HARDWARE IMPLEMENTATION OF A FINGERPRINTING ALGORITHM SUITED FOR DIGITAL CINEMA

G. Rouvroy[1,2], *F. Lefebvre*[3], *F.-X. Standaert*[1,2], *B. Macq*[3], *J.-J. Quisquater*[1,2], *J.-D. Legat*[2]

[1]UCL Crypto Group
[2]Laboratoire de Microélectronique
[3]Laboratoire de Télécommunications et Télédétection
Université Catholique de Louvain

rouvroy,standaert,quisquater,legat@dice.ucl.ac.be
lefebvre,macq@tele.ucl.ac.be

ABSTRACT

This paper presents the performance evaluation of a watermarking algorithm, an image and video processing technique used for Digital Right Management. Software developments are largely used in image copyright applications but there is a lack of real time applications for video. A previous work in image watermarking was already done and proposed a new algorithm with very interesting security properties. Nevertheless, the software implementation of this watermarking scheme showed some restrictions concerning real time applications. Hence, we propose to extend the previous paper with a lighter version of the algorithm and a complete hardware implementation allowing us to deal with high throughput image/video applications, such as digital cinema.

1. INTRODUCTION

The Digital Right Management issue constitutes a bottleneck for a large use of digital contents. The aim of this work is to propose a solution to protect video displaying for digital cinema. Some works have been done in image and audio security and a largely used method is the watermarking. According to [1], watermarking techniques depend on the working domain, type of document, human perception and applications.

The processing involved in video security for cinema is highly constrained. In such an application, the manipulated pictures have a size of 2048 × 1024 pixels or more. Hence, the algorithm processes a high dataflow of information. A high throughput is one of the main priorities. A real time process is also expected to personalize each movie in each projection room and ensure a tracking of the media after projection. Therefore, the watermark needs to be different in each projection room: it is called fingerprinting or labelling. This method is used to distinguish an object from other similar objects and helps tracing the stolen video sequence and the corrupted projection room.

The visual quality of pictures projected on the screen of a projection room should not be affected by the deterioration brought by the mark pattern addition. As described in [2], the requirements on watermarking scheme are invisibility, security, robustness, complexity, constant bit-rate, interoperability and retrieval.

A strong but slow algorithm detailed in [3] offers some interesting properties such as resistance against compression and Digital/Numeric conversion. This algorithm is based on secret key management where the embedding and extraction keys are the same and must be kept secret[1]. It works in uncompressed domain, so it does not depend on the compressed domain of the image or video sequence.

Hence, a fingerprinting platform should be based on a quick mark embedding algorithm and a robust and invisible spread signal added to the original multimedia stream. For very high multimedia bitstreams, the fingerprinting process need to be hardware implemented to meet the application constraints. We investigate a solution based on a reconfigurable device (a Virtex-II FPGA[2] from Xilinx). In addition, to increase the speed efficiency, we modify the original algorithm [3] in order to simplify the embedding process without deteriorating the robustness. FPGA fingerprinting designs propose a judicious tradeoff between speed, visual quality and robustness against different kind of attacks.

The description of the fingerprinting algorithm is described in the second section. To ensure a real time application, the FPGA implementation is detailed in the third section. Finally, the fourth section concludes this paper.

2. WATERMARKING ALGORITHM

2.1 Previous work

The watermarking presented in [3] is based on a secret key management. It allows embedding a 64-bit mark into one image. It is also a blind architecture meaning that we do not need original images to extract the watermark. This algorithm ensures a strong resistance against some attacks such as print and scan, compression, noise, cropping, translation and rotation. Tracking and copyright protection are also very interesting applications for this fingerprinting scheme. It was used successfully in some European projects [5] and [4].

The algorithm is divided in three parts: pattern generation which is the pseudo mark embedded, the psychovisual mask which is the mark weighting for the invisibility, and the synchronized block which is a template added to pattern generation to be resistant against geometrical deformation.

Even if the algorithm is very efficient for still images, video tests applied in projection room show some deficien-

This work has been funded by the Wallon region (Belgium) through the research project TACTILS *http : //www.dice.ucl.ac.be/crypto/TACTILS/T_home.html*

[1] As opposed to an asymmetric public watermarking system.

[2] Field Programmable Gate Array.

cies. The watermark embedded in the projected pictures is not always detected. The problem is due to the inefficiency of the synchronization block. In addition, this block is too slow and too complex for real-time digital cinema applications.

2.2 Description of the light algorithm

To avoid these previously detailed problems concerning digital cinema, we propose an evolution to a lighter algorithm as detailed in Figure 1. We replace the synchronization block by an off-line process, called RASH [7]. One of the RASH functionalities allows an efficient recovering of geometrical manipulations. Therefore, we get a fast and robust fingerprinting scheme.

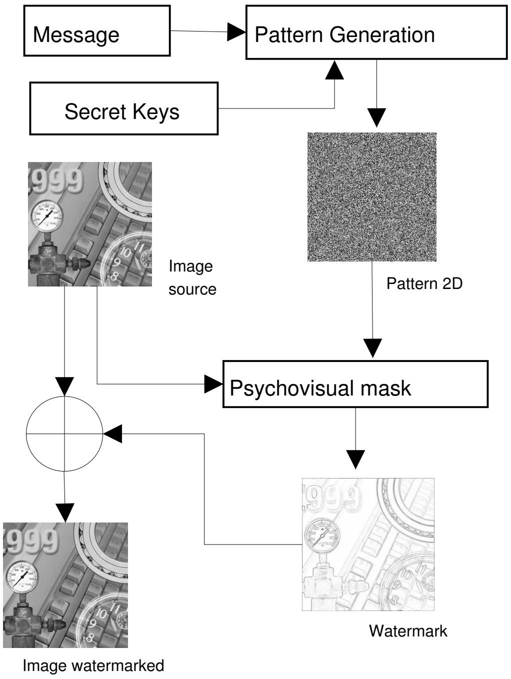

Figure 1: Watermarking block scheme.

The different blocks of our light watermark are shown below. More detailed information can be found in [3].

- **Psychovisual mask**: The psycho visual mask is based on image activity (local mean) that compares medium pixel intensity inside its environment (its neighbors), and the Weber-Fechner law that highlights the visibility of the pixel intensity.
- **Convolutional code**: We chose a convolutional code to encode the original message. Therefore, we extend the 64-bit original message to a 128-bit code. Soft Viterbi is used to recover the original message. Convolutional code offers real improvement for the watermarking extraction due to the redundancy of the 128-bit code. Indeed, the original message could be correctly revealed even if some errors appear in the extracted 128-bit code.
- **Pseudo-random generator**: An efficient watermark is a robust mark based on redundancy, an accurate recovery method and an undetectable mark for a user without right. A MLS[3] pseudo-random sequence provides most of the previous requirements. The length of this cyclic sequence is $n = 2^m - 1$, where m[4] is the number of stages. This code generates a Gaussian noise appearance and provides interesting detection properties. For secure extraction, we define a 40-bit key Key_0. This key is used as the secret seed for the generation of our MLS code. The combination of the convolutional code and the MLS code allows generating an 1-D cyclic sequence called $WORD$. Typically, the length of $WORD$ is 2^{15}.
- **2-D Pattern**: We create a 2-D cyclic pattern, expanding our $WORD$ into a matrix. The processed method is a linear computation between the image point coordinates and two 8-bit secret keys Key_1 and Key_2:

$$Pattern(i,j) = WORD[(i*Key_1 + j*Key_2) \\ mod\ \ length(WORD)] \qquad (1)$$

where (i, j) represents the image pixel coordinates. $Pattern(i, j)$ represents the way we are going to modify the (i, j) pixel intensity: if it is equal to 1 (0), the pixel intensity will be increased (decreased).

A translation or cropping operation on the captured image is equivalent to the same transformation on the 2-D pattern. It simply corresponds to a cyclic permutation of the $WORD$, that fully permits a robust extraction of the original mark:

$$Pattern(i+i_0, j+j_0) = WORD[((i+i_0)*Key_1 + \\ (j+j_0)*Key_2)\ \ mod\ \ length(WORD)]$$

$$Pattern(i+i_0, j+j_0) = WORD[((i*Key_1 + j*Key_2) + \\ (i_0*Key_1 + j_0*Key_2))\ \ mod\ \ length(WORD)]$$

This watermarking algorithm is almost fully image processing resistant and cropping/translation resistant.

2.3 Watermarking detection performance

To evaluate the robustness and performance of our watermarking method, we experiment on 40 real-world images taken from the USC-SIPI database [6].

In [1], an embedding process is defined as:

$$v'_i = v_i + \alpha b_i p_i$$

Where v'_i is the fingerprinted signal, v_i is the original signal, b_i the embedded message, p_i pseudo noise sequence, α is the force of the mark. In our case $\alpha.b_i.p_i$ are weighted by the psychovisual mask.

For each of the 40 images of the data base, we embed a message with a range of six α (0.02,0.04,0.08,0.1,0.15,0.2). To evaluate the image processing degradation due to fingerprinting insertion, we calculate the PSNR mean for each modified images according to the force α. Figure 2 shows the resulting PSNRs. An empirical value of $32db$ is a good PSNR threshold to achieve a not too visible added template.

For each fingerprinted image, we consider 4 image processing attacks, generating $40*6*4 = 960$ images, named processed images. The attacks are filtering (3x3 Gaussian filtering with standard deviation of 0.5), compression (JPEG

[3] Maximum Length Shift register or m-sequence.

[4] 40 in our case.

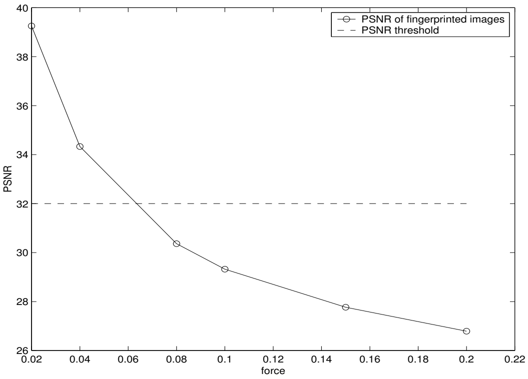

Figure 2: PSNR mean of 40 fingerprinted images regarding the force of the fingerprint

compression with 80% and 60% quality factor), and noise (salt and pepper).

The robustness results are given by Tables 1, 2, 3, 4. The term *Extracted* represents the number of processed images where the mark is correctly detected and extracted, *Only detected* represents the number of processed images where the the mark is correctly detected but too many bits are lost to compute a correct extraction process and *No detected* represents the number of processed images where the mark is not detected and no extracted.

Force	0.02	0.04	0.08	0.1	0.15	0.2
Extracted	29	39	40	40	40	40
Only detected	4	1	0	0	0	0
No detected	7	0	0	0	0	0

Table 1: Gaussian attack.

Force	0.02	0.04	0.08	0.1	0.15	0.2
Extracted	27	40	40	40	40	40
Only detected	4	0	0	0	0	0
No detected	9	0	0	0	0	0

Table 2: Noise attack.

Force	0.02	0.04	0.08	0.1	0.15	0.2
Extracted	26	37	40	40	40	40
Only detected	3	3	0	0	0	0
No detected	11	0	0	0	0	0

Table 3: Jpeg attack, quality=60.

Force	0.02	0.04	0.08	0.1	0.15	0.2
Extracted	30	39	40	40	40	40
Only detected	3	1	0	0	0	0
No detected	7	0	0	0	0	0

Table 4: Jpeg attack, quality=80.

Attacks and PSNR figures provide a good empirical value of the force, closed to 0.06, to obtain a good tradeoff robustness/visibility of the fingerprint.

3. HARDWARE ASPECT

3.1 Detailed block implementation

As previously mentioned, the fingerprinting insertion process needs to be hardware implemented to deal with the high bit rate of digital cinema. Figure 3 illustrates the global FPGA architecture of our fingerprinting scheme. We propose a complete unrolled and pipelined design to ensure the data processing throughput of digital cinema. We adapt the design to support 2048 × 1024 frames with a dataflow of 24 images per second.

The first watermarking step is to compute the convolutional code from the 64-bit original mark and the MLS sequence until the *WORD* sequence is completely generated. $WORD(n)$ means the n^{th} bit of the sequence. The proposed design allows us to change Key_0 and the mark to embed for each new frame. About 100,000 clock cycles ($\simeq$ 1 ms) are required to generate a new *WORD* from a new key or a new original message. Therefore, it is not a judicious choice to change these inputs for every new frame.

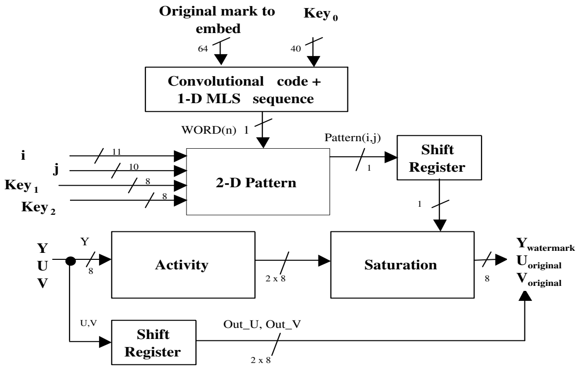

Figure 3: The global fingerprinting architecture.

Once the WORD is generated, we start to compute the 2-D pattern assuming that the image pixels are received in a one by one serial way (cycle by cycle) in the YUV domain, line by line. We first receive $(0,0)$, then $(1,0)$, $(2,0)$, ... , $(0,1)$, $(1,1)$ and so on. Every cycle a new pixel (i,j) is processed and a new $Pattern(i,j)$ is computed. Secret keys Key_1 and Key_2 can be modified for every new frame without any dead cycles, which it is not the case for Key_0. Nevertheless, changing only Key_1 and Key_2 regularly is not secured enough. It is better to change sometimes all the secret keys between two frames. Figure 4 shows the architecture concerning the calculation of Equation 1. The 32.768 bits of the *WORD* sequence are stored inside two separate RAM blocks[5].

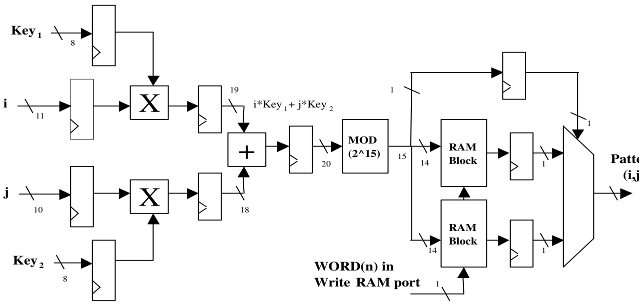

Figure 4: The 2-D pattern block.

[5] Virtex-II FPGAs have only internal 18-Kbit RAM blocks.

In parallel with the 2-D pattern calculation for the (i,j) pixel, we compute the first part of the psychovisual mask based on the local activity of the Y component. The activity of (i,j) pixel is the difference between the intensity of the (i,j) pixel and the mean intensity of the close pixels[6].

Figure 5 illustrates the activity calculation of one pixel. In addition, the block computes the absolute value of the activity value and returns Out_Y.

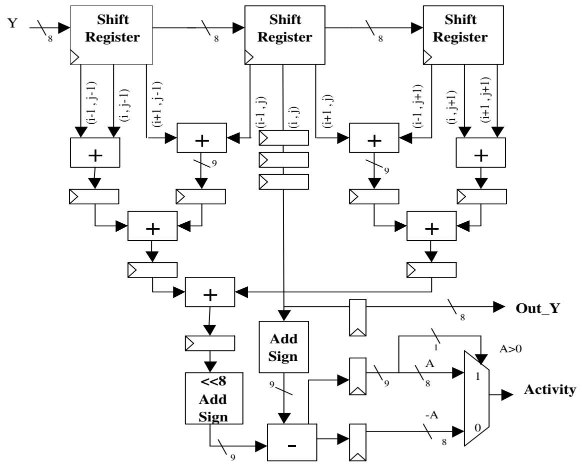

Figure 5: The Activity block.

Figure 6 completes the psychovisual mask applying the Weber-Fechner law (stored in a ROM). In addition, the saturation block inserts the mark thanks to the $Pattern(i,j)$ bit and ensures that the modified intensity is between 0 and 255.

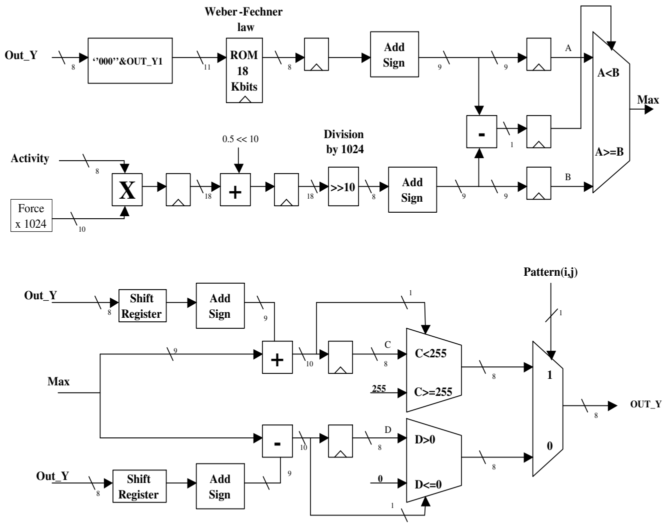

Figure 6: The Saturation block.

3.2 Complete implementation results

The synthesis of our complete fingerprinting design was done using Synplify Pro 7.2 from Synplicity. The placing and routing were done using Xilinx ISE 6.1.i. The final results are given in Table 5 for a Xilinx Virtex-II FPGA (XC2V500-6). We detail the resources used according to two frame sizes (2048 × 1024 and 1024 × 768).

Frame size	1024 × 768	2048 × 1024
LUTs used	1670	2727
Registers used	759	761
Slices used	**1065**	**1617**
RAM blocks used	4	4
Multipliers used	4	4
Max. Output every (cycles)	1	1
Frequency (MHz)	143.9	143.9
Max. Throughput (Mbps)	**3454**	**3454**
Nbr Images/seconde	**182.98**	**68.62**

Table 5: Final results of our complete fingerprinting scheme.

Our design is able to fingerprint all 2048 × 1024 video frames even if we need to project at a dataflow of 48 images per seconde. We fully meet the digital cinema requirements.

4. CONCLUSION

This paper presents a light fingerprinting algorithm perfectly adapted to digital cinema and tracking of media after its projection/diffusion. We evaluate the performance of the complete watermarking scheme and we tune the algorithm parameters to reach a good tradeoff robustness/visibility. To avoid the slowness of the software implementations, we propose a complete FPGA implementation of the fingerprinting insertion. The resulting design is able to deal with 2048 × 1024 video frames at a throughput of about 68 images/sec. This solution completely meet the digital cinema requirements for a very reasonable hardware cost.

REFERENCES

[1] Saraju P. Mohanty, ”Digital watermarking: a tutorial review”, 1999, http://www.csee.usf.edu/ smohanty/research/Reports/WMSurvey1999Mohanty.pdf.

[2] F.Hartung and B. Girod, ”Copyright Protection in Video Delivery Networks by Watermarking of precompressed Video”, *ECMAST*, 1997, p423-436, Tokyo, Japan.

[3] F. Lefebvre, D. Gueluy, D. Delannay and B. Macq,”A print and scan optimized watermarking scheme”, *MMSP* 2001, p511-516, Cannes, France.

[4] ”ASPIS: An Authentication and Protection Innovative Software System for DVD and Internet”, European project, IST-1999-12554.

[5] ”CERTIMARK: Certification For Watermarking Techniques”, European project, IST-1999-10987.

[6] ”USC-SIPI image database”, available at http://sipi.usc.edu/services/database/Database.html.

[7] F. Lefebvre, B. Macq and J.-D. Legat,”RASH:RAdon Soft Hash algorithm”, European Signal Processing Conference, 2002, Toulouse.

[8] Xilinx: ”Virtex-II Field Programmable Gate Array Data Sheet”, http://www.xilinx.com.

[6] In total, there are 8 pixels involved for the mean calculation, the direct 8 neighbors of the (i,j) pixel.