A Dual Voltage-Frequency VLSI Chip for Image Watermarking in DCT Domain

Saraju P. Mohanty, Member of IEEE, Nagarajan Ranganathan, Fellow of IEEE, and Karthikeyan Balakrishnan

***Abstract*— In this paper, we present a new VLSI architecture that can insert invisible or visible watermarks in images in the DCT domain. Several low power design techniques such as dual voltages, dual frequency and clock gating have been incorporated in the architecture to reduce the power consumption. The supply voltage levels and the operating frequencies are chosen such that there is a throughput and bandwidth match between low and high operating frequency modules. The proposed architecture exploits pipelining and parallelism extensively in order to achieve high performance. A prototype VLSI chip has been designed and verified using various Cadence and Synopsys tools based on TSMC 0.25μ technology with $1.4M$ transistors and $0.3mW$ of average dynamic power.**

***Index Terms*— discrete cosine transformation, visible and invisible watermarking, dual voltages, dual frequency, low power design, spread spectrum communication, still digital camera.**

I. Introduction

The ease of duplication of digital information has led to the need for effective copyright protection tools. One way to protect digital information would be to hide some special information or data within the digital object. The hidden information must be perceptually and statistically undetectable by others. An example application is to embed a watermark into an object which could contain characteristic information asserting the intellectual property rights of the owner.

Watermarking is the process that embeds data called a watermark, tag or label into a multimedia object for their copyright protection and authentication. Digital watermarks can be divided into different types, such as visible watermark and invisible watermark [1]. A visible watermark is a secondary translucent overlaid onto the primary image and it is visible when carefully inspected. The invisible (robust) watermark is embedded in such a way that alterations made to the pixel values are perceptually unnoticeable and the watermark can be recovered only with the appropriate decoding mechanism.

Several watermarking algorithms available in current literature, however few hardware designs have been reported [2], [3], [4], [17]. The design of custom VLSI hardware can be targeted at achieving better performance, low power and reliability [2], [17]. In this work, we develop a VLSI architecture that can perform both invisible and visible watermarking. The chip architecture is designed using the concepts of dual voltages, dual frequency and clock gating, and exploits pipelining and parallelism for high performance using minimal power. The supply voltage levels and frequency ranges are matched to contain the power consumption of the chip. The architecture uses a decentralized controller mechanism in which each module has its own controller to facilitate implementations of above low power and high performance features.

S. P. Mohanty is with Dept. of Computer Science and Engineering, University of North Texas, Denton, TX 76203, E-mail: smohanty@cs.unt.edu. N. Ranganathan is with Dept. of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, E-mail: ranganat@csee.usf.edu. K. Balakrishnan obtained masters degree from Dept. of Computer Science and Engineering, University of South Florida, Tampa, FL 33620.

II. DCT Domain Watermarking Algorithm

The invisible watermarking algorithm proposed in [6] and the visible watermarking algorithm proposed in [7], [8] are used in the VLSI chip implementation. The various notations used in the description of the algorithms are given in Table I.

A. Invisible-Robust Watermarking Algorithm

In the invisible watermarking technique [6], the watermark is inserted into the spectral components of the image using a method analogous to spread spectrum communication. The watermark insertion consists of the following steps:
(i) DCT of the original image is computed as a single block.
(ii) The perceptually significant regions of the image are found. The set of 1000 largest DCT coefficients is used.
(iii) The watermark $W = \omega_1, \omega_2, ..., \omega_{1000}$ is is constructed from a normal distribution $N(0, 1)$.
(iv) The watermark is inserted in the DCT domain through:

$$C_{I_W}(m, n) = C_I(m, n) + \alpha\omega_i \ . \quad (1)$$

B. Visible Watermarking Algorithm

In the visible watermarking algorithm [7], [8], the insertion function is based on the mathematical models developed for exploiting the human visual system (HVS). The visible watermarking insertion steps are:
(i) The original image I (to be watermarked) and the watermark image W are divided into blocks of size $N_B \times N_B$.
(ii) The DCT coefficients C_I for all the blocks of the original image are computed.
(iii) For each block of the original image the mean gray value is computed as $\mu_{DC\,I_k} = c_{I\,k}(0, 0)$. The normalized mean gray values is calculated using following equation.

$$\mu^*{}_{DC\,I_k} = \frac{\mu_{DC\,I_k}}{\mu_{DC\,I_{max}}} = \frac{c_{I\,k}(0,0)}{Max(c_{I\,k}(0,0))_{\forall k}} \quad (2)$$

Then, the normalized mean gray value of the whole image is,

$$\mu^*{}_{DC\,I} = \frac{N_I \times N_I}{N_B \times N_B} \sum_{k=0}^{\frac{N_I \times N_I}{N_B \times N_B} - 1} \mu^*{}_{DC\,I_k} \ . \quad (3)$$

TABLE I
NOTATIONS USED IN THE DESCRIPTION OF THE ALGORITHM

I	: Original (or host) image (a grayscale image)
W	: Watermark image (a grayscale image)
I_W	: Watermarked image
C_I	: DCT transformed original image
C_W	: DCT transformed watermark image
C_{I_W}	: DCT transformed watermarked image
(m, n)	: A pixel location
$N_I \times N_I$	: Original image dimension
$N_W \times N_W$	: Watermark image dimension
$N_B \times N_B$	: Dimension of a block
$NOIB$	: Number of original image blocks $\left(\frac{N_I \times N_I}{N_B \times N_B}\right)$
$NOWB$	: Number of watermark image blocks $\left(\frac{N_W \times N_W}{N_B \times N_B}\right)$
i_k	: k^{th} block of the original image I
w_k	: k^{th} block of the watermark image W
$c_{I\,k}$	: k^{th} block of the transformed original image C_I
$c_{W\,k}$	: k^{th} block of the transformed watermark image C_W
$c_{I_W\,k}$	: k^{th} block of transformed watermarked image C_{I_W}
α_k	: Scaling factor for k^{th} block (for host image scaling)
β_k	: Embedding factor for k^{th} block (for watermark scaling)
$c_{I\,k}(0,0)$	: DC-DCT coefficient of the k^{th} block DCT block $c_{I\,k}$
$c_{I\,max}(0,0)$	: Maximum of DC-DCT coefficients
$c_{I\,white}(0,0)$	: DC-DCT coefficient of a block with all white pixels
$\mu_{DC\,I_k}$	: Mean gray value of original image block
$\mu_{DC\,I}$	: Mean gray value of the original image I
$\mu_{DC\,I_{max}}$	: Maximum of mean gray value of any i_k
$\mu_{DC\,I_{white}}$	: Mean gray value of a image block with all white pixels
$\mu^*{}_{DC\,I_k}$	: Normalized $\mu_{DC\,I_k}$
$\mu^*{}_{DC\,I}$	: Normalized $\mu_{DC\,I}$
$\mu_{AC\,I_k}$	: Mean of the AC-DCT coefficients of a i_k
$\sigma_{AC\,I_k}$	: Variance of AC-DCT coefficients of a i_k
$\sigma_{AC\,I_{max}}$	: Maximum variance of AC-DCT coefficients of any i_k
$\sigma^*{}_{AC\,I_k}$	: Normalized $\sigma_{AC\,I_k}$
α_{max}, α_{min}	: The maximum and minimum value of α_k
β_{max}, β_{min}	: The maximum and minimum value of β_k
α	: A scaling factor used for invisible watermark insertion
I_{white}	: Gray value corresponding to pure white pixel

(iv) The mean and the variance of the AC-DCT coefficients for each block are calculated using the following equations.

$$\begin{aligned} \mu_{AC\,I_k} &= \tfrac{1}{N_B \times N_B} \textstyle\sum_m \sum_n c_{I\,k}(m,n) \\ \sigma_{AC\,I_k} &= \tfrac{1}{N_B \times N_B} \textstyle\sum_m \sum_n \left\{c_{I\,k}(m,n) - \mu_{AC\,I_k}\right\}^2 \end{aligned} \quad (4)$$

Here, m and n correspond to the locations of the pixels with respect to the k^{th} block of the original image. The normalized variance of the AC-DCT coefficients are computed as follows.

$$\sigma^*{}_{AC\,I_k} \;=\; \frac{\sigma_{AC\,I_k}}{\sigma_{AC\,I_{max}}} \;=\; \frac{\sigma_{AC\,I_k}}{Max\left(\sigma_{AC\,I_k}\right)_{\forall k}} \quad (5)$$

(v) The scaling and embedding factors for each block are,

$$\begin{aligned} \alpha_k &= \sigma^*{}_{AC\,I_k} \;\; exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\} \\ \beta_k &= \tfrac{1}{\sigma^*{}_{AC\,I_k}} \;\; \left[1 - exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\}\right] \end{aligned} . \quad (6)$$

The edge blocks are determined and the α_k and β_k values for edge blocks are taken to be α_{max} and β_{min}, respectively.
(vi) The DCT coefficients C_W for all the blocks of the watermark image are calculated.
(vii) The visible watermark is inserted in the host images block-by-block and watermarked image block is obtained.

$$c_{I_W\,k}(m,n) = \alpha_k \; * c_{I\,k}(m,n) \; + \; \beta_k \; * c_{W\,k}(m,n) \quad (7)$$

C. Modifications to Algorithms for Hardware Implementation

The two watermarking algorithms discussed above are modified with the goals of (i) efficient implementation, (ii) better performance, and (iii) reduced chip area without compromising on the overall quality obtainable from the original algorithms.

As per the original invisible algorithm, the 1000 largest AC-DCT coefficients need to be selected from a total of $N_I \times N_I$ elements, which could degrade the performance of the chip. So, we consider three largest AC-DCT coefficients of a $N_B \times N_B$ block and the process is repeated $\left(\frac{N_I \times N_I}{N_B \times N_B}\right)$ times as,

$$c_{I_W\,k}(m,n) \;\; = \;\; c_{I\,k}(m,n) + \alpha \; * w_k(m,n) \;\; , \quad (8)$$

where, (m, n) corresponds to the three largest AC-DCT values (three neighboring lowest frequency) for k^{th} block. The watermark block w_k is constructed from the random numbers. This block-by-block processing will be very suitable for pipeline processing, for example, when a stage of pipeline handle one block, its previous stage of the pipeline can handle a new block, thus greatly improving overall throughput.

The edge detection task performed using the Sobel operator in the visible watermarking algorithm, is replaced by a DCT domain technique for better efficiency in terms of hardware design. The first step in edge detection involves the summation of the absolute values of all the AC-DCT coefficients in each block.

$$\left|\mu_{AC\,I_k}\right| \;\; = \;\; \tfrac{1}{N_B \times N_B} \textstyle\sum_m \sum_n \left|c_{I\,k}(m,n)\right| \quad (9)$$

The maximum of the above values is $\left|\mu_{AC\,I_{max}}\right| = Max\left(\left|\mu_{AC\,I_k}\right|\right)$. A block is declared as an edge block if $\left|\mu_{AC\,I_k}\right| > \tau \left|\mu_{AC\,I_{max}}\right|$; τ is a threshold constant.

In Eqn. 2, the normalization is performed using the $c_{I\,max}(0,0)$, the maximum of $c_{I\,k}(0,0)$. The determination of $c_{I\,max}(0,0)$ out of the $\left(\frac{N_I \times N_I}{N_B \times N_B}\right)$ values of $c_{I\,k}$s can slow down the insertion process. So, to improve the performance of the VLSI chip, we use $c_{I\,white}(0,0)$ for normalization :

$$\mu^*{}_{DC\,I_k} \;\; = \;\; \frac{\mu_{DC\,I_k}}{\mu_{DC\,I_{white}}} \;\; = \;\; \frac{c_{I\,k}(0,0)}{c_{I\,white}(0,0)} \;\; . \quad (10)$$

The use of normalized numbers as in Eqn. 5 increases significantly the amount of hardware and computations required, especially due to division operation. Thus, the following modifications could be made to improve speed. Eqn. 6 can be modified to get the following equation by applying Eqn. 5.

$$\begin{aligned} \alpha_k &= \tfrac{\sigma_{AC\,I_k}}{\sigma_{AC\,I_{max}}} \;\; exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\} \\ \beta_k &= \tfrac{\sigma_{AC\,I_{max}}}{\sigma_{AC\,I_k}} \;\; \left[1 - exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\}\right] \end{aligned} \quad (11)$$

It is evident from the above equation that the factor $\sigma_{AC\,I_{max}}$ basically serves as a constant scaling factor. So, we remove the constant factor and redefine the equations as follows.

$$\begin{aligned} \alpha^c_k &= \sigma_{AC\,I_k} \;\; exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\} \\ \beta^c_k &= \tfrac{1}{\sigma_{AC\,I_k}} \;\; \left[1 - exp\left\{-(\mu^*{}_{DC\,I_k} - \mu^*{}_{DC\,I})^2\right\}\right] \end{aligned} \quad (12)$$

where, the α^c_k and β^c_k values are current values of α_k and β_k, respectively. Then, the α^c_k and β^c_k values are scaled to the range $(\alpha_{min}, \alpha_{max})$ and $(\beta_{min}, \beta_{max})$, respectively. The above equations contain exponentials, which can be approximated using Taylor series.

III. Proposed VLSI Architecture

The architecture for the proposed chip is shown in Fig. 1.

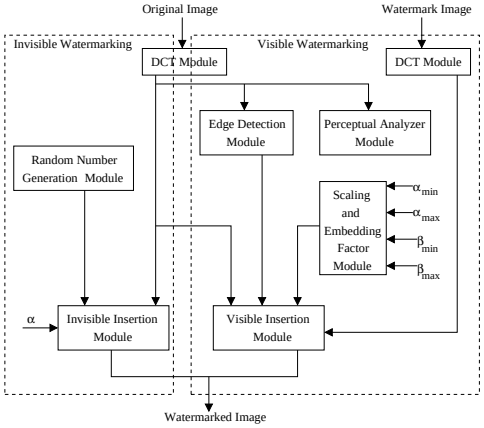


Fig. 1. Proposed Architecture for DCT Domain Watermarking Chip

A. Architecture for Invisible Watermarking

The invisible watermarking architecture consists of three modules as shown in Fig. 1. The original image is given as input to the DCT module that determined the DCT coefficients. The random number generation module generates pseudo-random numbers using a linear feedback shift register (LFSR) [15]. The outputs of both the DCT module and the random number generator module are given to the insertion module.

1) DCT Module: The DCT module consists of the following three sub-modules: (i) DCT_X, (ii) DCT_Y, and (iii) Controller. Flip-flops and latches are used to store and forward the appropriate AC-DCT coefficients to the insertion module. Both the DCT_X and DCT_Y modules have similar architecture [9], [10] and operate on a 4×4 block, but differ in their input and output widths. While calculating the DCT coefficients, the coefficients needed by the insertion module are stored and forwarded with the help of latches and flip-flops. The memory addresses where the coefficients are to be stored are handled by the controller, which also determines the time to trigger the insertion module and the random number generation module.

2) Invisible Insertion Module: The insertion module consists of a multiplier, an adder, and its own controller. The generated pseudo-random number is scaled with α and is added to the image DCT coefficients.

B. Architecture for Visible Watermarking

The proposed architecture for visible watermarking consists of five modules as shown in Fig. 1.

1) DCT Module: The architecture of the DCT module is the same as the one discussed in the previous subsection. There are two DCT modules for computing simultaneously the DCT of the original image and the watermark image.

2) The Edge Detection Module: The edge detection module determines the edge blocks in the original image (Fig. 2) using the Eqn. 9. The edge detection module is divided into three sub-modules. The first module called the accumulator finds the summation of the AC-DCT coefficients of the original image block. This sub-module operates on a single DCT 4×4 image block at a time. The summation values are passed on to the second sub-module called Max-Finder which determines their maximum. The result of this sub-module is then passed onto the third sub-module called the detector.

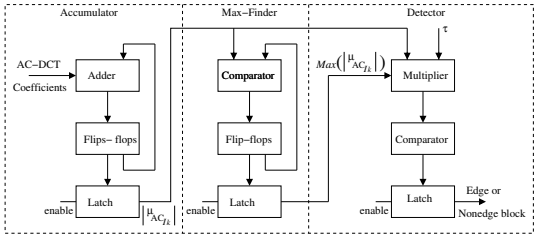


Fig. 2. Architecture of the Edge Detection Module

3) The Perceptual Analyzer Module: The perceptual analyzer module evaluates the Eqn. 2 and Eqn. 5 as shown in Fig. 3. The perceptual analyzer works on one DCT 4×4 image block at a time as soon as it is available. The first sub-module of the perceptual analyzer, is the DC-Mean. The second sub-module, namely the AC-Mean computes the mean of the AC-DCT coefficients. The summation of the AC-DCT coefficients of a block calculated using an adder and feedback flip-flops is shifted by 4 bits for division by 16. The result of this sub-module is passed onto the next sub-module called the AC-Variance that calculates the variance in the AC-DCT.

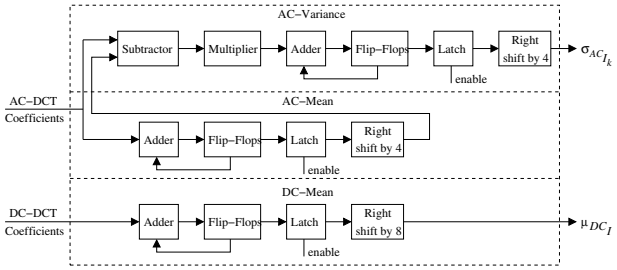


Fig. 3. Architecture of the Perceptual Analyser Module

4) The Scaling and Embedding Factor Module: The scaling factor α_k and the embedding factor β_k are computed by the scaling and embedding factor module shown in Fig. 4. The Taylor series approximated version of Eqn. 12 is evaluated by this module. The α_k and β_k values thus obtained are scaled to a specific range by a scaling module based on a fixed range from $(\alpha_{min}, \alpha_{max})$ to $(\beta_{min}, \beta_{max})$.

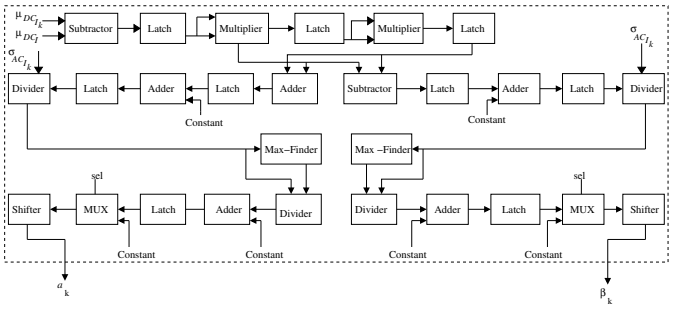


Fig. 4. Architecture of the Scaling and Embedding Factor Calculation

5) The Visible Insertion Module: Using the information provided by the edge detection module and the scaling and embedding factor module, visible watermark insertion module inserts the watermark into the original image. The architecture for this module consists of two multipliers and an adder for evaluating the Eqn. 7.

C. Integrated Architecture

In this section, we discuss certain features incorporated in the proposed design to address low power and high performance issues.

We now describe the low power and high performance features of the proposed architecture.

1) Dual Voltage, Dual Frequency and Clock Gating: The incorporation of dual voltage and dual frequency is shown in Fig. 5. Apart from the dual clock supplies, local clocks are automatically generated using localized controllers to trigger the operation of some modules. This type of clock generation within the chip helps to indirectly implement the clock gating. The architecture is developed in such a way that the clock for the non-DCT modules must be an exact multiple of the clock for the DCT module. The DCT block processes 4 image pixels at a time. The other modules in the chip operate on one pixel at a time. Hence, the DCT block can be clocked at one fourth the non-DCT clock frequency. The slack introduced in the DCT module makes it possible to operate it at a lower voltage.

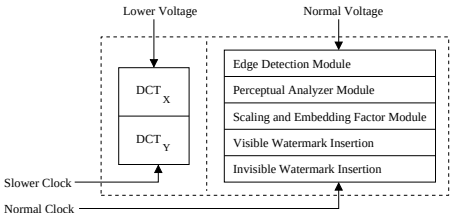


Fig. 5. Dual Voltage and Dual Frequency Operation

2) Pipelining and Parallelism: To improve the overall performance of the chip, some of the computations can be performed with temporal parallelism (using pipelining) and spatial parallelism (using parallel hardware). The visible watermarking hardware involves a three stage pipeline, whereas the invisible architecture is a two-stage pipeline (Fig. 6). In the first stage for the forwarding logic, the latches store all the DCT coefficients. These are then multiplexed as needed to the perceptual analyzer and the edge detection modules.

3) Decentralized Controller: The chip design is based on a decentralized control logic. The DCT module uses latches and demultiplexers to forward the outputs of the module to the edge detection module and the perceptual analyzer module. Appropriate control signals are generated by the main controller to trigger edge detection and perceptual analyzer module. The scaling and embedding factor module can operate only after the perceptual analyzer completes its entire operation, and hence is triggered using the perceptual analyzer module. When the scaling and embedding factor module completes its processing, it triggers the visible insertion module.

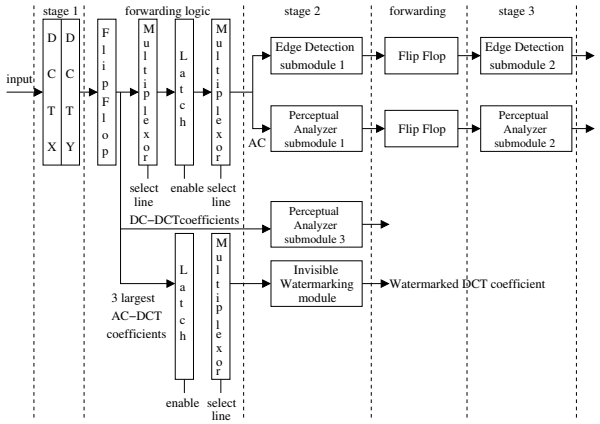
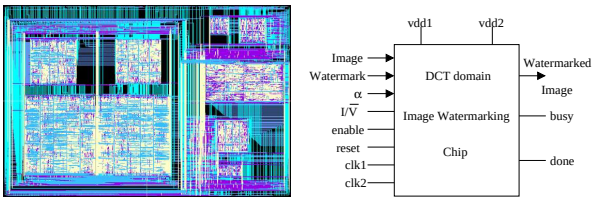


Fig. 6. Pipeline Stages in the Datapath

IV. Prototype Chip Implementation

The prototype chip implementation was designed using a hierarchical design flow approach. At the RTL-level, the architecture was designed using VHDL and the standard cell design methodology was used for generating the layout. The standard cell design library from [11] is based on the TSMC 0.25μ CMOS technology. The standard cell library included basic gates, flip-flops, IO pads and corner cells. The layout for each module was generated and later integrated to obtain the final chip. The layout and pin diagram of the complete chip are shown in Fig. 7(a) and Fig. 7(b), respectively.



(a) Layout (b) Pin Diagram

Fig. 7. The Prototype Chip

The design constraints such as area, power and delay were satisfied using a trial and error approach. The reduction of area and delay is possible while generating the structural netlist. The final area of the generated layout depends on the execution of the Silicon Ensemble tool. Both area and delay can be reduced simultaneously until they reach equilibrium or acceptable value range. The clock frequency is fixed in the Design Analyzer to optimize the design. Once the critical delay for the circuit is determined, the clock frequency is again adjusted.

The integer arithmetic operations like multiplication and addition are performed using the built-in function available in the IEEE.std_logic_arith package [12]. These components were implemented using simple and straightforward algorithms [13], [14], and the synthesis was done using standard cells in the library by Synopsys Design Analyzer. The divider within the embedding module uses the restoring division algorithm. The wordlengths correspond to the bit width necessary to process an 8-bit gray scale image, to process its DCT co-

efficients, to take into account the positive as well as negative AC-DCT coefficients.

The chip is implemented with dual voltage and dual frequency supplies. The single supply voltage level converter described in [16] is used in this implementation. The voltage level converter was designed as a standard cell and added to the existing standard cell library. The output of the DCT module is connected to the voltage level converters to step up the voltage. The delay caused by the voltage level converter is added with the clock period of the faster clock. The overall chip layout consists of two separate voltage islands, such as low voltage and high voltage. The low voltage and the high voltage island layouts are generated separately. When the final layout combining the two islands is generated the power supplies for the two islands are given from a common power ring. Later, the connection to low voltage island is deleted and a separate connection is given from low voltage V_{dd} pin. Silicon Ensemble joins the two power rings, low and high together and in Virtuoso, they are separated.

V. Experimental Results

Each module in the chip was tested individually with Nanosim. Typically, the length of netlist files were more than few hundred thousands of lines. A perl script takes two files as input; the netlist file and a file containing the input for the circuit alongwith a list of input and output pins. The script converts the netlist file into a EPIC VECTOR file and EPIC DIGITAL VECTOR file as required by Nanosim. All the node numbers are also changed to corresponding node names as used in layout for easy identification of the waveforms. The chip was estimated to operate at a dual frequency of $280MHz$ and $70MHz$ and at a dual voltage of $1.5V$ and $2.5V$ consuming $0.3mW$ of average power.

(a) Original [7], [8] (b) Watermarked

Fig. 8. Original and Watermarked Lena Image

We compared the values of scaling (α_k) and embedding (β_k) factors for hardware and software schemes to verify whether the proposed chip produces results as effective as that of the software implementations. It is observed that values of scaling and embedding factors obtained from the chip and that from the softwares are approximately the same. Further, as suggested by [2], [8], we calculated the signal-to-noise ratio (SNR) of the watermarked images. We then compared the SNR of the watermarked images obtained using the proposed chip with that of the watermarked images obtained using the software schemes. The SNR in both hardware and software schemes found to be approximately same (in the range of $24 - 28dB$ for visible watermarking and in the range of $32 - 35dB$ for invisible watermarking); further proving the correctness of the proposed chip. We carried out extensive simulation with various image data and presented one of the visible watermarked images in Fig. 8. The test images of size 256×256 are borrowed from [7], [8] for the simulations.

VI. Conclusions

We presented the VLSI architecture of a watermarking chip and its implementation using 0.25μ technology. The chip is capable of inserting both visible and invisible watermarks into an image. Dual voltage, clock gating and dual frequency techniques were used in this design for low power optimization along with a certain degree of pipelining and parallelism. The architecture developed in this design is the first hardware design with the capability to perform both visible and invisible watermarking in the DCT domain.

References

[1] F. Mintzer, G. Braudaway, and M. Yeung, "Effective and Ineffective Digital Watermarks," in *Proceedings of the IEEE International Conference on Image Processing*, 1997, vol. 3, pp. 9–12.

[2] N. J. Mathai, D. Kundur, and A. Sheikholeslami, "Hardware Implementation Perspectives of Digital Video Watermarking Algortithms," *IEEE Trans on Signal Processing*, vol. 51, no. 4, pp. 925–938, April 2003.

[3] T. H. Tsai and C. Y. Lu, "A Systems Level Design for Embedded Watermark Technique using DSC Systems," in *Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication Systems*, 2001.

[4] A. Garimella, et. al., "VLSI Impementation of Online Digital Watermarking Techniques with Difference Encoding for the 8-bit Gray Scale Images," in *Proc of the Intl Conf on VLSI Design*, 2003, pp. 283–288.

[5] L. D. Strycker, et. al., "Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor," *IEE Proceedings on Vision, Image and Signal Processing*, vol. 147, no. 4, pp. 371–376, Aug 2000.

[6] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, Dec 1997.

[7] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A DCT Domain Visible Watermarking Technique for Images," in *Proc of the IEEE International Conf on Multimedia and Expo*, 2000, pp. 1029–1032.

[8] S. P. Mohanty, "Digital Watermarking of Images," Thesis, Electrical Engineering, Indian Institute of Science, Bangalore, India, 1999.

[9] M. Kaul, R. Vemuri, S. Govindarajan, and I. Ouaiss, "An Automated Temporal Partitioning and Loop Fission approach for FPGA Based Reconfigurable Synthesis of DSP Applications," in *Proceedings of the IEEE/ACM Design Automation Conference*, 1999, pp. 616–622.

[10] S. Govindarajan, I. Ouaiss, M. Kaul, V.Srinivasan, and R. Vemuri, "An Effective Design Systems for Dynamically Reconfigurable Architectures," in *Proceedings of the Sixth Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 1998, pp. 312–313.

[11] J. B. Sulistyo and D. S. Ha, "Developing Standard Cells for TSMC $0.25\mu m$ Technology under MOSIS DEEP Rules," Technical Report, Electrical and Computer Engineering, Virginia Tech, 2002.

[12] P. J. Ashenden, *The Designer's Guide to VHDL*, Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[13] K. Miller, *Assembly Language Introduction to Computer Architecture: Using the Intel Pentium*, Oxford University Press, 1999.

[14] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design : A Systems Perspective*, Addison Wesley, Boston, MA, USA, 1999.

[15] V. P. Nelson, H. T. Nagle, J. D. Irwin, and B. D. Caroll, *Digial Logic Analysis and Design*, Prentice Hall, New Jersey, USA, 1995.

[16] R. Puri, et. al.s, "Pushing ASIC Performance in a Power Envelope," in *Proceedings of the Design Automation Conference*, 2003, pp. 788–793.

[17] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "A VLSI Architecture for Visible Watermarking in a Secure Still Digital Camera (S^2DC) Design," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, Vol. 13, No. 7, July 2005, pp. 808-818.