame, then the CRC code check can serve as a zero-motion detector between macroblocks. We elaborate on this application of the idea in Section II-B.

Note that in this paper, we are not proposing any innovative low-power modes of the display controller, display panel, or any new video compression algorithm. Rather, the contribution is a novel *image change detection* technique that enables better utilization of these existing low-power modes and data compression algorithms. Though in this paper the output of CRC unit is used to control self-refresh mode entry/exit and feed into motion estimation function, it can be used for other purposes, albeit with different performance and power implications. For example, if we have detected that M out of N successive frames are identical to their previous ones, we could lower the display refresh rate from 60Hz to 30Hz, like in [9].

A. Stream Image Change Detector used for Display Refresh

The proposed Stream Image Change Detector has three components: CRC Generator, Delay, and CRC Checker, shown in Fig. 3. Frame pixel data stream through the CRC generator at the pixel clock rate. The CRC generator block generates one value for each frame at the frame rate which is same as the vertical synchronization (VSYNC) clock. The Delay block stores the generated CRC value so that it can be compared against the CRC value of the next frame. The CRC checker XORs the stored CRC value in the Delay block and the new CRC value for the current frame. The CRC checker generates the logical value of zero, when both CRCs are equal. Otherwise, a logical value of one is given, indicating that the current frame is different than the previous one. The CRC generator is reset at every vsync signal to clean internal values. The 1-bit change/no-change verdict serves as the display self refresh-enable signal to the LCD controller. After a no-change signal is asserted by the CRC checker, the LCD controller stops fetching data from the system frame

buffer. It goes back to normal refresh mode when the CRC checker de-asserts the no-change signal. The stream image change detector can detect image changes very efficiently and at extremely low cost.

- It does not throttle the throughput of the display; it only introduces a few pixels worth of initial delay.
- The silicon cost and power consumption of the extra hardware is negligible.
- It does not require any change to existing graphics driver, graphics engine, frame buffer, or the LCD controller.

Compared against using pixel-wise comparisons, the most important benefit of this work is that it reduces frame buffer accesses caused by the unnecessary display refreshes, enabling the DRAM ranks to stay longer in lower-power modes such as standby or self-refresh. For static image contents, eliminating most of the display refresh-induced frame buffer accesses helps to significantly increase DRAM devices' residency in low power modes, which can easily have 3X to 10X lower power consumption than an idle standby mode [4]. Secondary benefits include power savings from reduced activities on interconnects such as FSB and IO busses. These power saving opportunities are denoted as dotted lines in Fig. 3.

B. Macroblock-Level Zero-Motion Detector used for Wireless Remote Display Data Compression

We propose to use checksum to detect zero-motion between the current MB that is being encoded and the same MB in the previous frame. This is before motion vectors are derived. If the checksum comparison indicates zero-motion between both MBs, then we literally eliminate the high cost of MB search. This can be thought of as an optimization to the early termination mechanism of existing motion estimation algorithms.

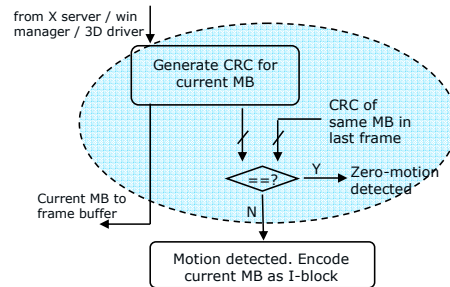


Fig. 4. Zero-Motion Detection Using Hardware CRC

Fig. 4 shows the basic flow of the algorithm. A CRC is calculated by hardware for each MB as it is written to frame buffer. Note that we do not fetch pixels from frame buffer in order to compute their CRC values. Rather, the pixel data are intercepted, e.g., from the 3D driver. The CRC value is stored in on-chip memory, and at the same time XOR-ed against the CRC value of the same MB of the previous

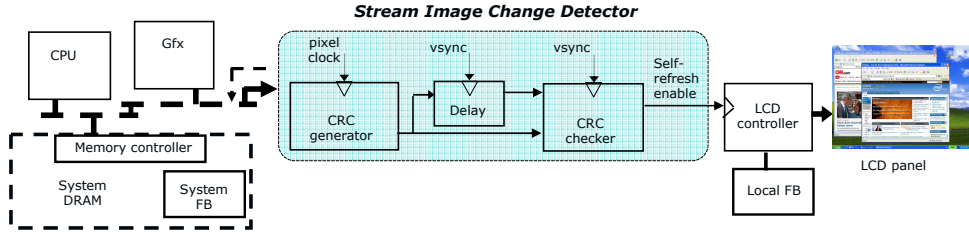


Fig. 3. Stream Image Change Detection Using Hardware CRC

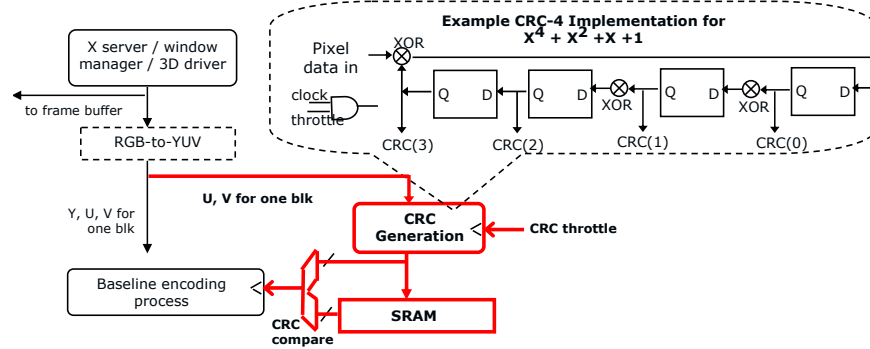


Fig. 5. A Hardware View of the CRC Unit-Augmented Zero-Motion Detector

frame in time, loaded from on-chip memory. Depending on the XOR result, we take one of two actions –

- XOR returns 0: This indicates that both MBs are identical. We derive a motion vector of (0, 0) and all-zero residual matrix for the current MB.
- XOR returns 1: There is difference (i.e., motion) between the two blocks. We need to start the baseline encoding process.

CRC values are compared between two adjacent frames in time, regardless of whether the previous frame is an I-frame, a P-frame, or a B-frame. The on-chip memory size is proportional to the supported picture resolution. For example, supporting 1600x1200-pixel pictures requires 7.3KB (=1600x1200/256) of SRAM storage. SRAM of this size consumes only a fraction of DRAM power. Note that the original frame data would require an impractical SRAM size (e.g., multi-M bytes). Eliminating the need to retrieve reference frame pixel data from DRAM helps to achieve significant power savings.

Partial CRC

To further reduce the CRC calculation overhead, we can ignore some of pixel components between the previous and current frame. For example, when encoding from 4:2:0 YUV, data ratio between Y (the brightness) and U/V components (the colors) is 2:1. If we only use the U and V components and skip checking Y components, the amount of CRC computation and the SRAM storage will both be cut to only 1/3 of the basic method. The intuition behind this optimization is that whenever a pixel changes, all the three values, Y, U and V, change. Emulation is work in progress to prove that from a human perception perspective, partial CRCs can capture

motion as confidently as full-coverage CRCs.

CRC Throttling

In the case when XOR returns 1, CRC adds an extra amount of computation and does not generate any benefit. To mitigate this overhead, we can introduce some CRC throttling mechanism that turns the CRC unit on and off, adapting to the amount of motion in the video. One possible implementation is to turn off CRC after the total number of XOR = 0 for R successive frames is no more than S , and turn on CRC after a (0, 0) motion vector has been generated for T successive frames. With this optimization, motion-rich contents will cause the CRC unit to be turned off after the application starts. One concept that we would like to clarify is that random noise from the camera, like the subtle perturbation from lighting changes in the environment. By contrast, pictures generated by the graphics card do not have noise; even in games with fast-moving objects, majority of the macroblocks in successive frames are identical, and can benefit from our fast zero-motion detector.

Fig. 5 contains more detailed information on the CRC-augmented video encoder. Bold lines denote new hardware.

Using CRC instead of the conventional pixel-by-pixel comparison approach to detect changes in the frames has advantages in power consumption and encoding speed.

DRAM Power

Power savings primarily come from reduced frame buffer accesses. Pixel-by-pixel comparisons need to load pixel data of the reference frame from the system frame buffer, while the proposed CRC-based zero-motion detector only need to fetch CRC codes from SRAM. This is illustrated in Fig. 6. Assuming that an average $P\%$ of the whole frames do not

change, and that in each of the $1 - P\%$ of the frames with motion in them, $Q\%$ of the MBs actually changed, we can extrapolate a zero-motion percentage of $1 - (1 - P\%) \times Q\%$ on a macroblock basis. E.g., if $P = 60$, $Q = 30$, then 88% of the MBs would be zero-motion MBs. Frame buffer accesses to DRAM will thus be drastically reduced. During the intervals when there is no mouse movement or keyboard typing, DRAM ranks can be put to a low-power mode, resulting in significant DRAM power savings.

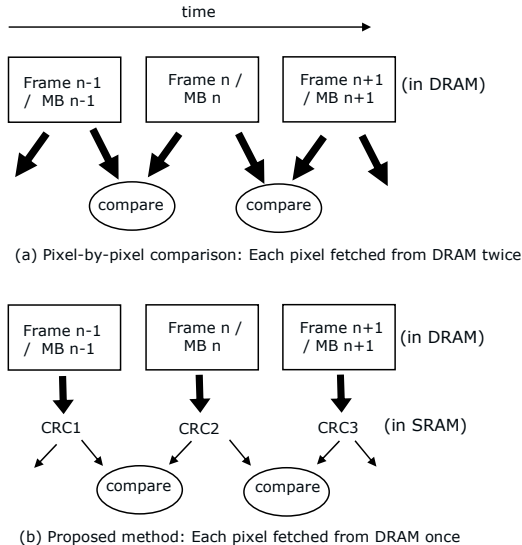


Fig. 6. Using CRC Code Obviates the Need to Fetch Reference Frame Pixels from DRAM

Encoding Speed

The CRC value is generated when pixels are written to the frame buffer. By contrast, traditional pixel-by-pixel comparisons add to the critical path of encoding, since pixel data of the reference frame would have to be fetched from the frame buffer first.

C. Periodically Overwriting Checksum Results for Failure Recovery

There is a slight possibility that motion between two blocks of data (a block being one frame or a 16×16 macroblock) be missed by short CRCs. In our experiments (explained in next section), we did not really observe occurrences of human-perceivable loss of image quality. The reason is that under normal scenarios, even if CRC fails to capture an image content change, changes between subsequent frames quickly re-establish correctness of the CRC results. For example, displaying one stale macroblock in one frame out of a 30FPS video stream is hardly perceivable to human eyes at all. However, for pathological combination of image contents or frame sequences, failure of CRC-based image change detection could result in repeated rendering of an obsolete image. To overcome this problem, we can periodically de-assert the CRC-based image change detection output, and thus force it into an “image changed” case. For

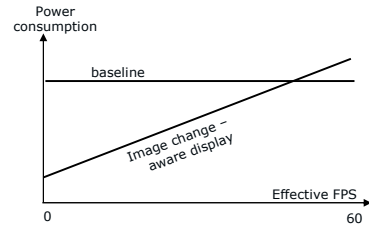


Fig. 7. Power Consumption of the Display System as a Function of the Effective Frames Per Second

example, the period for checksum cancellation could be 3 seconds after a “no image change” result is generated. By doing this, we guarantee that the display system can recover from an incorrect state.

III. PROTOTYPING AND EVALUATION

A. Prototype

We built a prototype using Altera FPGA and a 1280×800 AUO LCD display in order to prove our claim that CRC can capture image changes well. The system organization is similar to the one shown in Fig. 3. We used whole-frame CRC codes to check if two successive frames are identical. That is, one CRC value is generated per frame of 1280×800 resolution. Even if one single pixel changes between two successive frames, the CRC check will return “image changed”. If CRC check returns “no change”, the LCD controller switches to a low-power mode and stops fetching frame data from the frame buffer, shown in Fig. 5.

A demo can be downloaded from URL http://www.cs.utah.edu/~zfang/CRC_demo.wmv. In the prototype, we inserted our CRC-based Stream Image Change Detector (SICD) in the LCD-host interface on the test board on the left. If SICD detects that there is no image change, it signals the LCD controller to lower the panel refresh rate from 60Hz to 30Hz. The laptop screen on the right graphically shows the power consumption of the LCD panel. In this demo, at first SICD is turned off, so on the laptop screen we see a full power consumption. Then the user manually turns on SICD on the test board. When there is no image change on the LCD panel, LCD panel refresh drops to 30Hz, resulting in much lower power consumption. When the user starts to move the mouse, normal display refresh resumes. Utilizing an existing variable refresh rate technique[9], the prototype shows the effectiveness of CRC in detecting motion in images.

B. Quantitative Evaluation

Fig. 7 qualitatively illustrates the relationship between display system’s power consumption and Effective Frames Per Second (EFPS). Intuitively, EFPS represents the amount of frame changes in the image sequence. Next, we quantitatively evaluate the power savings of the proposed mechanism.

The total display system power can be expressed as

$$P_{DRAMdevice} + P_{DRAMbus} + P_{DispLocalFrameBuffer} + P_{MemoryController} + P_{LCDcontroller} + P_{others} \quad (1)$$

TABLE I
POWER CONSUMPTION OF KEY COMPONENTS IN A HYPOTHETICAL
BATTERY-POWERED DEVICE

Module Name	Power (mW)
System DRAM active + standby	350
System DRAM powerdown	105
System DRAM busses	leakage 40 + dynamic 40
Memory controller	leakage 192 + dynamic 48
LCD controller interface link	leakage 100 + dynamic 140
Display local frame buffer	80 (always full power)
CRC-based image change detector	20 (always full power)
FSB, IO busses, LCD backlight	always full power

in which most of the items consist of two components: active and idle power. Use η to denote percentage of time that a component is in idle state, we have

$$P = (1 - \eta) \times P_{active} + \eta \times P_{idle} \quad (2)$$

Assuming that aggressive power control techniques are employed, P_{idle} roughly equals leakage power for logics and wires, and “powerdown” power for DRAM devices¹. The difference between P_{active} and P_{idle} is mostly leakage/powerdown power. That is, $P_{active} = P_{leakage} + P_{dynamic}$.

For mostly-static display contents, our proposed mechanism can help clock gating utilities to save majority of the dynamic power, and help the memory controller to maximize DRAM devices’ residency in the powerdown mode. As a first-order approximation, total power savings of the proposed work in the display system is

$$\begin{aligned} P_{baseline} - P_{optimized} = & \\ & \eta \times (P_{DRAMpowerdown} + P_{DRAMbusDynamic} + \\ & P_{MemoryControllerDynamic} + P_{LCDcontrollerDynamic}) \\ & - P_{DispLocalFrameBuffer} - P_{CRCoverhead} \end{aligned} \quad (3)$$

where η is the average fraction of frame content that is not changing.

Table I contains power consumption numbers that we use as parameters in our power savings estimate. These parameters are primarily compiled from a number of publications including research papers [9], [16], [17], DRAM device vendors’ data sheets [18], [19], [20] and low-power system design guides [4]. In our estimate of power saving benefits, we do not consider power consumption reduction in FSB, IO interconnect and LCD panel backlight when the display is in idle modes. We will use the parameters in Table I to drive Equation(3), based on η that we measured for each application, presented next.

We extensively tested a large number of interactive and non-interactive applications. These include video creation,

¹Different DRAM device vendors use different terms and define different power states for the DRAM devices and peripheral circuitry. In the powerdown mode of this paper, we assume that all the IO buffers, sense amplifiers and row/column decoders in the DRAM device are deactivated but the internal PLL is on.

office productivity, E-learning, 3D modeling, Mobile Productivity, playback of an animated DVD movie, and active web browsing on cnn.com. Except for web browsing, these applications were sampled from standard benchmarks such as SysMark2007 [21] and MobileMark2007 [22].

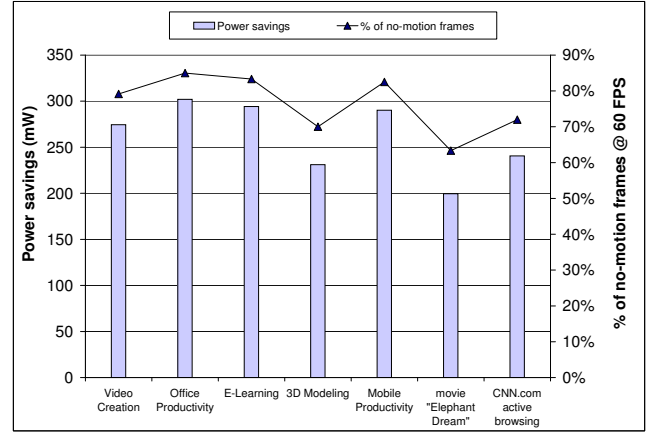


Fig. 8. Frame Change Characteristics and Estimated Power Savings

Fig. 8 shows the percentage of still frames for each benchmark that we measured. Based on the power consumption parameters in Table I, we projected power savings of using the frame change detector. The estimate only considered dynamic power savings, and assumed no leakage power reduction.

Fig. 9 shows numbers for the percentage of no-change image content measured at the frame granularity (first bar for each usage case) and projected at macroblock granularity (the other three bars for each usage case). The first three usage cases are standard benchmarks. The last one, “CNN.com browsing”, is content and network traffic dependent. When we transmit the MPEG frames at 60 FPS, for example, 27% of the frames for animated movie “Elephant Dream” have been measured to be identical to their previous ones, denoted by the first bar. If we assume that of those frames that have

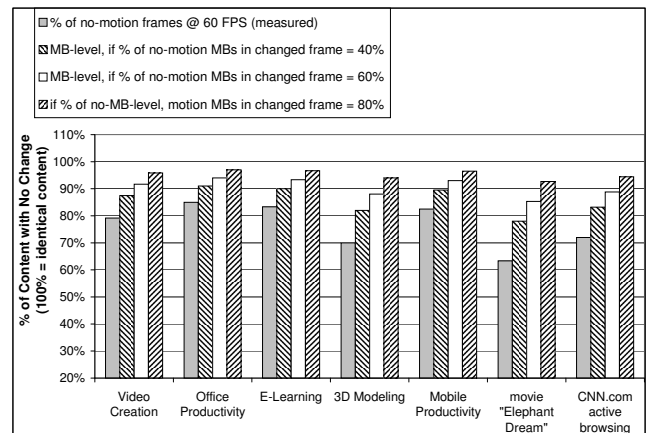


Fig. 9. Extrapolated Amount of Motion at Macroblock Level

any change in them (i.e., 73% of all frames), 80% of the 16x16 MBs do not change, then at the MB level 94% of all MBs are identical to their counterparts in the previous frames. The proposed method would be able to save 94% of frame buffer accesses when compressing data for the wireless remote display. Further instrumentation and direct measurement of power consumption is work in progress.

IV. SUMMARY

We propose and prototype a novel image change detection method using CRC code. Using CRC code as opposed to pixel-wise comparison can help to reduce significant percentage of frame buffer reads, therefore saving DRAM access power, for typical local display refresh and wireless remote display data compression. Our prototype verifies that short CRC codes can capture image changes rather well. As part of the future work, we plan to experiment with checksum/hash functions other than CRC, and to investigate other application of the proposed image change detector.

Acknowledgement

The FPGA prototype and data collection are part of a larger project in which we work together with other colleagues including Ajaya Durg, Quang Le, Jeremy Lee, Alfredo Alvarez and Wayne Proefrock. We thank Yu Dai, Guoqing Li, Yen-Kuang Chen and Yi-jen Chiu for their help on wireless remote display and video compression.

REFERENCES

[1] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," in *Proc. of ACM Multimedia*, 2003, pp. 582–591.

[2] H. Shim and N. Chang, "A compressed frame buffer to reduce display power consumption in mobile systems," in *Proc. of Asia and South Pacific Design Automation Conference*, 2004, pp. 819–824.

[3] P. Ranganathan, E. Geelhoed, M. Manahan, and K. Nicholas, "Energy-aware user interfaces and energy-adaptive displays," *IEEE Computer*, vol. 39, pp. 31–38, Mar. 2006.

[4] Microsoft. MSDN library, mobile and embedded development. [Online]. Available: <http://msdn.microsoft.com/en-us/library/aa936209.aspx>

[5] L. Brakmo, D. Wallach, and M. Viredaz, "uSleep: A technique for reducing energy consumption in handheld devices," in *Proc. of Int. Conf. Mobile Systems, Applications, and Services*, 2004, pp. 12–22.

[6] N. Chang, I. Choi, and H. Shim, "DLS: Dynamic backlight luminance scaling of liquid crystal display," *IEEE Trans. VLSI Syst.*, vol. 12, pp. 837–846, Aug. 2004.

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

[8] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan, "Energy-adaptive display system designs for future mobile environments," in *Proc. of Int. Conf. Mobile Systems, Applications, and Services*, 2003, pp. 245–258.

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

[10] RealVNC. The RFB protocol, Version 3.8. [Online]. Available: <http://www.realvnc.com/docs/rfbproto.pdf>

[11] Microsoft. MSDN library, remote desktop protocol. [Online]. Available: <http://msdn.microsoft.com/en-us/library/aa383015.aspx>

[12] Citrix. [Online]. Available: <http://www.citrix.com>

[13] ISO/IEC, "Information technology generic coding of moving pictures and associated audio information: Video."

[14] I. Richardson, *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, 2004.

[15] P. Peterson and D. Brown, "Cyclic codes for error detection," in *Proc. of the IRE*:49, Jan. 1961, pp. 228–235.

[16] J. B. Fryman, C. M. Huneycutt, H.-H. Lee, K. M. Mackenzie, and D. E. Schimmel, "Energy-efficient network memory for ubiquitous devices," *IEEE Micro*, vol. 23, pp. 60–70, Sept. 2003.

[17] H. Huang, K. Shin, C. Lefurgy, and T. Keller, "Improving energy efficiency by making DRAM less randomly accessed," in *International Symposium on Low Power Electronics and Design*, 2005, pp. 393–398.

[18] Micron. Technical note on mobile DRAM power-saving features and power calculations. [Online]. Available: <http://download.micron.com/pdf/technotes/tm4612.pdf>

[19] Elpida. Mobile RAM low-power modes. [Online]. Available: http://www.elpida.com/en/70nm/70nm_mobile.html

[20] Samsung. Technical note on low power mode for DDR registered DIMM. [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/dram/downloads/applicationnote/APP_061301.pdf

[21] BAPCO. SysMark 2007. [Online]. Available: <http://www.bapco.com/products/sysmark2007preview/>

[22] ———. MobileMark 2007. [Online]. Available: <http://www.bapco.com/products/mobilemark2007/>