Design of a Low Power Image Watermarking Encoder using Dual Voltage and Frequency

Saraju P. Mohanty
Dept. of Computer Science and Engineering
University of North Texas, Denton, TX 76203
Email: smohanty@cs.unt.edu

N. Ranganathan and K. Balakrishnan
Dept. of Computer Science and Engineering
University of South Florida, Tampa, FL 33620
Email: ranganat@csee.usf.edu

Abstract

In this paper, we propose a VLSI architecture and provide prototype implementation of a chip that can insert both invisible and visible watermarks in DCT domain. To our knowledge, this is the firstever low power watermarking chip having such watermarking functionalities. Various techniques, such as multiple voltages, multiple frequency, and clock gating are incorporated to reduce power consumption of the chip. The proposed architecture has a three stage pipeline structure and also uses parallelism to improve the overall performance. A prototype chip is designed and verified using various Cadence and Synopsys tools using TSMC 0.25μ technology. It runs at a dual frequency of $280MHz$ and $70MHz$ and at a dual voltage of $2.5V$ and $1.5V$ and contains $1.4M$ transistors. The average power consumption of the chip is estimated to be $0.3mW$, which is five times less than its single supply voltage and single frequency operation.

1 Introduction

Watermarking is the process that embeds data called a watermark, tag or label into a multimedia object, such as images, video or text for their copyright protection and authentication. The invisible watermark is embedded in such a way that alternations made to the pixel value is perceptually not noticed, whereas, in visible watermarking a secondary translucent is overlaid into the primary image. In invisible watermarking scheme, the watermark is detected or extracted to make an assertion about the object. The watermark can be applied in either spatial or frequency domain. Frequency domain is preferred over spatial domain in the applications in which robustness is the most desired characteristics. The above JPEG codec can be a part of a scanner, a digital camera, or any other multimedia device so that the digitized images are watermarked right at the origin. The hardware implementation watermarking schemes has advantages over the software implementation in terms of low power, high performance, and reliability.

A comparative view of the watermarking chips available in current literature is provided in Table 1. Strycker, et. al. [3] proposed the implementation of a real-time spatial domain watermark embedder and detector on a Trimedia TM-1000 VLIW processor. Mathai, et. al. [4] present a chip implementation of the same video watermarking algorithm. A DCT domain invisible watermarking chip is presented by Tsai and Lu [5]. Garimella, et. al. [6] proposed a VLSI architecture for invisible-fragile watermarking in spatial domain. Mohanty, et. al. [1] described a watermarking chip that has spatial domain invisible robust and fragile watermarking functionalities. Mohanty, et. al. [2] propose a chip that has two spatial visible watermarking functionalities.

Table 1: Watermarking Chips in Current Literature

Work	Type	Object	Domain	Chip Statistics
Mathai, et. al. [4]	Invisible Robust	Video	Wavelet	0.18μ
Tsai and Lu [5]	Invisible Robust	Image	DCT	$0.35\mu, 3.0 \times 3.0mm^2$ $3.3V, 50MHz, 62.7mW$
Garimella et. al. [6]	Invisible Fragile	Image	Spatial	$0.13\mu, 3453 \times 3453mm^2$ $1.2V, 37.6\mu W$
Mohanty et. al. [1]	Robust Fragile	Image	Spatial	$0.35\mu, 3.3V,$ $545MHz, 2.0mW$
Mohanty et. al. [2]	Visible	Image	Spatial	$0.35\mu, 3.34 \times 2.89mm^2$ $3.3V, 292MHz, 6.9mW$

In this paper, we propose chip that has both invisible and visible DCT domain watermarking functionalities. The low power feature such as, multiple supply voltages, dynamic frequency clocking, and clock gating have been used to keep the power consumption of the chip at minimal. The performance improvement is done with a three stage pipeline and parallel architecture. Single supply level low power consuming level converters are used to interface modules operating at different supply voltages. The lower supply voltage module is run at a lower frequency. The architecture uses a decentralized controller mechanism to facilitate the dual voltage, dual frequency and clock gating implementations.

2 DCT Domain Watermarking

The spread spectrum invisible watermarking algorithm from [7] and the visible watermarking algorithm from [8] are chosen for VLSI implementation. Table 2 shows the notations used in algorithm description.

Table 2: Notations used in Description of Algorithms

I	: Original (or host) image (a grayscale image)
C_I	: DCT transformed original image
W	: Watermark image (a grayscale image)
C_W	: DCT transformed watermark image
(m,n)	: A pixel location
I_W	: Watermarked image
C_{I_W}	: DCT transformed watermarked image
$N_I \times N_I$	: 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 $\left(\frac{N_W \times N_W}{N_B \times N_B}\right)$
$c_{I\,k}$	: The k^{th} block of the DCT transformed original image C_I
$c_{W\,k}$	: The k^{th} block of the DCT transformed watermark image C_W
$c_{I_W\,k}$	: The k^{th} block of DCT transformed watermarked image C_{I_W}
α_k	: Scaling factor for k^{th} block (used for host image scaling)
β_k	: Embedding factor for k^{th} block (for watermark image scaling)
$c_{I\,k}(0,0)$	: DC-DCT co-efficient of the k^{th} block DCT block $c_{I\,k}$
$c_{I\,max}(0,0)$	: Maximum of DC-DCT co-efficients
$\mu_{DC\,I_k}$	: Mean gray value of original image block i_k
$\mu_{DC\,I}$	: Mean gray value of the original image I
$\mu_{DC\,I_{max}}$	: Maximum of mean gray value of any original image block
$\mu_{DC\,I_{white}}$	: 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 AC-DCT co-efficients of original image block i_k
$\sigma_{AC\,I_k}$	: Variance of AC-DCT co-efficients of original image block i_k
$\sigma_{AC\,I_{max}}$	: Maximum variance of AC-DCT co-efficients of any block
${\sigma^*}_{AC\,I_k}$	: Normalized $\sigma_{AC\,I_k}$
$\alpha_{max}, \alpha_{min}$	: 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

The invisible watermarking algorithm in [7] computes the DCT co-efficients for insertion of watermark assuming that the entire original image is one block. The 1000 largest of these co-efficients are identified as the perceptually significant for the image. The watermark $X = x_1, x_2, ..., x_{1000}$ is computed where each x_i is chosen according to $N(0,1)$, where $N(0,1)$ denotes a normal distribution with mean 0 and variance 1. Assming, $\alpha = 0.1$, the watermark is inserted in the DCT domain of the image using,

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

However, we consider the three largest AC DCT co-efficients of every block as the candidates for invisible watermark insertion in chip implementation.

$$c_{I_W\,k}(m,n) \quad = \quad c_{I\,k}(m,n) \,+\, \alpha\; r_k(m,n) \qquad (2)$$

Where, (m,n) corresponds to the three largest AC DCT values for k^{th} block, and $(k = 0 \rightarrow NOIB - 1)$. The random number matrix r_k is constructed from the random numbers.

The visible insertion algorithm in [8] divides the original image I and the watermark image W into blocks of size $N_B \times N_B$. The DCT co-efficient C_I for all the blocks of the original image are found out. 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 as,

$${\mu^*}_{DC\,I_k} = \frac{\mu_{DC\,I_k}}{\mu_{DC\,I_{max}}} = \frac{c_{I\,k}(0,0)}{c_{I\,max}(0,0)} = \frac{c_{I\,k}(0,0)}{Max(c_{I\,k}(0,0))_{\forall k}}. \qquad (3)$$

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

$${\mu^*}_{DC\,I} = \frac{1}{NOIB}\sum_{k=0}^{NOIB-1} {\mu^*}_{DC\,I_k}. \qquad (4)$$

The mean and variance of AC DCT co-efficients of each block are calculated using the following equations.

$$\begin{aligned} \mu_{AC\,I_k} &= \frac{1}{N_B \times N_B}\sum_m \sum_n c_{I\,k}(m,n) \\ \sigma_{AC\,I_k} &= \frac{1}{N_B \times N_B}\sum_m \sum_n \left(c_{I\,k}(m,n) - \mu_{AC\,I_k}\right)^2 \end{aligned} \qquad (5)$$

Where, the values of m and n corresponds to the locations of each pixel for each k^{th} block with reference to the pixel locations of the original image. The normalized variance of AC DCT co-efficients are computed as follows.

$${\sigma^*}_{AC\,I_k} \quad = \quad \frac{\sigma_{AC\,I_k}}{\sigma_{AC\,I_{max}}} \quad = \quad \frac{\sigma_{AC\,I_k}}{Max\left(\sigma_{AC\,I_k}\right)_{\forall k}} \qquad (6)$$

The scaling and embedding factor for each block are,

$$\begin{aligned} \alpha_k &= {\sigma^*}_{AC\,I_k}\; exp\left(-({\mu^*}_{DC\,I_k} - {\mu^*}_{DC\,I})^2\right) \\ \beta_k &= \frac{1}{{\sigma^*}_{AC\,I_k}}\left(1 - exp\left(-({\mu^*}_{DC\,I_k} - {\mu^*}_{DC\,I})^2\right)\right). \end{aligned} \qquad (7)$$

The α_k and β_k are scaled to the range $(\alpha_{min}, \alpha_{max})$ and $(\beta_{min}, \beta_{max})$, respectively. The edge blocks are determined, and the α_k and β_k are taken to be α_{max} and β_{min} for them, respectively. The DCT co-efficient C_W for all the blocks of the watermark image are found out. For $k = 0 \rightarrow NOWB - 1$, the visible watermark is inserted in the host images block-by-block as,

$$c_{I_W\,k} \quad = \quad \alpha_k\; c_{I\,k} \,+\, \beta_k\; c_{W\,k}. \qquad (8)$$

In the original algorithm Sobel edge detector is used, but we use a DCT based edge detection scheme in chip implementation. The first step of edge detection involves summation of the absolute values of all AC DCT coefficients of each block as follows.

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

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

In Eqn. 3 the normalization is performed using the $c_{I\,max}(0,0)$, the maximum of $c_{I\,k}(0,0)$. Finding $c_{I\,max}(0,0)$ out of $\frac{N_I \times N_I}{N_B \times N_B}$ 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;

$c_{I_{white}}(0,0)$ is the DC DCT of a block having all white pixels. Thus, the Eqn. 3 is modified to the following:

$$\mu^*{}_{DCI_k} = \frac{\mu_{DCI_k}}{\mu_{DCI_{white}}} = \frac{c_{I_k}(0,0)}{c_{I_{white}}(0,0)} \tag{10}$$

Now, we aim at improving the performance degradation due to normalization involved in Eqn. 6. Using Eqn. 6 in Eqn. 7, we have the following equation.

$$\begin{aligned} \alpha_k &= \frac{\sigma_{ACI_k}}{\sigma_{ACI_{max}}}\, exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right) \\ \beta_k &= \frac{\sigma_{ACI_{max}}}{\sigma_{ACI_k}}\, \left(1 - exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right)\right) \end{aligned} \tag{11}$$

The factor $\sigma_{ACI_{max}}$ in Eqn. 11 serves as a constant scaling factor. Hence, we redefine the equations as follows.

$$\begin{aligned} \alpha_k^c &= \sigma_{ACI_k}\, exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right) \\ \beta_k^c &= \frac{1}{\sigma_{ACI_k}}\, \left(1 - exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right)\right) \end{aligned} \tag{12}$$

Where, the α_k^c and β_k^c values are current values of α_k and β_k, respectively. Using Taylor series approximation upto the square term, the above equations are rewritten as follows.

$$\begin{aligned} \alpha_k^c &= \sigma_{ACI_k}\left(1 - (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2 + (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^4\right) \\ \beta_k^c &= \frac{1}{\sigma_{ACI_k}}\left((\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2 - (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^4\right) \end{aligned} \tag{13}$$

Now, the α_k^c and β_k^c are scaled to the range $(\alpha_{min}, \alpha_{max})$ and $(\beta_{min}, \beta_{max})$, respectively to get the required factors.

3 Proposed VLSI Architecture

The overall architecture for the proposed DCT domain watermarking chip that can insert invisible and visible watermarks is shown in Fig. 1. This is a decentralized controller architecture, where each module has its own controller.

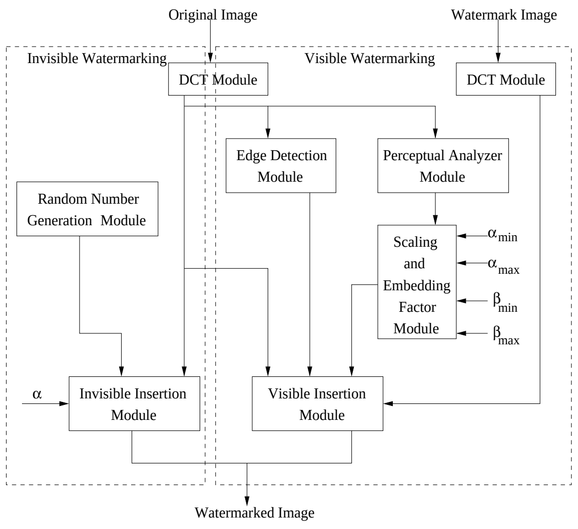


Figure 1: Proposed Architecture

3.1 Invisible Watermarking Architecture

The modules used for invisible watermark insertion are DCT, random number generator, and invisible insertion. After the DCT co-efficients of the host image is calculated, insertion module adds the random numbers to them. The α parameter is also given as input to the insertion module. The three appropriate AC DCT coefficients for each block are chosen for watermark insertion using a counter. The DCT module is shown in Fig. 2(a). The DCT module consists of the following three sub-modules: DCT$_X$, DCT$_Y$, and controller. Apart from the above, flip-flops and latches are also used to store and forward the appropriate AC-DCT coefficients to the insertion module. The architecture of both the DCT$_X$ and DCT$_Y$ modules are borrowed from [9, 10]. Both DCT$_X$ and DCT$_Y$ use sixteen multipliers and twelve adders. The DCT_controller determines the coefficients to be forwarded, the memory addresses where the coefficients are to be stored, the time to trigger the invisible insertion module, and the random number generation module. The invisible watermark insertion module, which consists of a multiplier and an adder, has its own controller. The insertion module scales the random number generated with α and adds it to the DCT coefficient. The random number generation module consists of linear feedback shift registers [11].

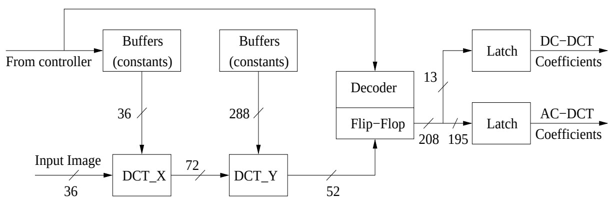
3.2 Visible Watermarking Architecture

The five modules involved in visible watermarking are DCT module, edge detection module, perceptual analyzer module, scaling and embedding factor module, and visible watermark insertion module, details shown in Fig. 2.

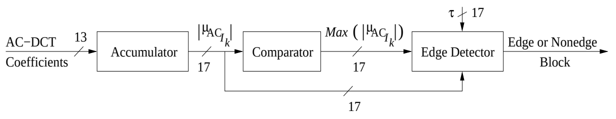
The edge detection module determines the edge blocks in the original image. The threshold constant τ is given as input to the edge detection module. The three parts of the edge detection module implement a particular function, such as accumulation, comparison and detection needed for edge detection (refer Eqn. 9).

The perceptual analyzer module evaluates the Eqns. 3 and 6. Similar to the edge detection module, the perceptual analyzer module is also divided into three sub modules. The first sub module, namely the mean calculator computes the mean of the AC-DCT coefficients. The result of this sub-module is passed onto the next sub-module called the variance calculator module, which calculates the variance in the AC-DCT coefficients. The DC-DCT mean calculator is the third sub-module of the perceptual analyzer. These submodules are implemented with adders, and feedback flip-flops, etc.

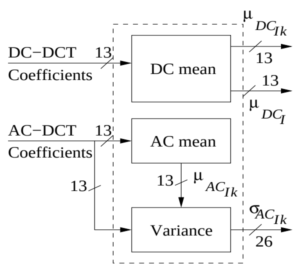
The scaling factor α_k and the embedding factor β_k are computed by the Scaling and Embedding factor module using Eqn. 13. This module is divided into two sub modules. The first module calculates the scaling factors and the embedding factors and is called the alpha-beta module. The second sub module scales down the scaling and embedding factors to a particular range depending on the user defined



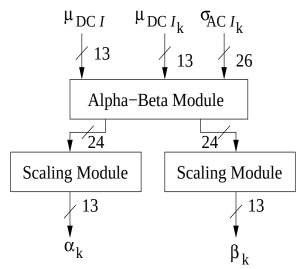
(a) DCT Module



(b) Edge Detection Module



(c) Perceptual Analyser



(d) Scaling and Embedding

Figure 2: Architecture of the individual Modules

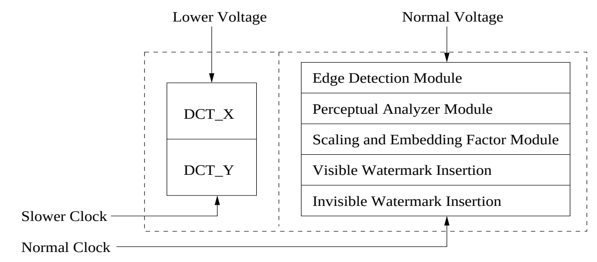

Lower Voltage
Normal Voltage
DCT_X
DCT_Y
Edge Detection Module
Perceptual Analyzer Module
Scaling and Embedding Factor Module
Visible Watermark Insertion
Invisible Watermark Insertion
Slower Clock
Normal Clock

Figure 3: Dual Voltage and Dual Frequency Operation

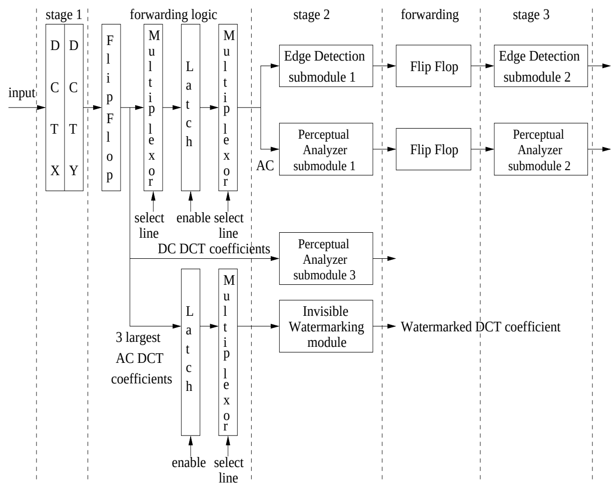

stage 1
forwarding logic
stage 2
forwarding
stage 3
input
DCT X
DCT Y
Flip Flop
Multiplexor
Latch
Multiplexor
select line
enable
select line
Edge Detection submodule 1
Flip Flop
Edge Detection submodule 2
AC
Perceptual Analyzer submodule 1
Flip Flop
Perceptual Analyzer submodule 2
DC DCT coefficients
Perceptual Analyzer submodule 3
3 largest AC DCT coefficients
Latch
Multiplexor
enable
select line
Invisible Watermarking module
Watermarked DCT coefficient

Figure 4: Pipeline Stages in the Datapath

ranges $(\alpha_{min}, \alpha_{max})$ and $(\beta_{min}, \beta_{max})$.

The last module is the visible watermark insertion module. It serves the purpose of inserting the watermark into the original image. Using the information provided by the edge detection module and the scaling and embedding factor module, the watermark is inserted into the original image. It consists of two multipliers and an adder for evaluating the Eqn. 8. Multiplexors are used to select appropriate values of α_k and β_k for a non-edge blocks and an edge blocks.

3.3 Dual voltage, dual clock and clock gating

To minimize the power consumption, the chip is to be operated with dual frequency dual voltage supplies (refer Fig. 3). Apart from the dual clock supplies, local clocks are automatically generated to trigger the operation of some modules. These local clocks are generated from the localized controllers embedded within each module. This type of clock generation within the chip helps to indirectly implement the clock gating technique. A low voltage supply is used for the DCT modules. The chip is implemented 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 delay of the DCT module is less than its clock period. In this way there is a slack introduced in the DCT module which makes it possible to operate the DCT module at a lower voltage.

3.4 Pipelining and Parallelization

To improve the overall performance of the chip, some of the operations involved in the algorithm are pipelined and parallelized as shown in Fig. 4. The visible architecture involves a three stage pipeline, whereas the invisible architecture is a two stage pipeline. The output of the edge detection submodule 2 in stage 3 goes to the edge detection submodule 3. Similarly, the outputs of the other modules in other stages goes to subsequent modules for further processing. The invisible watermarked DCT coefficients are available in the second pipeline stage. In the first forwarding logic, latches store all the DCT coefficients. These are then multiplexed as needed to the perceptual analyzer and the edge detection modules. The edge detection module and perceptual analyzer module operate in parallel. The second forwarding flip flops are internal to respective modules.

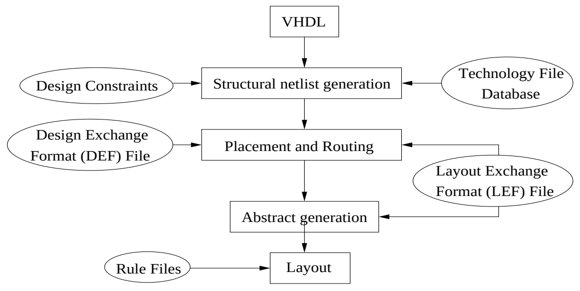


Figure 5: Watermarking Chip Design Flow

3.5 Decentralized Controller

Latches and demultiplexers are used to forward the outputs of the DCT module to the edge detection module and the perceptual analyzer module. The latches store four DCT coefficients at any time. Appropriate control signals are generated by the main controller to trigger edge detection and perceptual analyzer module at the right time. The latches and demultiplexers are controlled by the main controller. 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. The scaling and embedding factor module completes its processing it triggers the visible insertion module. Thus, the overall chip is implemented with decentralized control logic.

4 Prototype Chip Implementation

The chip was designed at the RTL-level using VHDL. A hierarchical design approach was adopted in implementing the chip. Standard cell design methodology was used for generating the layout. The standard cell design library obtained from [12] was designed using TSMC 0.25μ CMOS technology. The standard cell library includes 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 various steps in the design cycle is given as a flow chart in Fig. 5.

All the arithmetic operations done in the chip pertain to the IEEE 754 standard. Arithmetic operations like multiplication and additions are done by calling the in-built function available in the IEEE.std_logic_arith package in step 1 of the design flow. The logic used in this package is the paper and pencil method of carrying out addition and multiplication. These operations are synthesized and implemented using the standard cells in the library by Synopsys design Analyzer in step 2 of the design flow. The divider within the embedding module uses the restoring division algorithm.

The single supply voltage level converter described in [13] is used in this implementation. This voltage level converter is superior to the conventional DCVS in terms of power and delay. 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 low voltage modules are grouped together during the final place and route of the chip. A separate power and ground rings are drawn around this group isolating this low voltage island from the high voltage island. Silicon Ensemble connects the power and ground nodes of all modules and the low voltage island to a common power and ground ring. Since the low voltage island is treated as a separate module, only the outer power and ground rings are used in routing. After exporting the layout of the final chip to the layout editor, the power supply for the low voltage island is deleted and a separate low voltage power supply is given. As a result of separation of the two voltage islands, routing the two different clocks is also made easy.

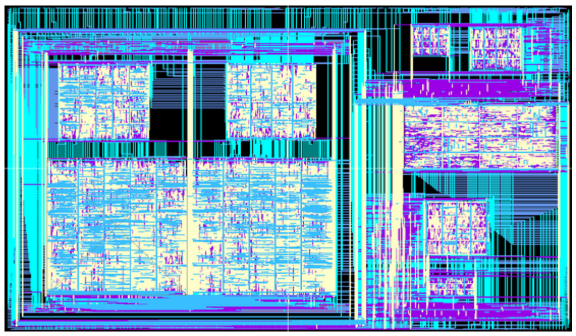


Figure 6: Layout of the Proposed Chip

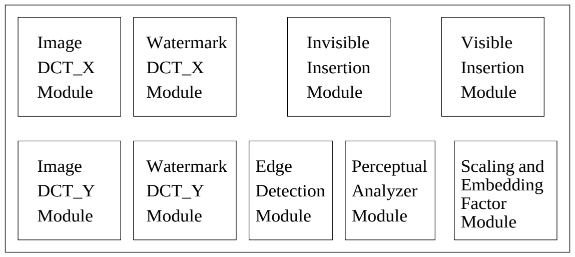


Figure 7: Floorplan of the Proposed Chip

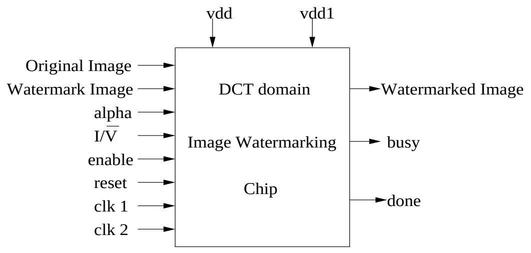


Figure 8: Pin Diagram for the Proposed Chip

The layout and the floorplan of the complete chip are shown in Fig. 6 and Fig. 7 respectively. The pin diagram of the entire chip is shown in Fig. 8. Many other I/Os can be multiplexed to reduce the number of I/O pins. Only the

I/Os of the visible and invisible insertion modules are multiplexed. The choice between invisible watermarking and visible watermarking is made with the help of an $I/\overline{V}$ line. When the $I/\overline{V}$ line is high it represents invisible watermarking and when the $I/\overline{V}$ line is low it represents visible watermarking. During visible watermarking the invisible watermarking part of the chip is disabled and vice-versa. So both visible watermarking and invisible watermarking cannot be carried out at the same time. When the $I/\overline{V}$ line is high the invisible watermarking part of the chip begins to operate. Other modules which belong to visible watermarking part are disabled. Certain IOs in the design are implemented as INOUT pins to reduce the number of IOs. The needed logic to implement this type of IOs is assumed to be present externally. The visible insertion module and the invisible module share the same output pins. A multiplexer is used to select the appropriate output based on the type of watermarking being processed.

Table 3: Power and Area of different Modules

Module	Power (mW)	Area ($micron^2$)
DCT_X		921830.4144
DCT_Y		2551704.12
DCT (at $2.5V$)	1.805985431180	
DCT (at $1.5V$)	0.245994	
Edge Detection	0.004588340388	124749.9792
Scaling and Embeding Factor	0.023358465732	1376637.7680
Visible Insertion	0.004350	314182.6704
Invisible Insertion	0.007900	158818.1904
Perceptual Analyser	0.032035950208	467363.649
Overall Chip		16200000
Overall Chip (dual voltage)	0.3	
Overall Chip (normal operation)	1.95	

5 Results and Conclusions

Each module in the chip was tested individually with nanosim. The functionality and the delay of each module was verified with the help of waveforms. The model file was obtained from MOSIS website. A minimum simulation time of $5000ns$ and a maximum simulation time of $200000ns$ was used. The simulation time depends on the module being tested. Random vectors were used for testing. So the individual power values obtained were added together along with the overhead of the small datapath which lies between the DCT module and the edge detection and perceptual analyzer module. Typical length of netlist files were more than few hundred thousands of lines. Hence, a perl script was written to automate the process of simulation. The perl script takes two files as input. One of them is the netlist file and the other is the file which contains the input for the circuit which should be simulated and a list of input and output pins. The inputs must be given as integers. The script converts the netlist file into a EPIC VECTOR file as required by nanosim. It also generates EPIC DIGITAL VECTOR file which has the inputs in nanosim specified format. All the node numbers are changed to corresponding node names as used in layout. Hence, the output waveforms can be viewed without searching for node numbers from the netlist. The area and power values are given in Table 3.

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 such architecture to be able to perform both visible and invisible watermarking. Further work is necessary to reduced the size of the DCT module by rearranging the arithmetic operations. Particularly, the size of the multipliers can be minimized. More efficient area optimization can be done if each module within the DCT module is placed and routed at the final place and route step. Then, there would be an increase in speed of the DCT module and a decrease in area due to the reduced size of the multipliers. The submodule in the perceptual analyzer has a multiplier followed by an adder. This combinational block dictates the speed of the faster clock. If this block can be divided into two stages with a multiplier and an adder, the speed of the faster clock can be increased further.

References

[1] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Invisible Digital Watermarking Algorithms Towards the Developement of a Secure JPEG Encoder," in *Proc. of the IEEE Workshop on Signal Processing Systems*, 2003, pp. 183–188.

[2] S. P. Mohanty, N. Rangnathan, and R. K. Namballa, "VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera Design," in *Proc. of the 17th Intl. Conf. on VLSI Design*, 2004, pp. 1063–1068.

[3] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, and G. Depovere, "Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor," *IEE Proc. on Vision, Image and Signal Processing*, vol. 147, no. 4, pp. 371–376, Aug 2000.

[4] N. J. Mathai, D. Kundur, and A. Sheikholeslami, "Hardware Implementation Perspectives of Digital Video Watermarking Algortithms," *IEEE Trans. on Signal Processing*, 2003.

[5] T. H. Tsai and C. Y Lu, "A System Level Design for Embedded Watermark Technique using DSC System," in *Proc. of the IEEE Intl. Workshop on Intelligent Signal Processing and Comm. System*, 2001.

[6] A. Garimella, M. V. V. Satyanarayan, R. S. Kumar, P. S. Murugesh, and U. C. Niranjan, "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. 792–796.

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

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

[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 *Proc. 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 System for Dynamically Reconfigurable Architectures," in *Proc. of the Sixth Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 1998, pp. 312–313.

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

[12] J. B. Sulistyo and D. S. Ha, "Developing Standard Cells for TSMC 0.25um Technology under MOSIS DEEP Rules," Tech. Rep., Dept. of Electrical and Comp. Engineering, Virginia Tech, VISC-2002-02, 2002.

[13] R. Puri, L. Stok, J. Cohn, D. Sylvester, A. Srivastava, D. Kung, D. Pan, and S. Kulkarni, "Pushing ASIC performance in a power envelope," in *Proc. of the Design Automation Conference*, 2003, pp. 788–793.