A WATERMARKING CO-PROCESSOR FOR NEW GENERATION GRAPHICS PROCESSING UNITS

Saraju P. Mohanty
Computer Science and Engineering
Univ of North Texas, TX 76203.
Email: smohanty@cse.unt.edu

Nishikanta Pati
Computer Science and Engineering
Univ of North Texas, TX 76203.
Email: nishi@unt.edu

Elias Kougianos
Electrical Engineering Technology
Univ of North Texas, TX 76203.
Email: eliask@unt.edu

***Abstract*— Recent growth of high speed internet and high resolution imaging has enabled electronic storage and transfer of digital multimedia contents without resorting to the loss of quality. In order to protect the illegal reproduction of the digital multimedia elements, many researchers have suggested digital watermarking as a feasible solution. Like other signal and image processing works, digital watermarking is a computationally intensive process. For efficient, high performance, real time and low cost watermarking we propose two alternatives: (1) Using the Graphics Processing Unit (GPU) available on the modern graphics cards for the complex mathematical computations or (2) Implementing a dedicated processor chip, a coprocessor for the GPU, to accomplish the task. In this paper we present the later alternative, a coprocessor for the GPU to do multimedia watermarking for real-time applications.**

I. Introduction and Related Research

The watermarking technology makes it possible to identify the creator, distributor or authorized consumer of a multimedia element [1]. In order to be effective, a watermark should be perceptually invisible and the data owner or an independent control authority should be able to extract it easily for authentication. Moreover, it must be difficult for an attacker to remove or destroy the watermark embedded inside. Many applications like video broadcasting require the copyrighting process to run in real-time at video frame rates [2].

Many significant contributions [3] [5], [6] have been made to the field of digital watermarking. Most of the robust watermarking methods often need computational intensive operations. This constraint puts the watermarking processing to be done off-line. However, real-time insertion of the watermark as and when data is recorded is very useful for applications like video broadcasting, traffic monitoring, etc. Now-a-days, most of the graphics cards have a powerful processor chip, referred to as Graphics Processing Unit (GPU), to accelerate the graphics processing. Recent performance improvement of the graphics cards have gained lot of interest among the reserach communities to harness the tremendous power GPU for general purpose computing [4], [7]. Fung et al. [8] have proposed to use graphics cards for efficient computation of computer vision algorithms leaving CPU free for other tasks.

Inspired by the above discussed challenges and ventures, we propose a dedicated a co-processor chip targeted for real-time watermarking. The chip will be integrated as a co-processor to the existing GPU. We have customized a renowned robust invisible image watermarking algorithm to facilitate building of the real-time architecture of the watermarking processor followed by hardware implementation. The proposed architecture is designed aiming at an easy integration as an module into any existing GPUs.

II. The Proposed New Generation GPU

Fig. 1 depicts the proposed schematic architecture for new generation GPU with watermarking processor. The graphics pipeline as shown in Fig. 1 are fundamental blocks of all the modern GPUs as described by Owens *et al.* [4].

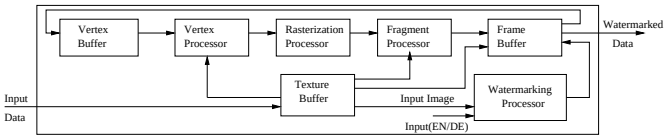

Fig. 1. The Proposed GPU Architecture with Watermarking Co-Processor

We propose to add a watermarking processor to the existing hardware organization to enable the GPU to watermark multimedia elements, like images, efficiently. For normal graphics applications, the pipeline takes a list of geometry, expressed in terms of vertices, as input. The functional blocks perform the necessary processing and render the final image into the frame buffer. However, for real-time watermarking the watermarking processor will be activated. The watermarking processor will take the data from global memory inside texture unit. Based on the input whether to embed or extract (EN / DE) a watermark, the watermarking processor executes the required instructions. The output from the watermarking processor gets saved in the frame buffer. While the output of the watermarking processor is a watermarked image if a watermark is embedded into the host image, a binary signal describing the presence or absence of a watermark is sent out during extraction.

III. The Watermarking Algorithm

We present an invisible robust watermarking algorithm amicable for hardware implementation based on the methodology proposed by Piva *et al.* [5] that does not need original image for extraction and authentication (*blind*). The algorithm enables us to design an efficient VLSI Architecture without losing the robustness of the blind invisible watermarking.

The steps of invisible *insertion* is described below:

1) Find the 8×8 block-wise DCT of the host image.

2) Locate the blocks which are in the center quarter of the image, where each block represents a 8×8 pixel area.
3) Choose four coefficients, the $F_{3,2}, F_{2,3}, F_{4,1}, F_{1,4}$ coefficients, from the qualifying blocks and generate a vector $V = \{v_{1,i}, v_{2,i}, v_{3,i}, v_{4,i}, ..., v_{1,M}, v_{2,M}, v_{3,M}, v_{4,M}\}$, where M is the number of 8×8 pixel blocks in the central quarter of the image and $v_{k,i}$ represents the selected coefficients of the DCT matrix of block i.
4) Select a watermark $W = \{w_1, w_2, ..., w_{4\times M}\}$, which is a pseudo random sequence of $4 \times M$ real numbers, and each value w_i is a random number having a standard normal distribution with zero mean and unit variance.
5) The watermark W is embedded into the vector V to obtain a new vector $V^{'} = \{v_1^{'}, v_2^{'}, ..., v_{4\times M}^{'}\}$, according to the following rule, where $i = 1, 2, .., 4 \times M$:

$$v_i^{'} = v_i + \alpha |v_i| w_i. \quad (1)$$

6) The vector $V^{'}$ is then reinserted to the block-wise DCT matrix at chosen coefficients of corresponding blocks.
7) The inverse DCT algorithm is performed, obtaining the watermarked image $I^{'}$.

During the *blind extraction* watermark, given possibly corrupted image I^*, the 8×8 DCT is applied to I^*. The blocks which lie in the center quarter of the image are located. The selected coefficients chosen during watermarking process from the central quarter blocks are taken to generate a vector $V^* = v_{1,i}^*, v_{2,i}^*, ..., v_{3,M}^*, v_{4,M}^*$. The correlation between the extracted coefficients V^*, and the watermark itself is taken as a measure of the watermark presence. By comparing the value of correlation γ to a predefined threshold T_γ, it is possible determine whether a given watermark is present or not. The correlation is defined as:

$$\gamma = \frac{W.V^*}{M} = \frac{\Sigma w_i v_{L+i}^*}{M}, \quad (2)$$

and the threshold value determined as:

$$T_\gamma = \frac{\alpha}{3M\Sigma_{i=1}^{M} v_i^*}. \quad (3)$$

IV. Architecture of the Proposed Chip

The architecture for the proposed chip is shown in Fig. 2. The watermarking co-processor that can perform blind invisible watermarking contains the encoder and decoder units at the highest level of abstraction. Based on the input EN/DE the processor will either create a watermarked image or extract the watermark and verify whether it contains the valid watermark or not. The functional modules used during watermarking process are shown in Fig. 3. The DCT module inside the encoder (watermark embedder) computes the block-wise DCT. The coefficient selection unit selects the coefficients and sends them to the embedding unit for further processing. The pseudo random number generator unit generates a sequence of random numbers of desired length of a given seed to serve as a watermark. The controller controls the operation of the functional units. The decoder (extractor) module of the co-processor has similar modules with additional modules for extraction and authentication.

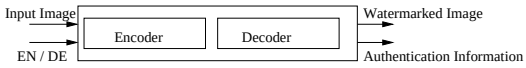

Fig. 2. The Co-Processor for Blind Invisible Robust Watermarking

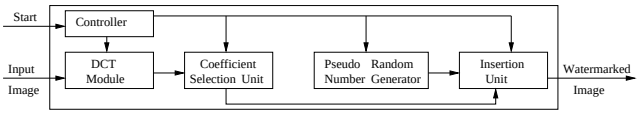

Fig. 3. Block Diagram of the Encoder

V. Experiment And Simulation Results

The proposed co-processor has been simulated in MATLAB and tested with several images, sample image Lena is shown in Fig. 4. Further, several common signal processing techniques and geometric distortions were applied to this image to evaluate the algorithm. The response of the watermark detector for random number sequences (watermarks) with 1000 different seeds is shown in Fig. 4(c). This reveals the functionality and robustness of the proposed system.

(a) Original Lena (b) Watermarked Lena (c) Detector's Response

Fig. 4. Experimental Results of Proposed Watermarking Co-Processor

VI. Conclusion

The proposed architecture is a novel idea in its domain. Though the results are shown only for images, it can be easily extended for other types of multimedia. The development of a parallel, pipelined, low power architecture is under progress and will appear in a longer version of the paper.

References

[1] S. P. Mohanty, Digital Watermarking of Images, Master's thesis, Dept. of Electrical Engineering, Indian Institute of Science, Bangalore, India, 1999.

[2] S. P. Mohanty, R. Kumara C., and S. Nayak, "FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder", in *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, Vol. 3356, pp. 344-353, 2004.

[3] S. P. Mohanty, N. Ranganathan, and K. Balakrishnan, "A Dual Voltage-Frequency VLSI Chip for Image Watermarking in DCT Domain", *IEEE Transactions on Circuits and Systems II*, 53(5), May 2006, pp. 394-398.

[4] J. D. Owens, et. al, "A Survey of General-Purpose Computation on Graphics Hardware", *Eurographics 2005*, State of the Art Reports, August 2005, pp. 21-51.

[5] A. Piva, et. al, "DCT-Based watermark recovering without restoring to the uncorrupted original image", in *Proc. of IEEE ICIP*, 1997.

[6] I. J. Cox, et. al., "Secure Spread Spectrum Watermarking for Images, Audio and Video", *Proc. IEEE ICIP*, vol. III, pp. 243-246, 1996, .

[7] J. Krger and R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms", *ACM Transactions on Graphics*, Vol. 22 No. 3, July 2003, pp. 908-916.

[8] J. Fung and S. Mann, "Computer Vision Signal Processing on Graphics Processing Units", in *Proc. of the IEEE ICASSP*, pp. V-93 - V-96, 2004.