FPGA IMPLEMENTATION OF IMAGE WATERMARKING USING XILINX SYSTEM GENERATOR

Garima Gupta¹, Vijay Kumar Gupta², Mahesh Chandra³

¹, ²Department of ECE, Indraprastha, Engineering College Ghaziabad
³School of ECE, REVA University Bengaluru

ABSTRACT

In this paper, co-hardware simulation of Image Watermarking has been performed using Xilinx System Generator and VIRTEX 6 target device on ML605 FPGA Board. In Watermarking, valuable information can be hidden in the set of digital media by modifying the graphic contents. Here, Watermarking is performed in DWT domain after scaling of watermark data. Watermarking is implemented in MATLAB Simulink with Xilinx System Generator toolbox. Then, Co-hardware simulation is performed using VIRTEX 6 device on ML605 Board. The visual results of both simulation and Co-hardware simulation models are achieved and it is observed that the Co-hardware simulation results are better. The device utilization during the implementation of Co-hardware simulation of watermarking process on ML605 Board is compared with the implementation on LX240 Board. The ML605 implementation shows a requirement of smaller number of components as compared to other researchers work.

Keywords: Image Watermarking, DWT, Scaling, Xilinx System Generator(XSG), Co-hardware Simulation

I. INTRODUCTION

In this paper, hardware co-simulation of image watermarking is performed using MATLAB and Xilinx System Generator on ML605 board with Virtex 6 as the target device. Hardware setup is implemented using either DSP’s or FPGA’s. Since design flexibility with FPGA is more than DSP, it is more feasible to implement such system with FPGA for optimized performance as discussed[1] by Ana Toledo Moreo et al.

Roy, Li and Shoshan[2] have done hardware implementation of the Digital Watermarking system for compressed video authentication. The authors worked on the real-time implementation of invisible watermarking using FPGA. Karthigaikumar, Anumol and Baskaran[3] as well as Korrapati, Nelakudit and Mandhala[4] worked on watermarking using Simulink block in MATLAB and then the algorithm is converted into Hardware Description Language (HDL) using Xilinx System Generator tool. The algorithm is prototyped in Virtex-6 (vsx315tff1156-2) FPGA. Joshi, Mishra and Patrikar [5] worked on real time implementation of Digital Watermarking for Image and Video. They made hardware implementation of digital watermarking using DCT domain. Bhaisare et al. [6] implemented real time method for watermarking system. They embed invisible, semi fragile watermark information into compressed video streams using DCT. Also, Hajjaji, Mohamed Ali, et al. [7] worked on Xilinx System Generator and they used HAAR DWT for watermarking. This model was implemented on ML507 Evaluation Platform which is based on the Virtex-5 FPGA using Xilinx System Generator tool. Saidaniet al. [8] worked on the co-hardware simulation for video watermarking. The design was implemented using a Spartan3 device (3S200PQ208) then a Virtex-II Pro (xc2vp7- 6ff672). Shivdeep, Ghosh and Rahaman [9] performed FPGA and ASIC implementation of color image watermarking. The use of pseudo noise code enhances the security of watermarking scheme. Xilinx Spartan 3E FPGA kit with XC3S500E device is used for performing the task. Xilinx ISE 14.7 project navigator along with XST synthesis tool and ISim simulator are used for interfacing, RTL synthesis and simulation, respectively. Pexaras, Karybali and Kalligero [10] presented image and video watermarking schemes for low-cost applications. They presented FPGA implementation of robust invisible watermarking using spatial domain. In this method, area calculated for blocks consumption is small as compared to existing methods as shown by the authors.
With the development of technology, efficient and rapid prototyping systems have emerged in current scenario. These systems require a development environment targeting the hardware design platform. Creating specialized hardware would greatly reduce the time consumed by these processes. Simulink in MATLAB is an environment for modeling and simulation. This paper presents a work where watermarking is first modeled and simulated on MATLAB-Simulink environment on XSG tool and afterword co-simulation is done on ML605 board as shown in Fig. 1. System Generator is a design tool based on MATLAB-Simulink. Xilinx is a family member of the System Generator tool. The tool provides high-level generalizations that are automatically amassed into an FPGA at the push of a button. The Xilinx Integrated Software Environment (ISE) is a powerful design environment that is working in the background while implementing System Generator blocks.

When XSG is configured with MATLAB, it activates Xilinx block set library to appear in MATLAB-Simulink environment. With the help of Simulink and Xilinx block set, model is designed and successfully run; a net list is automatically generated for the given model. Then schematic representation of our synthesized source file can be displayed with the help of RTL view. This schematic shows a representation of the pre-optimized design in terms of standard symbols such as multipliers, counters, adders, AND gates, and OR gates that are independent of the targeted Xilinx device. This schematic also calculates the actual resource utilization for the designed model. Then mapping of synthesize design to physical resource of the target device is done to create implemented design. In this paper, hardware implementation of image watermarking is presented on ML605 Board.

II. PRELIMINARY

2.1 Low Pass Filter and High Pass Filter

For 2-D DWT implementation, the image is first converted into four sub bands- LL, LH, HL and HH. This frequency wise decomposition separates the image into high frequency and low frequency components. Frequency components are separated by using high pass and low pass filters. Fig. 2 shows the 2-D decomposition of input image. Here, \( g_0[n] \) is low pass filter and \( h_0[n] \) is high pass filter representation. After down sampling four components (LL, LH, HL and HH) are obtained as shown in Fig. 2. Xilinx FIR Compiler 5.0 of XSG has been used to design LPF and HPF which are further used to convert the image into four sub bands LL, LH, HL, HH as shown in Fig 2. The FDA Toolbox is used to define the filter order and coefficients for FIR compiler as shown in Fig 3. Having proper parameter settings of FDA Toolbox and Xilinx FIR Compiler 5.0 block, filter coefficients are automatically added from FDA Toolbox in both LPF and HPF filters designed with FIR Compiler as shown in Fig. 4. Coefficients for LPF and HPF are shown in Fig. 5. This figure shows four tables of filter coefficients- two for embedding and two for extraction. The output of the filters is displayed using spectrum scope.
III. METHODOLOGY

3.1 Watermark Embedding

The DWT based image watermark embedding method is used in experiment. Both input and watermark images are either black and white image or color image. Fig. 6 shows the block diagram for watermark embedding and Fig. 7 shows the XSG implementation for watermark embedding. After this model is simulated in MATLAB-Simulink environment. Net list is generated and then configured for hardware co-simulation. Steps for watermark embedding are as follows-

Step 1: Both host and watermark images are read in MATLAB-Simulink environment and pre-processing is applied on them. Here Pepper image is considered as host and Leena as the watermark image. Host, watermark images are color image, gray scale image respectively.

Step 2: Further pre-processing like red component selection from host image, resizing, 2D to 1D conversion, frame conversion and buffering are done before passing the images to the XSG model as input.

Step 3: InXSG model, 2-D DWT of both host and watermark image is first calculated. Four components after DWT of host image are \(LLh, LHh, HLh, HHh\). Four components after DWT of watermark image are \(LLw, LHw, HLw, HHw\).

Step 4: Each component of watermark image \((LLw, LHw, HLw, HHw)\) is scaled separately with a scaling factor before embedding with host image.

Step 5: Scaled components with a scaling factor \(\alpha\) of watermark image are embedded with components of the host image according to the Eqn.1, Eqn.2, Eqn. 3 and Eqn.4 as given below:

\[
LL = LLh + \alpha \cdot LLw \tag{1}
\]

\[
LH = LHh + \alpha \cdot LHw \tag{2}
\]
\[
\begin{align*}
HL &= H_{Lh} + \alpha \ast H_{Lw} \\ 
HH &= H_{Hh} + \alpha \ast H_{Hw}
\end{align*}
\] (3) (4)

**Step 6:** By applying inverse DWT to the new values of \(LL, LH, HL\) and \(HH\) the watermarked image is formed.

![Table of Filter Coefficients](a) (b)

![Table of Filter Coefficients](c) (d)

**Figure 5:** Filter coefficients

- (a) LPF Coefficients for Embedding
- (b) LPF Coefficients for Extraction
- (c) HPF Coefficients for Embedding
- (d) HPF Coefficients for Extraction

![Diagram of Watermark Embedding](a)

**Figure 6:** Block Diagram for Watermark Embedding with Host (Pepper Image) and Watermark (Leena Image)

Fig. 8 shows Co-hardware simulation of embedding system. Co-hardware simulation of embedding system is generated, and inputs-outputs are connected to the generated model.
3.2 Watermark Extraction

Watermark extraction gives back the embedded watermark. Hence the embedding steps in the reverse direction need to apply to get back embedded watermark.

**Step 1:** Watermarked image is taken as an input of the extraction system. Pre-processing operations like resizing, 2D to 1D conversion are applied at the time of embedding.

**Step 2:** DWT is applied on the watermarked image to get $LL$, $LH$, $HL$ and $HH$ sub-bands.

**Step 3:** As, $\text{Watermarked image} = \text{host image} + \alpha \ast \text{watermark image}$

Therefore, $\text{Watermark image} = \frac{(\text{Watermarked image} - \text{host image})}{\alpha}$

Where $\alpha$=scaling factor. The value of $\alpha$ can be chosen randomly, but it is kept same both at the time of embedding and extraction.

**Step 4:** The extracted watermark is compared with the original watermark for its similarity. More similar will be the extracted watermark to the original watermark, better will be the results.

Fig. 9 shows block diagram of watermark extraction. Input watermarked image is called from MATLAB workspace. Host image is Pepper image that is connected as another input of extraction system. After applying DWT to both the watermark and host image, host image sub-bands and watermarked image sub-bands are obtained. Host sub-bands are subtracted from watermarked sub-bands and then inverse scaling is applied. At the output, four sub-bands of watermark are obtained and when IDWT is applied to these sub-bands, watermark image is obtained. Fig. 10 shows the implementation of extraction system using XSG. Fig. 11 shows Co-hardware simulation model of extraction system.
IV. RESULTS AND DISCUSSION

Models for watermark embedding and extraction shown in Fig. 7 and Fig. 10 are implemented. In this section, the results of these models for only simulation and hardware co-simulations are presented, compared, and discussed. Higher similarity between watermarked image and host image is observed. In this implementation, host image and watermark image are Pepper image and Leena image, respectively.
4.1 Hardware Co-Simulation results

Embedding and extraction system is simulated through XSG and then interfacing of XSG with ML605 kit provides results after co-simulation.

4.1.1 Watermark Embedding

Results for Simulation only through Simulink and Hardware Co-simulations on ML605 board with target device Virtex 6 FPGA are shown in Table-1. Here peppers image and Leena image are used as host and watermark images respectively shown in Fig. 13. Four different sub-bands of watermarked image LL, LH, HL and HH, are shown in Table-1. Here, Scaling Factor is taken as 0.2. It is also observed from Table-1 that Hardware Co-simulations results are better than simulation results.

Table-1: Co-hardware simulation results of four sub-bands at the time of embedding ($\alpha=0.2$)

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Sub-bands</th>
<th>Host Sub-bands</th>
<th>Watermark Sub-bands</th>
<th>Watermarked Sub- bands</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>LL</td>
<td><img src="image1.png" alt="Host LL Image" /></td>
<td><img src="image2.png" alt="Watermark LL Image" /></td>
<td><img src="image3.png" alt="Watermarked LL Image" /></td>
</tr>
<tr>
<td>2.</td>
<td>LH</td>
<td><img src="image4.png" alt="Host LH Image" /></td>
<td><img src="image5.png" alt="Watermark LH Image" /></td>
<td><img src="image6.png" alt="Watermarked LH Image" /></td>
</tr>
<tr>
<td>3.</td>
<td>HL</td>
<td><img src="image7.png" alt="Host HL Image" /></td>
<td><img src="image8.png" alt="Watermark HL Image" /></td>
<td><img src="image9.png" alt="Watermarked HL Image" /></td>
</tr>
<tr>
<td>4.</td>
<td>HH</td>
<td><img src="image10.png" alt="Host HH Image" /></td>
<td><img src="image11.png" alt="Watermark HH Image" /></td>
<td><img src="image12.png" alt="Watermarked HH Image" /></td>
</tr>
</tbody>
</table>
## Table 2: Visual comparison between Simulink results and hardware Co-simulation results

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Name of Sub-band</th>
<th>Image through Simulink</th>
<th>Image through Hardware Co-Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>LL</td>
<td><img src="image1" alt="Image through Simulink" /></td>
<td><img src="image2" alt="Image through Hardware Co-Simulation" /></td>
</tr>
<tr>
<td>2.</td>
<td>LH</td>
<td><img src="image3" alt="Image through Simulink" /></td>
<td><img src="image4" alt="Image through Hardware Co-Simulation" /></td>
</tr>
<tr>
<td>3.</td>
<td>HL</td>
<td><img src="image5" alt="Image through Simulink" /></td>
<td><img src="image6" alt="Image through Hardware Co-Simulation" /></td>
</tr>
<tr>
<td>4.</td>
<td>HH</td>
<td><img src="image7" alt="Image through Simulink" /></td>
<td><img src="image8" alt="Image through Hardware Co-Simulation" /></td>
</tr>
</tbody>
</table>

### 4.1.2 Watermark Extraction

With the help of four sub-bands of watermarked image and host image, watermark image is again recovered in the extraction stage. All the four parts of extracted watermark are combined back to give an extracted watermark. Extracted watermark is same as embedded one. The watermark extracted and obtained through matrix viewer is shown in Fig. 14.

![Extracted Watermark](image9)

**Figure 14: Extracted Watermark**

### 4.2 Design Summary and Device Utility for Embedding Process

Fig. 15 shows actual device utilization for embedding process. As RTL view represents the design using Macro blocks. Device utility show how much gates, registers, Look up Tables (LUTs), Flip flops (FFs), Input Output Blocks (IOBs) and RAMs has been utilized for the purpose of designing.
Table-3 shows the comparison of this scheme with schemes [11-16] in terms of device utilization. Authors implemented the system using virtex-6 LX240T board and XSG. Table-3 proves the effectiveness of implemented scheme for slice register and flip-flop utilization as compared to the other schemes. Here utilization is 0% for register and flip flop whereas utilization is 1% [4]. However bonded utilization is 21% in the implemented system.

![Device utility for Embedding Process](image)

**Table 3: Comparison of presented scheme with existing scheme**

<table>
<thead>
<tr>
<th>Research Work</th>
<th>Design Type</th>
<th>Processing Domain</th>
<th>Watermark Type</th>
<th>Area (Logic blocks or mm²)/Bonded I/O</th>
<th>No. of slice Registers</th>
<th>No. of flip flops</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mohanty, Kumara and Sridhara)</td>
<td>FPGA (Xilinx Virtex-II)</td>
<td>Spatial</td>
<td>Invisible Robust</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Ghosh and Sudip)</td>
<td>FPGA (Xilinx Virtex-II Pro)</td>
<td>Spatial</td>
<td>Invisible Robust</td>
<td>1669 LUTs</td>
<td>959</td>
<td>896</td>
</tr>
<tr>
<td>(Maitya and Maity)</td>
<td>FPGA (Xilinx Spartan-3E)</td>
<td>Spatial</td>
<td>Reversible</td>
<td>11291 LUTs</td>
<td>9881</td>
<td>9347</td>
</tr>
<tr>
<td>(Mohanty, Ranganathan and Namballa)</td>
<td>ASIC (0.35µ)</td>
<td>Spatial</td>
<td>Invisible Robust/Fragile</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Mohanty, Ranganathan and Balakrishnan)</td>
<td>ASIC (0.25µ)</td>
<td>DCT</td>
<td>Invisible Robust</td>
<td>16.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Shivdeep, Ghosh and Rahaman)</td>
<td>FPGA</td>
<td>Spatial (PN sequence)</td>
<td>Invisible</td>
<td>20 IOBs</td>
<td>195</td>
<td>188</td>
</tr>
<tr>
<td>(Korrapati, Nelakudit and Mandhala)</td>
<td>FPGA</td>
<td>-</td>
<td>Invisible</td>
<td>50 IOBs</td>
<td>16</td>
<td>156</td>
</tr>
<tr>
<td>Presented Method</td>
<td>FPGA</td>
<td>DWT</td>
<td>Invisible</td>
<td>129 IOBs</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**V. CONCLUSION**

In this paper a method for hardware implementation of image watermarking is presented. Results obtained after hardware simulation are satisfactory and better than existing methods in terms of hardware utilization. In future, this method can be used for implementation of online videos using watermarking. Further, this method can be improved by introducing more robustness and higher security. Still, lot of scope exist to improve robustness for real time implementation of watermarking.
REFERENCES


