FPGA BASED VIDEO WATERMARKING

BY

PRIYANKA RAVAL

10MCEC30

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2012

FPGA BASED VIDEO WATERMARKING

Project Report

the following student has successfully completed the project

within the given duration in

Master of Technology in Computer Science and Engineering

By

Priyanka Raval

10MCEC30

Guided by

Dr. S.N.Pradhan

Program Coordinator

Department of Computer Engineering

Nirma University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2012

Declaration

This is to declare that this work has not been submitted elsewhere for the use of taking diploma or degree certification. I also here by declare that this is my own work and due acknowledgements and references are cited where ever needed.

Priyanka R. Raval

Certificate

This is to certify that the Major Project entitled "FPGA Based Video Watermarking" submitted by Priyanka R. Raval(10MCEC30), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. S.N.Pradhan
Guide & PG Coordinator
Computer Science & Engg. Department,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D.J.Patel
HOD, Professor,
Computer Science & Engg. Department,
Institute of Technology,
Nirma University, Ahmedabad

Dr K Kotecha
Director,
Institute of Technology,
Nirma University,Ahmedabad

Acknowledgements

With immense pleasure, I would like to present this project report on 'FPGA Based Video Watermarking'. I am very thankful to all those who helped me for the successful completion of this Seminar and for providing valuable guidance throughout the project work.

I would first of all like to offer thanks to **Dr. S.N.Pradhan**, Program Coordinator, Professor & Guide, Institute of Technology, Nirma University, Ahmedabad whose keen interest and excellent knowledge base helped me to direct on right line of thought. His constant support and interest in the subject equipped me with a great understanding of different aspects of the required architecture for the work.

I am thankful to **Prof. Samir Patel** who has guided me for watermarking aspects and helped to narrow down the problem definition. I am also thankful to **Asst. Prof. Aakash Meckwan**, EC Department, Nirma Uinversity, who have solved various difficulties related to electronics hardware and guided me for programming on FPGA kit.

I would like to express my hearty thanks and indebtedness to my parents **Mrs. Daxa Raval** & **Dr. Rajesh Raval** for their continuous encouragement throughout the course and have always given me a real example of 'Vidya Dadati Vinayam'. They gave me an opportunity to do my thesis work and provide all resources required for my project work.

Last but not least I am so much thankful to **God** for standing with me in every movement of my life and using me as instrument for his wonderful and divine tasks.

- Priyanka Raval

10MCEC30

Abstract

With the availability of web cameras, high resolution mobile phones and animation creating softwares, video contents are easy to be created. Various video contents like movies, video clips, audio-visual chat recordings, games are getting more and more popularity over internet and other networks. Wide usage of video content demands effective provision of content ownership and copyright protection.

An effective video processing software should process 24 to 30 frames per second and deal with the complexity of video watermarking algorithm to produce outcome with a reliable speed. In real time video applications, coping up with the speed and complexity makes video watermarking a challenging job.

Most of the research to date for video watermarking has been performed using general purpose processor, application specific processors and DSP processors too.

However as FPGA's have grown in capacity, improved in performance and decreased in cost they have become a viable solution for performing computationally intensive task with the ability to tackle applications for custom chips and programmable DSP devices. Though this involves intensive research on the hardware implementation of video watermarking algorithm.

This project work is mainly based on implementing video watermarking algorithms in FPGA and come up with an effective embedded solution for video ownership and copyright protection.

Contents

List of Figures

LIST OF ABBREVIATIONS

ASIC - Application Specific Integrated Circuits

BSP - Board Support Package

CCL - CMOS configuration latches

CFI - Common Flash Interface

CLB - Configurable Logic Block

CPLD - Complex Programmable Logic Devices

DCT - Discrete Cosine Transform

DSP - Digital Signal Processors

EDK - Embedded Development Kit

ELF - Executable Linked Format

EMC - external memory controller

FFT - Fast Fourier Transform

FPGA - Field Programmable Gate Array

FSL - Fast Simplex Link

GPP - General Purpose Processor

GUI - Graphical User Interface

HDL - Hardware Description Language

IDCT - Inverse Discrete Cosine Transform

IOB - Input Output Block

ISE - Integrated Software Environment

ISF - In System Flash

LMB - Local Memory Bus

LUT - Look Up Table

MDM - Microprocessor Debug Module

MFS - Memory File System

OPB - On-chip Peripheral Bus

PLB - Programmable Logic Board

RAM - Random Access Memory

RTL - Register Transfer Level

SDK - Software Development Kit

SPI - Serial Peripheral Interface

TAP - Test Access Port

XC - Xilinx Chip

XMD - Xilinx Microprocessor Debug Commands

XPS - Xilinx Platform Studio

Chapter 1

Introduction

1.1 Watermark Theory

The process of embedding information into a digital signal which may be used to verify its authenticity or the identity of its owners is known as **Digital watermarking**. An example is a paper bearing a watermark for visible identification. The signal may be audio, pictures, or video in digital watermarking. The information is also carried in the copy if the signal is copied. At the same time, a signal may carry more than one watermark.

In **visible digital watermarking**, the information is visible in the picture or video. Typically, the information is text or a logo, which identifies the owner of the media. A television broadcaster adds its logo to the corner of transmitted video. This is a visible watermark. The objective of visible watermarking is to attach ownership or other descriptive information to the signal in a way that is difficult to remove.

In **invisible digital watermarking**, information is added as digital data to audio, picture, or video, but it cannot be perceived as such (although it may be possible to detect that some amount of information is hidden in the signal). Invisible water-

marking is used in **copyright protection systems**, which are intended to prevent unauthorized copying of digital media. In this use, a copy device retrieves the watermark from the signal before making a copy; the device makes a decision whether to copy or not, depending on the contents of the watermark.

Another application of invisible watermarking is in **source tracing**. A watermark is embedded into a digital signal at each point of distribution. If a copy of the work is found later, then the watermark may be retrieved from the copy and the source of the distribution is known. Source of illegally copied movies can be detected by this technique.

1.2 Motivation

Multimedia applications are increasingly becoming popular nowadays. Sharing of video information over networks and internet is increasing day by day. To provide identity and copyrights to these applications is becoming basic need for multimedia users.

To overcome the challenges in video watermarking and on the parallel not loosing speed and time boundaries of a system are main motivation for an external device which can perform video watermarking.

FPGA technology is having benefits like reduced inventory costs, easy prototyping etc over other technologies like GPP and ASIC. This project utilizes benefits of FPGA technology and attempts to develop an external embedded solution for video watermarking requirements.

1.3 Scope of Work

The previous sections have given reasons why it is interesting and worthwhile to investigate further the exploitation of FPGAs for video watermarking applications. For more detail, the disadvantages of using FPGAs identified previously improve the video watermarking application developer in several ways:

- The designer has to think in terms of hardware architectures rather than algorithms.
- The designer must be able to use some form of hardware description language. The lower the level of the HDL, the more detail the developer must expertise.
- The design cycle, with its numerous intermediate stages, can be much slower than the traditional edit-compile-execute software cycle.

These unfortunate consequences are the underlying motivation for this thesis. The general goal is to support the video watermarking application developer in exploiting FPGAs by providing more appropriate software tools. In particular, the major objectives are:

- To provide a high level-programming environment, this will support to bridge the gap between algorithms and architecture descriptions.
- To hide as much as possible of the details of the FPGA hardware and its environment. This in turn will reduce the learning curve.
- To speed up the design cycle by eliminating some of the intermediate stages, which are currently necessary. This is useful for rapid experimentation, which is important for developing video watermarking applications.

Chapter 2

Literature Survey

Literature survey for this project work is divided in to two sections. Literature survey for video watermarking which identifies challenges in video watermarking and literature survey for FPGA technology which tries to figure out study needs in FPGA.

2.1 Literature Survey for Video Watermarking

This section contains analytical part to find out if real-time video watermarking using FPGA technology is feasible or not. It also includes study of various attacks and evaluation criteria for video watermarking. This section briefs about studied video watermarking techniques which are trying to implement video watermarking using hardware approach. At last this sections is concluded with identifying challenges in real-time video watermarking.

2.1.1 Feasibility Analysis

The exact problem in video watermarking is to deal with the complexity of algorithm and on parallel not loosing the time constraint of real-time video.
If one wants to solve this problem with the help of FPGA technology then one has to prove that the FPGA approach must be enough faster than current approaches.

Figure 2.1 is the analysis of a most cited technical paper that shows the fastness of FPGA approach over GPP for some image processing algorithms. A video is collec-

Timing Result edge detection algorithm on a 256 x 256 gray scale image.

		Xilinx Vertex-E FPGA		Pentium III	
		Freq [MHz]	Time [ms]	Freq [MHz]	Time [ms]
Median Filter, Morphological Operation		25.9	2.56	1300	51
Gaussian convolution	A	25.9	2.62	1300	31
	B	42	1.57	1300	31
Gaussian Smoothing	C	42.03	1.58	1300	16
	D	50.99	1.31	1300	16
Edge detection		16	4.2	1300	47

A: Direct division by 115
B: Division using right shift(>> 7)
C: Direct Multiplication
D: LUT based Multiplication

Figure 2.1: Image Processing Algorithms on Reconfigurable Architecture[16]

tion of number of frames or images passing with a speed so that it creates illusion of moving picture to the human eyes. So, as per the reference of the paper(figure 2.1), The image processing algorithm can become almost 20 times faster using FPGA approach with lowest possible frequency in Vertex-E FPGA. This is a good speed up which creates a hope that video processing also can be made faster enough to solve real-time needs.

One more obvious speed advantage due to FPGA approach is TRUE parallelism exposed by FPGA. Figure 2.2 presents increasing number of watermarks inclusion effects to the video. The table shows that increasing number of watermarks does not increase the time overhead linearly because various logics could be duplicated to get

# 16-bit marks	50	98	162	200	242	288	338	392	450	512
% resources	3.31	6.48	10.71	13.23	16.01	19.05	22.35	25.93	29.76	33.86
% timing	-1.04	-0.47	3.17	-7.15	-4.69	1.65	-11.53	2.47	11.95	-5.23

MIPS R2000 – Impact of number of 16-bit marks on resources and speed

# 16-bit marks	2	50	98	184	288	374	428
% resources	0.05	1.33	2.61	4.90	7.68	9.97	11.41
% timing	-10.74	3.46	-25.93	-7.99	-13.50	10.25	-1.57

ATR - Impact of number of 16-bit marks on resources and speed

# 16-bit marks	2	50	98	158	200	242	298
% resources	0.11	2.86	5.60	9.03	11.43	13.83	17.03
% timing	-22.98	-14.83	-5.07	-1.90	11.05	-11.93	-3.28

DES - Impact of number of 16-bit marks on resources and speed

Figure 2.2: Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks[10]

the speed benefits.

Conclusion of Feasibility Study:

Above analysis of papers related to Watermarking algorithms and FPGA technology shows positive signs for the FPGA approach. It seems feasible to resolve real-time needs and computational complexity of video watermarking algorithms through FPGA technology Though proving this needs implementation of a prototype algorithm using FPGA approach.

2.1.2 Types of Attacks

The designer has to think of the attacks to the watermarking techniques to provide the robustness to the design. Ownership deadlock, counterfeit ownership and forged ownership are the possible threats to a watermarking design[9]. Based on these threats, the attacks can be categorized as ambiguity attack, removal attack, copy attack or key copy attack.

These all attacks can break the service provided by the watermark design. Figure 2.1 shows brief of these attacks and associated threats. The selection of an algorithm should be such that it should not be easily braked by any of these attacks.

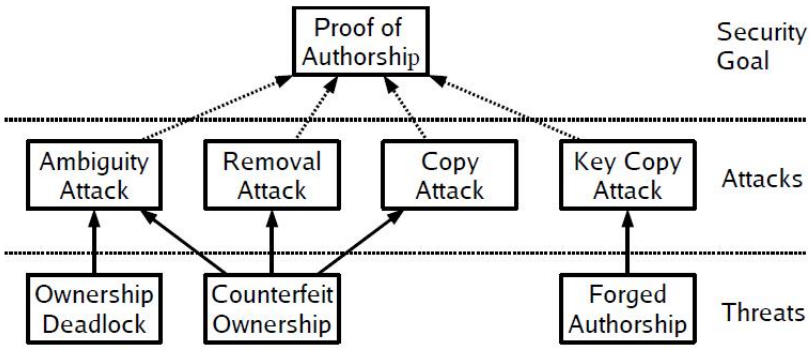

Figure 2.3: An overview of threats, attacks and the watermarking security goal of the proof of authorship[9]

2.1.3 Evaluation Criteria

There are several criteria to evaluate video watermark. These criteria help in choosing an effective video watermarking algorithm. These are the common criteria mentioned in various books and online documents. The video watermark criteria referred in this project work are as below[8]:

a. **Functional correctness:** This is the most important criteria. If the watermark process destroys the functional correctness, it is useless to distribute the core.

b. **Resources overhead:** Many watermark algorithms need some extra resources. Some for the watermark itself, some because of the degradation of the optimization results from the design tools. language. The lower the level of the HDL, the more detail the developer must master.

c. **Transparency:** The watermark procedure should be transparent to the design tools. It should be easy to integrate the watermarking step into the design flow, without altering the common design tools.

d. **Verifiability:** The watermark should be embedded in such a way that simplifies the verification of the authorship. It should be possible to read out the

watermark only with the given product without any further information from the design flow, which must be ordered from the accused company.

e. **Difficult to remove:** The watermark should be resistant against removal. The effort to remove the watermark should be greater than an effort needed to develop a new core or removal of watermark should cause corruptness of the functionality of the core. Watermarks which are embedded into the function of the core are more robust against removal than additive watermarks.

f. **Strong proof of authorship:** The watermark should identify the author with a strong proof. It should be impossible that other persons can claim the ownership of the core. The watermark procedure must be resistant against tampering.

2.1.4 Studied Watermarking Techniques

Below is list of video watermarking papers those are referred during this project work. Various algorithms are being tried for video watermarking using hardware approach. Also some techniques are used to optimize video watermarking computations.

a. Use of Multiple small watermarks for more security.[10]

b. Embed the watermark in the place and route phase of the design cycle(Advantage: No area or timing overhead)[11]

c. Wavelet transform based Visible, invisible, spatial, frequency domain watermarking techniques(Hardware solution to real time video acceleration)[12]

d. Rate-scalable compression to encode the video(One solution for real-time videos)[13]

e. Use of MMX technique(To decrease processing time: This constraint is used in various applications with different scopes. The word in our work defines that a watermark embedding process should not be recognizable during video

playing time. That is, watermark embedding time should be shorter than a frame processing time, so watermarking routine should not make playing time delayed.)[14]

f. Fragile watermarking[15]

2.1.5 Challenges in Video Watermarking

Based on studied watermarking solutions for the videos, we can summarize video watermarking challenges as follows:

a. In video watermarking due to high spatial correlation between successive frames. In most cases, successive frames of video are not independent and have a high degree of similarity. If independent watermarks are embedded on each frame, an attacker could perform frame averaging to remove significant portions of the embedded watermark.

b. Embedding the same watermark in all frames may be insecure, as the attacker would have a lot of information about the structure of the watermark for estimation and removal.

c. Some applications, such as watermarking live content, embedding copy-control information, and embedding fingerprints for content tracking, require real-time watermark embedding.

d. Other applications, such as detecting copy protection information and on-the-fly video authentication, require real-time watermark detection.

e. Embedding a watermark must not significantly increase the data rate of the watermarked video stream, especially for streaming video applications where bandwidth is scarce.

2.1.6 Selected Visible Watermarking Technique

After studying general video watermarking techniques, the author has selected[19] DCT domain video watermarking technique. Figure 2.1 shows the idea of the algorithm.

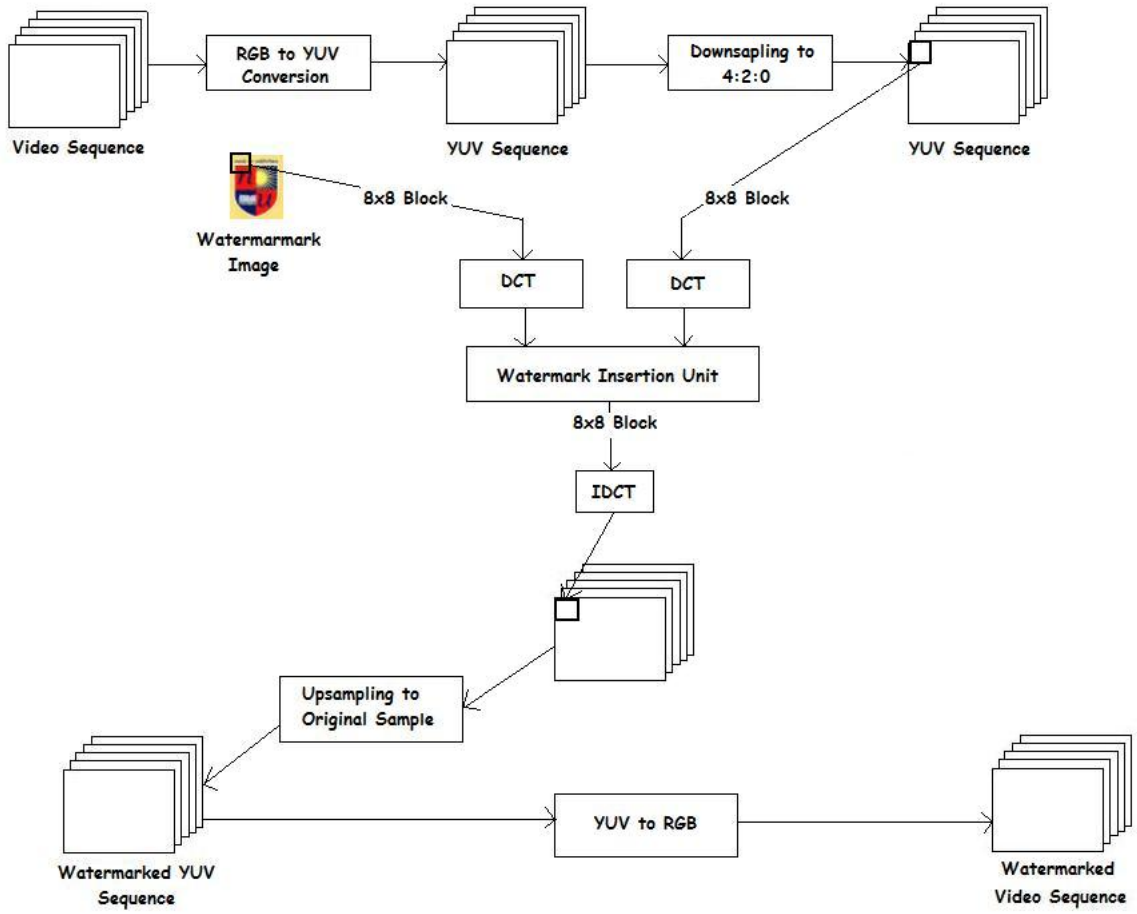

Figure 2.4: Figure explaining algorithm execution

Input video is first converted to YUV format. YUV frames are then down sampled to 4:2:0 sampling format. The down sampled YUV frames will be processed block by block bases.

For each 8x8 block 2D DCT coefficient will be found. According to the pixel values,

90% of the information will be taken from frame's coefficient and 10% of the value will be considered from watermark coefficient and added together to get watermarked video coefficient. Figure 2.5 shows Watermark Insertion Unit in depth.

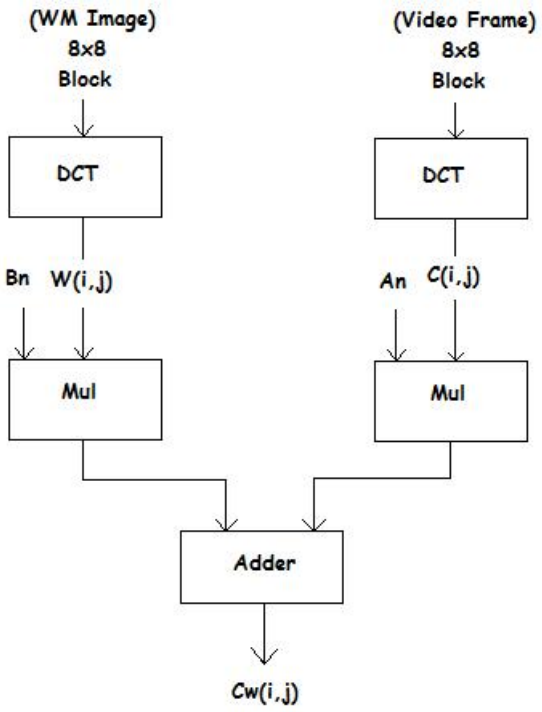

Figure 2.5: Watermark Insertion Unit

The reason for selecting DCT domain video watermarking technique is as below:

a. This technique involves enough computational complexity to test whether FPGA approach can handle complexity of any video watermarking algorithm.

b. It is one of the popular techniques to watermark videos.

c. Calculation of DCT is time consuming process. So if the FPGA approach could produce watermarked video with reliable speed for this algorithm, then FPGA approach can be declared as generic approach for video watermarking at least for speed and complexity criteria.

2.2 Literature Survey for FPGA Technology

This section contains analytical part for FPGA Technology. It starts with analyzing current logic capacity of FPGAs available in the market. It contains basic architecture detail for FPGA. It focuses on specifications and configurations of spartan 6 FPGA which is the target hardware being used for this project. This section also introduces programming language and tools being used in the project work.

2.2.1 Analysis for Logic Capacity of FPGA

A Field-programmable Gate Array (FPGA) is a type of Field Programmable Device(FPD). There are many FPDs available in the market today. According to the logic capacity they can be categorized as, Simple Programmable Logic Devices(SPLDs), Complex Programmable Logic Devices(CPLDs) and Field Programmable Gate Arrays(FPGAs). A graph representing logic capacity of each of these programmable logic devices with reference to their manufacturing firms is shown in figure 2.6.

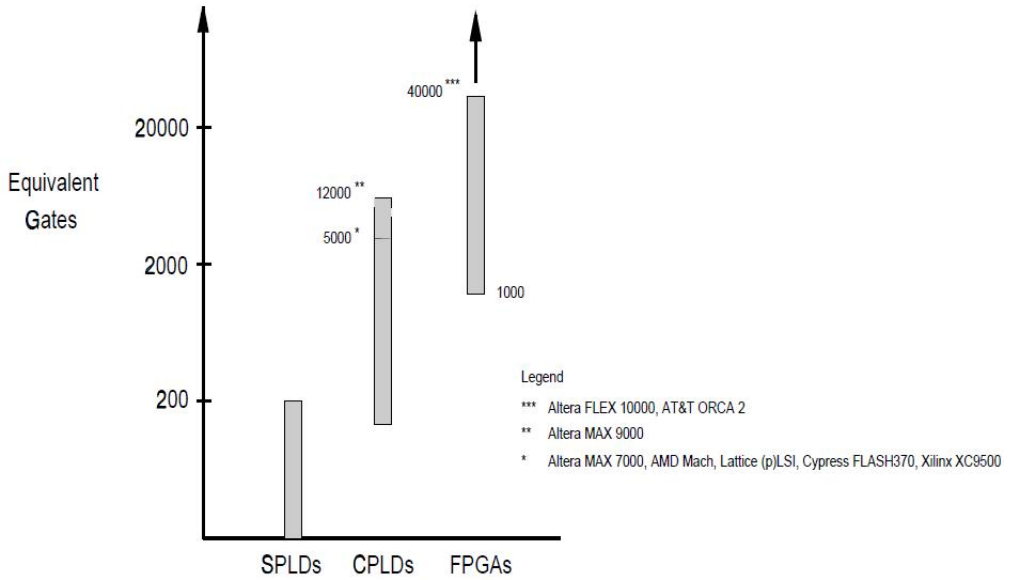

Figure 2.6: FPD Categories by Logic Capacity (Spartan 6 - 150,000 logic cells)[3]

FPGA is a evolving technology since its birth. Efforts are made to grow logic capacity in FPGAs and also to decrease cost of FPGAs. Improvements in FPGAs have reached to a level so that FPGAs have become a viable solution in computationally intensive tasks. Figure 2.6 presents a graph showing logic capacity in FPGAs with respect to other programmable logic devices.

Spartan 6 FPGA is the target hardware to solve video watermarking real-time needs. It is having almost 1,50,000 logic cells capacity. As per Resource utilization study of various video processing researches, logic capacity of spartan 6 very well meets the project requirement.

How FPGAs are advantageous over GPPs and ASICs:

Sr No	GPP	ASIC	FPGA
1	It can be used in variety of environment	Its usage is specific to application	It can be designed specific to the application.
2	Slower, power hungry	Fastest, lower power	Between GPP and ASIC
3	Such devices are ready to use but general purpose	They take months to be fabricated on manufacturing line.	Production slower than GPP because needs to be configured but much more faster than ASIC
4	Uses much more number of transistors than actually required by the application	Uses lesser transistors	Uses more transistors than ASIC and lesser than GPP

Table I: Advantages of FPGA Technology over GPP and ASIC

The role of FPGA in Embedded Systems is gaining importance due to its increasing capabilities and availability of powerful FPGA design software tools. The digital

video applications are driving FPGA market and enabling use of FPGA for broad range of applications.

FPGA devices have got advantages over General Purpose Processor and Application Specific Processor to design an embedded system. Below table tries to describe some important points among the three technologies.

Moreover,below are some application life-cycle specific advantages of FPGAs[1]:

- No wait for the final design. The design can be programmed and tested into FPGA immediately.
- FPGA is excellent prototyping vehicle. Because jump from prototype to product is easier.
- They can be used in several different designs reducing inventory costs.
- Performance gains are obtained by bypassing the fetch-decode-execute overhead of general purpose processors.

Thus,FPGA offers a compromise between the flexibility of general purpose processors and the hardware-based speed of ASICs.

2.2.2 Exploring FPGA Architecture

As shown in figure 2.7, an FPGA is a matrix of logic blocks that are connected by a switching network. The logic blocks and the switching network both are reprogrammable. This allows application specific hardware to be constructed and allows to change the functionality of the system with ease.

FPGA is a silicon chip with unconnected logic gates. It is an integrated circuit containing many identical logic cells that can be viewed as standard components. The

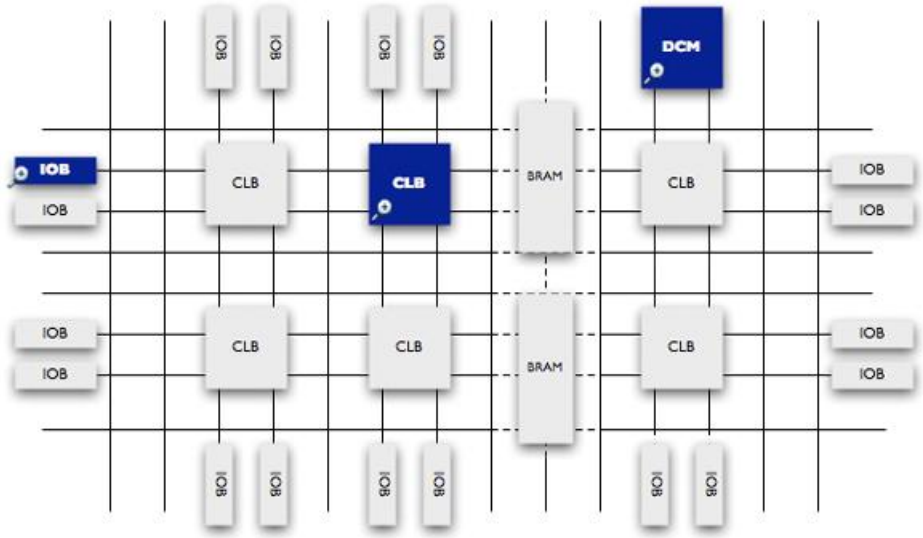

Figure 2.7: Basic FPGA Architecture[2]

individual cells are interconnected by a matrix of wires and programmable switches.

Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up.

The FPGA has three major configurable elements:

a. configurable logic blocks(CLBs)

b. input-output blocks(IOB)

c. Interconnects

Each CLB contains a logic element which is implemented as a lookup table(See figure 2.8). This logic element operates on four one-bit inputs and outputs single data bit. Using CLB any boolean function of four inputs can be performed.

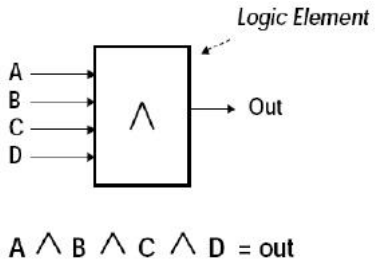

Figure 2.8: Basic CLB Architecture[7]

The Configurable Logic Block is the basic logic unit in an FPGA. Exact numbers of CLBs and features of the CLB vary from device to device. Every CLB consists of a configurable switch matrix with 4 or 6 inputs, some selection circuitry (MUX, etc.), and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers or RAM.

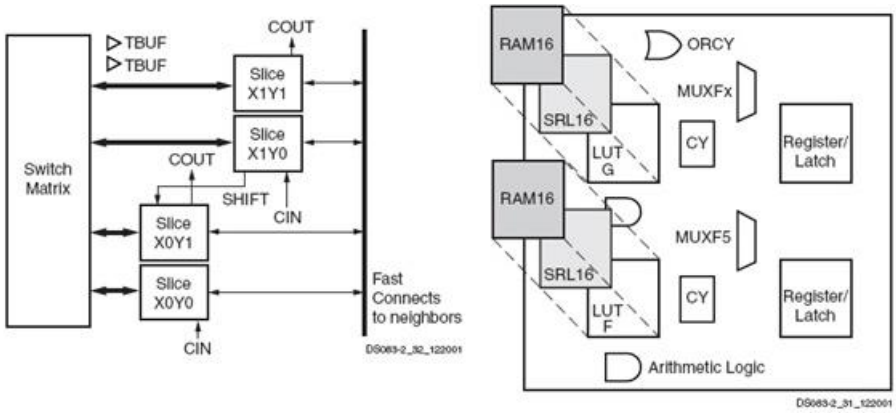

Figure 2.9: Basic CLB Organization[2]

FPGAs provide support for various useful I/O standards. I/O in FPGAs is grouped in

banks with each bank independently able to support different I/O standards. FPGA

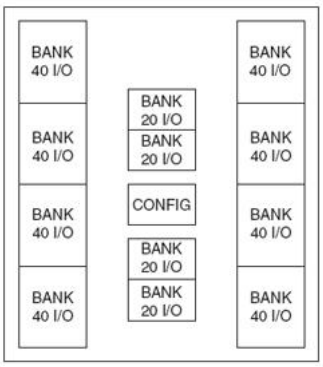

Figure 2.10: I/O Block Banks[2]

architecture supports various kind of interconnects to fulfil interconnection need for various components on board. Short wires, general-purpose wire, global interconnects and specialized clock distribution networks are example of such interconnects.

The reason behind why FPGAs need different types of wires is, because wires can introduce a lot of delay and also wiring networks of different length and connectivity need different circuit designs.

2.2.3 Exploring Spartan 6 FPGA Specifications

For this project work spartan 6 SP605 board is available target hardware device. Figure 2.11 represents snapshot of sp605 board with its features labeled on it. Key features of spartan sp605 board is as below[6]:

Spartan-6 FPGA

- XC6SLX45T-3FGG484 device

Configuration

- Onboard JTAG configuration circuitry

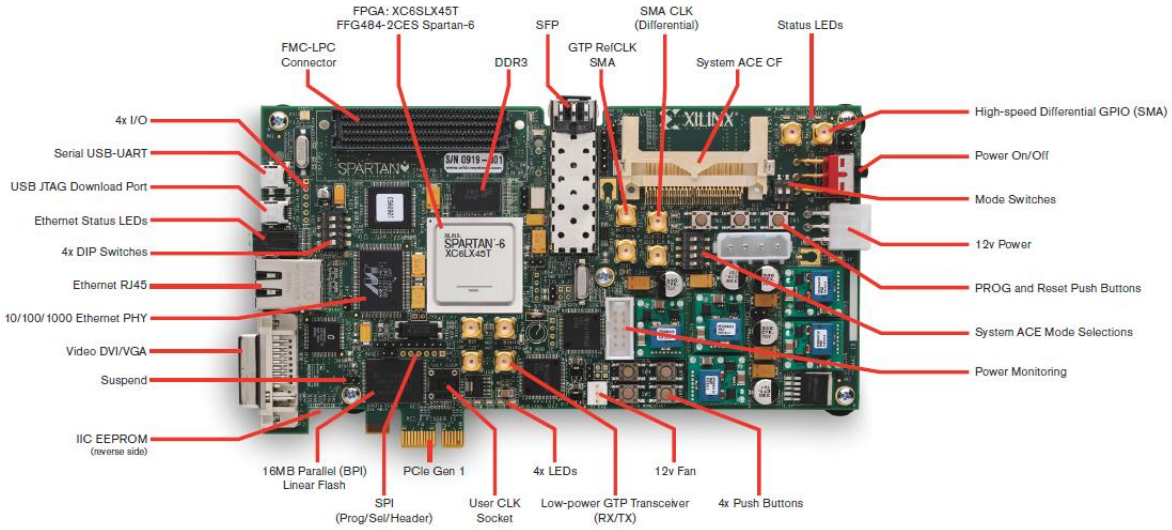

Figure 2.11: Spartan-6 FPGA SP605 Board Features[2]

- 8 MB Quad SPI Flash
- 32 MB Parallel (BPI) Flash
- System ACE CF 2 GB CompactFlash (CF) card

Memory

- 128 MB DDR3 Component Memory
- 32 MB Parallel (BPI) Flash (Also available for configuration)
- 8 Kb IIC EEPROM
- 128 MB Quad SPI Flash (Also available for configuration)

Communications and Networking

- 10/100/1000 Tri-Speed Ethernet
- SFP transceiver connector
- GTP port (TX, RX) with four SMA connectors

- USB To UART bridge
- PCI Express x1 edge connector

Expansion Connectors

- FMC LPC connector (1 GTP transceiver, 68 single-ended or 34 differential userdefined signals)
- User GPIO with two SMA connectors
- 4 user I/O (1 x 6 header)

Power

- 12V wall adapter or ATX
- Voltage and current measurement capability of 2.5V, 1.5V, and 1.2V supplies

There are various options to configure sp605 board with the user program bit-stream file. Section 2.2.4 gives configuration overview of spartan 6 sp605 board.

2.2.4 Spartan SP-605 Configuration Overview

Spartan-6 FPGAs are configured by loading application-specific configuration data a bitstream into internal memory. Spartan-6 FPGAs can load themselves from an external nonvolatile memory device or they can be configured by an external smart source, such as a microprocessor, DSP processor, microcontroller, PC, or board tester.

In any case, there are two general configuration datapaths. The first is the serial datapath that is used to minimize the device pin requirements. The second datapath is the 8- or 16-bit datapath used for higher performance or access (or link) to industry-standard interfaces, ideal for external data sources like processors, or x8- or x16-parallel flash memory.

Like processors and processor peripherals, Xilinx FPGAs can be reprogrammed, in system, on demand, an unlimited number of times.
Because Xilinx FPGA configuration data is stored in CMOS configuration latches(CCLs), it must be reconfigured after it is powered down. The bitstream is loaded each time into the device through special configuration pins.

These configuration pins serve as the interface for a number of different configuration modes:

- JTAG configuration mode
- Master Serial/SPI configuration mode (x1,x2, and x4)
- Slave Serial configuration mode
- Master SelectMAP/BPI configuration mode (x8 and x16)
- Slave SelectMAP configuration mode (x8 and x16)

Master Modes

The self-loading FPGA configuration modes, generically called Master modes. The Master modes leverage various types of nonvolatile memories to store the FPGA configuration information. In Master mode, the FPGA configuration bitstream typically resides in nonvolatile memory on the same board, generally external to the FPGA.

The FPGA provides a configuration clock signal called CCLK (the source is from either an internal oscillator or an optional external master clock source GCLK0/USERCCLK), and the FPGA controls the configuration process.

Slave Modes

The externally controlled loading FPGA configuration modes, generically called Slave modes, are also available with either a serial or byte-wide datapath. In Slave mode,

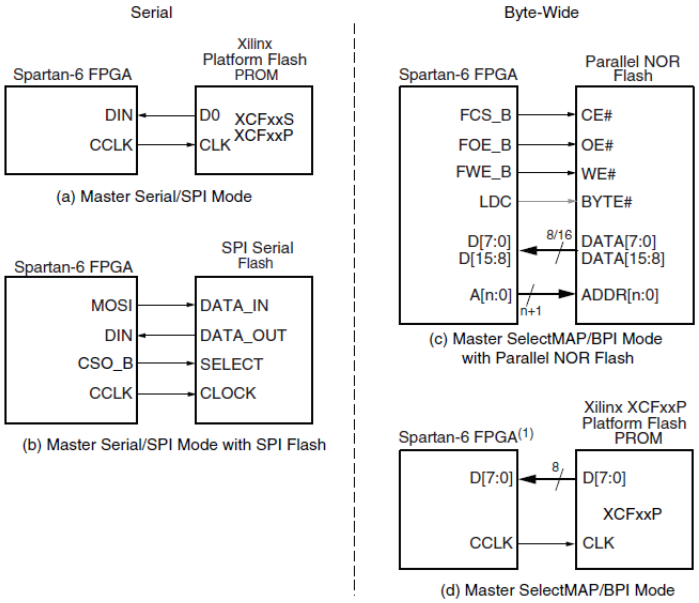

Figure 2.12: Master Configuration Modes[20]

an external intelligent device such as a processor, micro-controller, DSP processor, or tester downloads the configuration image into the FPGA.

The advantage of the Slave configuration modes is that the FPGA bitstream can reside almost anywhere in the overall system. The bitstream can reside in flash, onboard, along with the host processor's code. It can reside on a hard disk. It can originate somewhere over a network connection or another type of bridge connection.

Boundary-Scan and JTAG Configuration

Spartan-6 devices support IEEE Std 1149.1, defining Test Access Port (TAP) and boundary-scan architecture. These standards ensure the board-level integrity of individual components and the interconnections between them.

In addition to connectivity testing, boundary-scan architecture offers flexibility for vendor-specific instructions, such as configure and verify, which add the capability

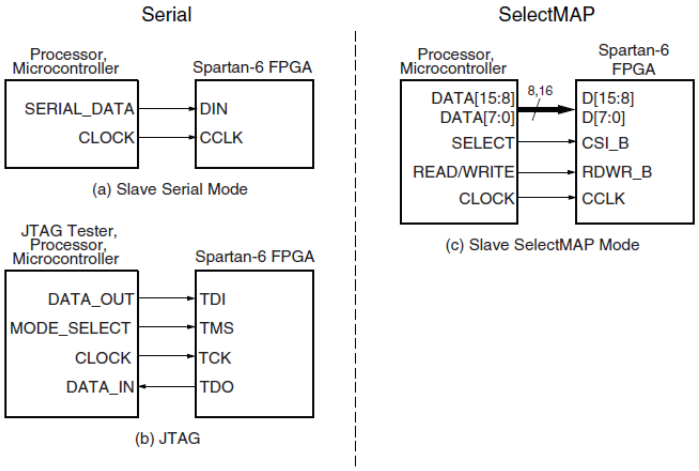

Figure 2.13: Slave Configuration Modes[20]

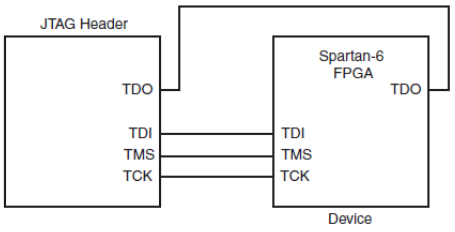

Figure 2.14: Single-Device JTAG Programming Connections[20]

of loading configuration data directly to FPGAs and compliant PROMs. TAP and boundary-scan architecture is commonly referred to collectively as JTAG.

A typical JTAG setup with the simple connection required to attach a single device to a JTAG signal header, which can be driven from a processor or a Xilinx programming cable under control of iMPACT software. TCK is the clock used for boundary-scan operations.

The TDO-TDI connections create a serial datapath for shifting data through the JTAG chain. TMS controls the transition between states in the TAP controller.

2.2.5 Identifying Programming Language and Tools

In order to create an FPGA design, a designer has several options for algorithm implementation. There exist several high-level hardware description languages(HDLs) to design FPGAs.

Verilog HDL:
VHDL and Verilog are the basic proprietary hardware design languages. They were originally came as simulation languages. Verilog is more user friendly compared to VHDL in terms it is having more C - like syntexts[18].

Handel-C:
Handel-C is a programming language developed by the Hardware Compilation Group at Oxford University Computing Laboratory, and now sold by Celoxica Ltd. Handel-C is a C-based language with true parallelism and priority-based channel communication. It can be compiled to hardware. Handel-C is having capability of optimized logic synthesis.

Catapult-C:
The Catapult C Synthesis tool from Mentor Graphics. It targets compute-intensive applications such as wireless communication, satellite communication and video/image processing etc, in ASICs and FPGAs.

SystemC:
SystemC is a language built in standard C++ by extending the language with the use of class libraries. SystemC addresses the need for a system design and verification language for hardware and software. It is particularly suited to model system's partitioning, to evaluate and verify the assignment of blocks and measure the interactions between and among functional blocks. It is used by leading companies for the

development of Intellectual Property Cores(IP Cores).

Mitrion-C and Mitrion-IDE:

Mitrion-C is a C-like FPGA programming language that has been developed by Mitrionics. It addresses the needs of scientific programmers. The tools provided include the Mitrion Integrated Development Environment (IDE) and the Mitrion Virtual Processor. The IDE provides all the tools necessary for synthesizing and simulating Mitrion-C code as well as standard libraries and libraries for hardware support and simulation.[5]

Nallatech DIME-C and DIMETalk:

Nallatechs DIME-C language is based on a subset of ANSI-C. This gives it a number of obvious advantages over both Handel-C and Mitrion-C. Firstly, the programmer does not need to learn the syntax or semantics of a new language in order to use DIME-C. They simply have to learn which parts of ANSI-C cannot be used. Secondly, DIME-C code can be compiled and debugged using a standard C compiler.

Xilinx tools:

The Xilinx tools provide a VHDL and Verilog development environment with a full range of editing, synthesis, simulation and implementation tools. These tools are required regardless of whether programmer actually use the editing,synthesis,simulation parts of the tool suite.

Because all of the tools mentioned above require vendor specific place and route tools in order to place and route designs onto that vendors FPGAs.

Xilinx Spartan VI - SP605 board is used for having FPGA functionalities during the project work. Xilinx provide Xilinx ISE 13.2 Design Suite to operate with SP605 board. This tool set supports HDL, VHDL and Verilog languages.

Since VHDL/Verilog is having more C like syntaxes than HDL, it will be used as programming language for this project. There are three reasons to select Verilog and VHDL as a programming language:

a. It is supported by Xilinx tools available with the Spartan 6 - SP605 board.

b. It is freely available and not proprietary.

c. There is no need of external place and route mechanism is required.

2.2.6 FPGA Programming Process

One can compare various steps of FPGA programming with the corresponding stages of program development in a microprocessor. Though the first level picture of FPGA Programming looks similar to the microprocessor programming, the actual programming process is quite different. Figure 2.15 shows the FPGA programming process in detail.

As depicted in figure 2.15, the output of verilog code compilation is RTL netlist. When input to a synthesizer, the verilog is converted into a gate-level netlist. It is capable of being mapped into FPGA hardware. This gate-level verilog can be compiled and simulated[17]. so we can debug at the actual gate level.

The simulation of the RTL verilog is called functional simulation, while the simulation of the synthesizer verilog output is called gate-level simulation.

In gate-level simulation, synthesizers can optimize FPGA netlists. Area optimization is possible during gate-level simulation which will attempt to use the fewest number of gates (silicon area) on an FPGA at the expense of execution speed.

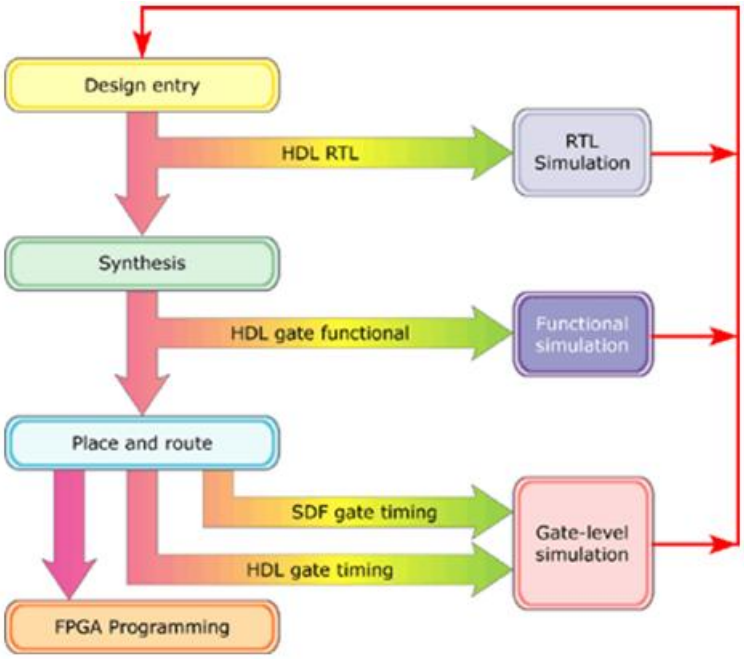

Figure 2.15: FPGA Programming Process[4]

Delay optimization attempts to maximize the execution speed, even if more FPGA area is required.That is why the functional code written in verilog at the RTL level may have different implementations.

Chapter 3

Design Details

This chapter focuses on design details of the project work. With the reference to the figure 2.4 several modules have been identified for the algorithm. Identification of the modules is done on the basis of the computational task divided on each stage of the algorithm.

Below is the list of modules identified.

a. RGB to YUV Conversion

b. Discrete Cosine Transform

c. Watermark Insertion Unit

d. Inverse Discrete Cosine Transform

e. YUV to RGB Conversion

3.1 Design Approach 1

As a very first design approach, xilinx 13.2 Integrate software environment(ISE) was the tool chosen to implement all the above modules. ISE is a GUI provided by Xilinx to develop embedded components.

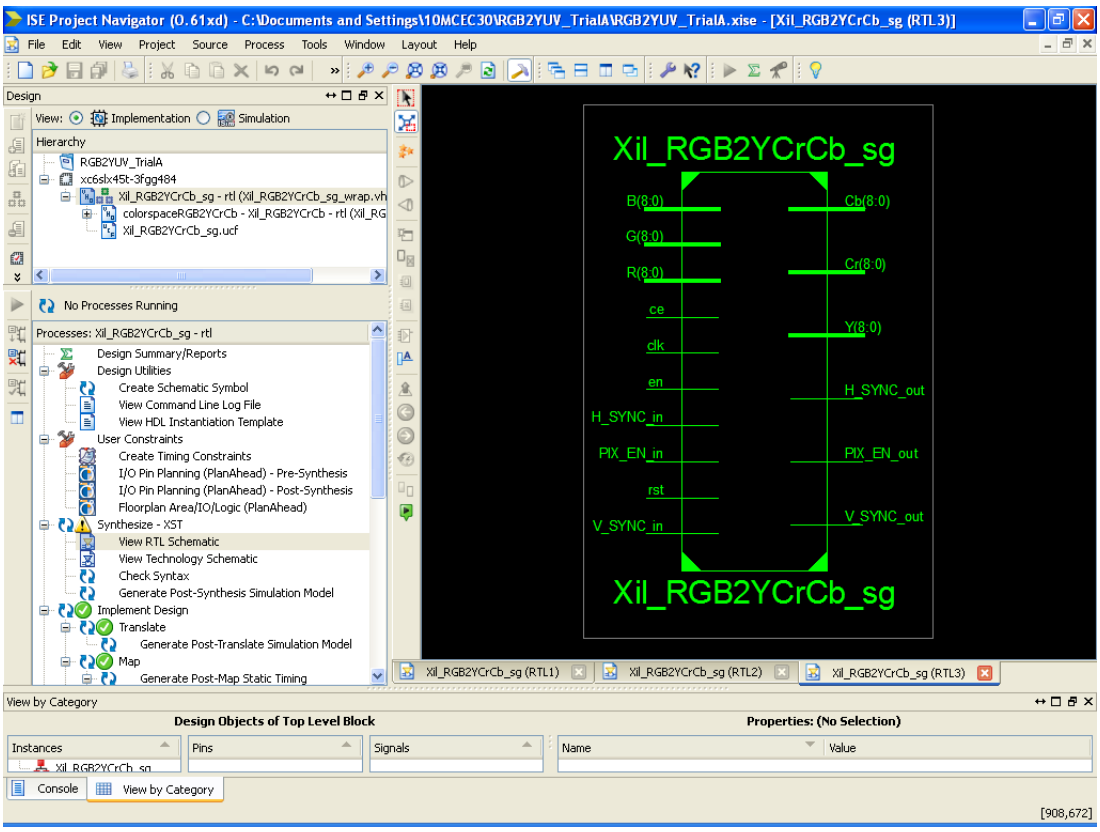

Figure 3.1: RTL View of RGB to YUV Module

The developed module was burnt to the spartan 6 FPGA through iMpact tool that is part of Xilinx ISE. RTL diagram snapshot of RGB to YUV module designed in Xilinx ISE 13.2 is depicted in figure 3.1. Detailed RTL view is shown in figure 3.2.

The RGB to YUV module is being tested on ISim simulator provided with xilinx ISE 13.2 package. ISim is a simulator providing GUI for testing hardware level design developed using xilinx ISE tool. For the design approach 1, the component level diagram of fpga on sp605 board will look like figure 3.3.

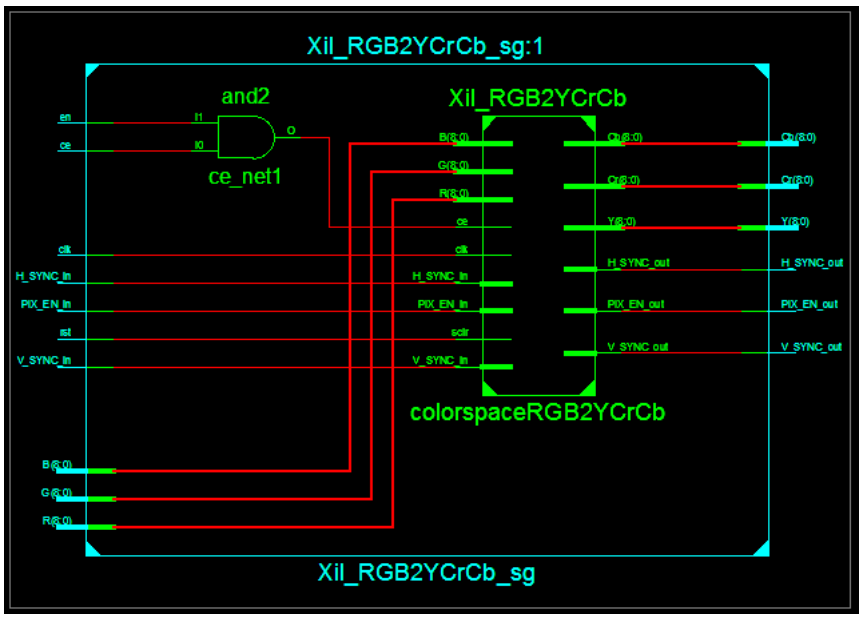

Figure 3.2: Detailed RTL View of RGB to YUV Module

Major challenge in design approach 1:

The major challenge in design approach 1 is memory interface. To access different fpga board memories from modules developed in ISE, there is no programmable or customizable interface from ISE. It required to develop a memory interface in vhdl/verilog language that bridges the gap between ISE and on board fpga memories.

Developing such a memory interface is a time consuming job. It is also not a safe design to maintain various counters for different memories. It also required memory initialization and flushing operations to be designed and implemented in xilinx ISE.

To overcome the gap of memory interfacing in design approach 1, a controller module is needed which can carry out task of memory interface, pc interface and controller for designed ISE modules. To fulfil these requirements, xilinx supported tools were studied to find out a customizable core.

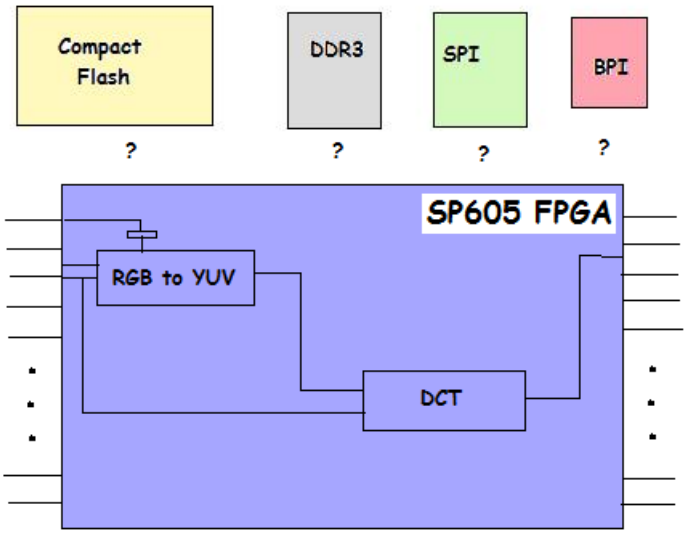

Figure 3.3: Design approach 1

Xilinx provides various customizable cores with the EDK environment those can be utilized by embedded system designer to design their systems. One such customizable processor named Microblaze is being introduced in our design and design approach 2 was adopted for the project work.

3.2 Design Approach 2

Looking to the memory management requirement in design approach 1, we have moved to concept of Microblaze processor which has been depicted as design approach 2. Purpose of module microblaze is to insert a module that act as a controller for memory interface and the modules designed using ISE in design approach 1.

Microblaze is a customizable soft processor core designed for Xilinx FPGAs. It is distributed as a customizable component with the xilinx EDK. It supports library packages to access all the memories available on SP605 board as well as to support interface to external environment like user PC.

Figure 3.4 shows XPS(Xilinx Platform Studio) environment snap. XPS is a GUI environment provided under Xilinx EDK(Embedded Development Kit). This tool allows hardware designer to create highly customized embedded processor systems like microblaze processor and integrate those designs into Xilinx FPGAs.

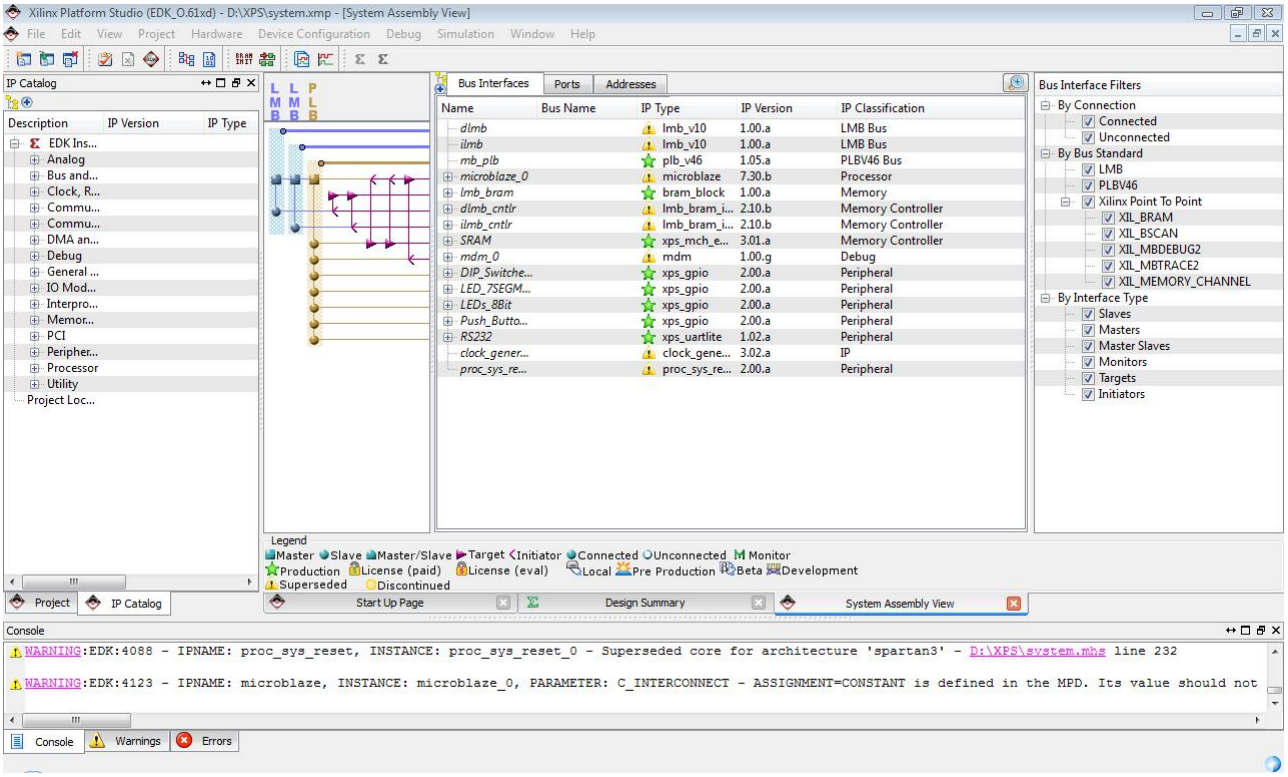

Figure 3.4: XPS design window snap

While creating design of a microblaze soft processor, XPS allows to choose memory and peripherals specific to board. External modules developed in ISE can be imported as peripheral to this soft processor which will be part of a single FPGA chip.

XPS features include[21]:

- Base System Builder allows creation of a fully functional processor system.
- System Assembly View allows user to customize and configure design details.
- IP configuration dialogs open automatically when new IP is added to a design
- Auto bus connectivity on AXI based designs.
- Extensive catalog of AXI and PLB based processors, peripherals, and utility IP
- Create / Import IP wizard automates creation of custom IP templates, and provides mechanism to import user IP into XPS, and Bus Functional Model simulation support for custom IP.
- Hardware project export to the Software Development Kit (SDK)
- Automatic creation of design documentation

After inclusion of microblaze processor on SP605 FPGA chip in design approach 1, our design will get modified as shown in figure 3.5. Once the design of microblaze

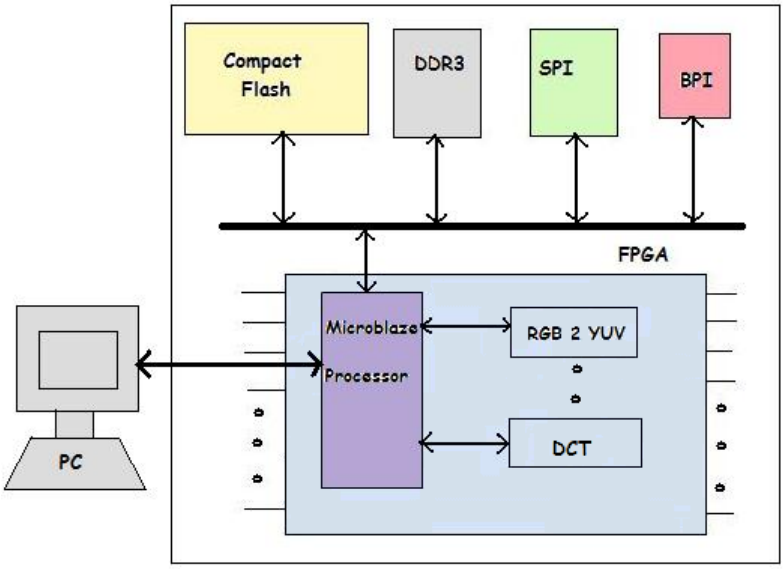

Figure 3.5: Design Approach 2

processor is verified, bitstream is generated in XPS and it is verified with the SP605 board, the microblaze design is exported to Xilinx SDK(Software Development Kit). SDK is Xilinx GUI tool that provides full fledged embedded software development environment for Xilinx embedded processor. Figure 3.6 shows a SDK window snap.

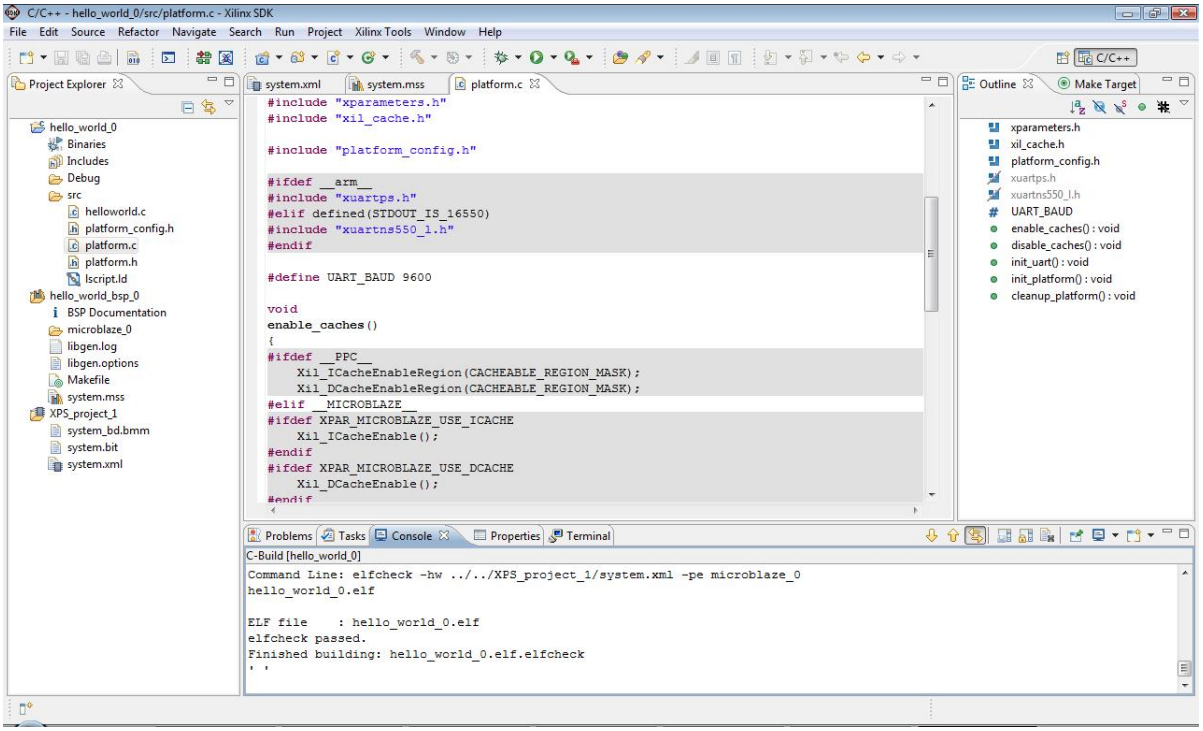

Figure 3.6: SDK Environment Snap

In SDK, the memory interface for the microblaze processor is implemented. For each memory SDK provides library support for various memories supported by Spartan 6 SP605 board. Once the memory interface is developed using SDK, the final design can be deployed to FPGA using 'Program FPGA' feature of SDK.

Chapter 4

Implementation Details

4.1 Implementation Area Identification

FPGA based video watermarking system is divided into several modules as per the computational components involved in the algorithm and hardware implementation while designing the system in chapter 3. In design approach 2, concept of microblaze processor and memory interface was added which required xilinx XPS and SDK tools.

Figure 4.1 shows an overview of all the modules identified and a color scheme notifying where the module will reside on sp605 board. The modules in blue color indicates components to be programmed and reside in FPGA chip - XC6SLX45T-3FGG484 device on SP605 board.

The module in orange indicates memory interface which will utilize on board sp605 memory component. It will reside outside the FPGA chip - XC6SLX45T-3FGG484 and on sp605 board.

The modules in green indicate interface external to the SP605 board. This module will utilize IO components of sp605 board and communicate with external device

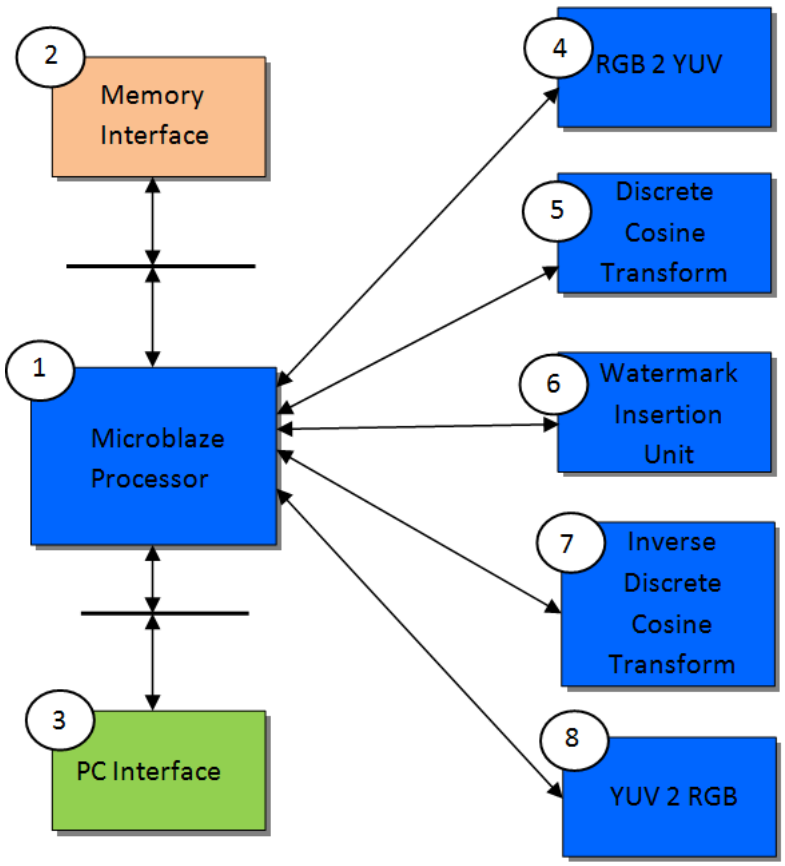

Figure 4.1: Identified modules and color scheme notifying where the module will reside on sp605 board

to sp605 board. It may utilize one of the configuration options mentioned in section 2.2.4.

4.2 Implementation Details of Modules

This section describes implementation details of all the modules identified in this project work.

4.2.1 Microblaze Processor

The MicroBlaze embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs). Figure 4.2 shows a functional block diagram of the MicroBlaze core. The

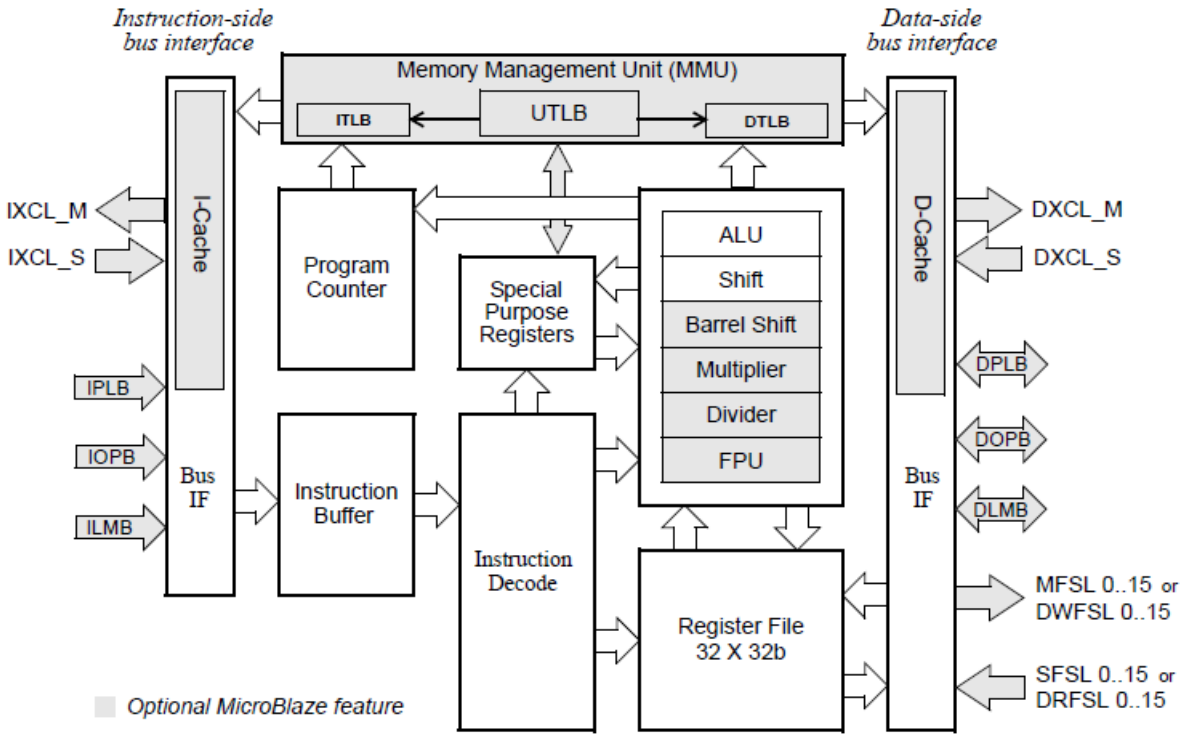

Figure 4.2: Microblaze processor functional diagram[22]

MicroBlaze soft core processor is highly configurable, allowing user to select a specific set of features required by user's design. The fixed feature set of the processor includes[22]:

- Third party 32-bit general purpose registers
- 32-bit instruction word with three operands and two addressing modes
- Separate 32-bit instruction and data buses with direct connection to on-chip block RAM through a LMB(Local Memory Bus).

- 32-bit address bus.
- Sigle issue pipeline
- Hardware debufg logic
- Instruction and Data cache
- FSL (Fast Simplex Link) support

In addition to these fixed features, the MicroBlaze processor is parameterized to allow selective enabling of additional functionality.

4.2.2 Memory interface

Xilinx EDK supplies libraries and board support packages, in addition to the drivers for the peripherals, to help the user develop a software platform. The following is the distribution of the software packages available for the user to include in his platform[23]:

- Standard C Libraries (libc, libm).
- Standalone Board Support Package (BSP).
- Xilkernel - An embedded Kernel.
- LibXil MFS - A Memory File System.
- LibXil FATFS - A FAT file system.
- LibXil Drivers - Device drivers for supported peripherals.
- LibXil Flash - A library that provides read/write/erase/lock/unlock and device specific functionalities for parallel flash devices.
- LibXil Isf - In-System-Flash library that supports the Xilinx In System Flash hardware.

In the current implementation, LibXil Flash is being used to have memory interface with SPI on board memory. The XilFlash library provides read/write/erase/lock/unlock features to access a parallel flash device. Flash device family specific functionalities are also supported by the library. This library requires the underlying hardware platform to contain 'xps_mch_emc' or similar core for accessing the flash.

This library implements the functionality for flash memory devices that confirm to the Common Flash Interface(CFI) standard. CFI allows a single flash library to be used for an entire family of parts. This library supports Intel and AMD CFI compliant flash memory devices.

4.2.3 PC Interface

Deployment of the final design to the SP605 board requires support of programming FPGA. This feature is supported by Xilinx SDK. Using this functionality the .bit, .elf files are first programmed to FPGA. Then the video and image data are being burnt in the flash memory using "Program Flash" option of xilinx SDK tool.

Once these two parts are ready, the program containing algorithm is being run on hardware. All these process uses JTAG connection of configuring FPGA.

4.2.4 RGB to YUV Conversion

RGB color model is used in computer graphics, YUV model is used in video systems. Transferring color information from one industry to another requires transformation from one set of values to another.

Once RGB values of frames are converted to YUV values, luminance component of an image can be processed without affecting color component. This is the main reason of YUV conversion in this project work.

Figure 4.3 is the detailed RTL diagram of implemented RGB to YUV conversion module.

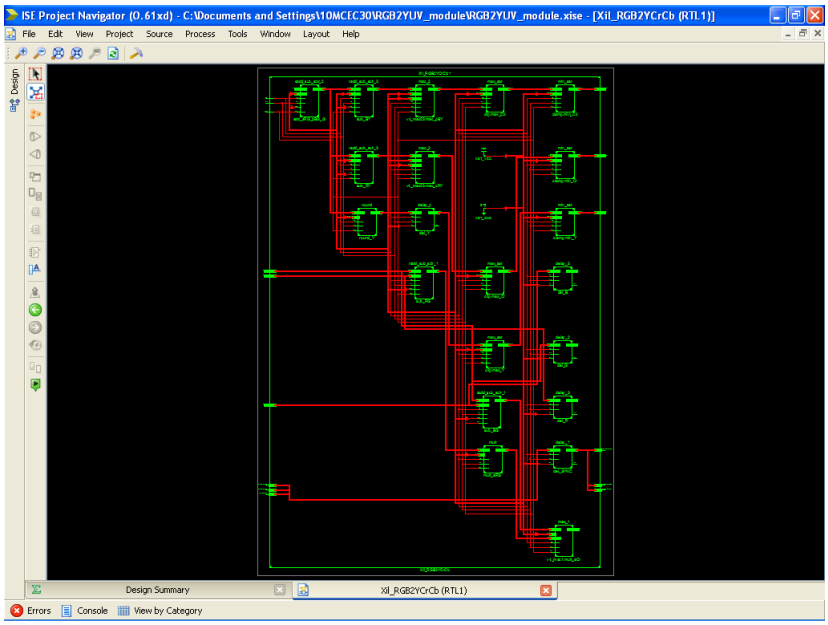

Figure 4.3: Detailed RTL diagram of RGB to YUV conversion module

4.2.5 Discrete Cosine Transform Module

Discrete cosine transform transforms a signal or image from the spatial domain to the frequency domain. To implement 2D DCT in VHDL, the problem is factorized to 1D DCT first. 1D DCT is applied to columns first and then again applied DCT to resultant coefficients.

Figure 4.4 is the detailed RTL diagram of DCT module implementation.

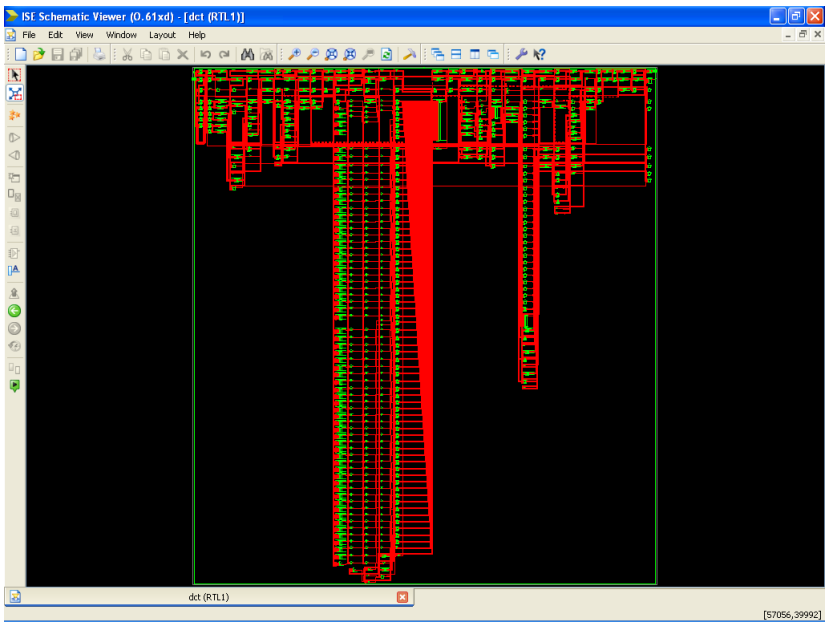

Figure 4.4: Detailed RTL diagram of discrete cosine transform module

4.2.6 Watermark Insertion Unit

For each 8x8 block 2D DCT coefficient will be found. According to the pixel values, 90% of the information will be taken from frame's coefficient and 10% of the value will be considered from watermark coefficient and added together to get watermarked video coefficient. Figure 4.5 shows Watermark insertion unit detailed RTL diagram.

4.2.7 Inverse Discrete Cosine Transform Module

The watermarked DCT coefficient is generated as output of watermark insertion unit. This DCT coefficient is watermarked coefficient which is used as input to IDCT module. Figure 4.6 is the detailed RTL diagram of IDCT module.

4.2.8 YUV to RGB Conversion

Finally, the resultant YUV frames are converted to RGB format to get the output watermarked video. Figure 4.7 depicts detailed RTL view of YUV to RGB module.

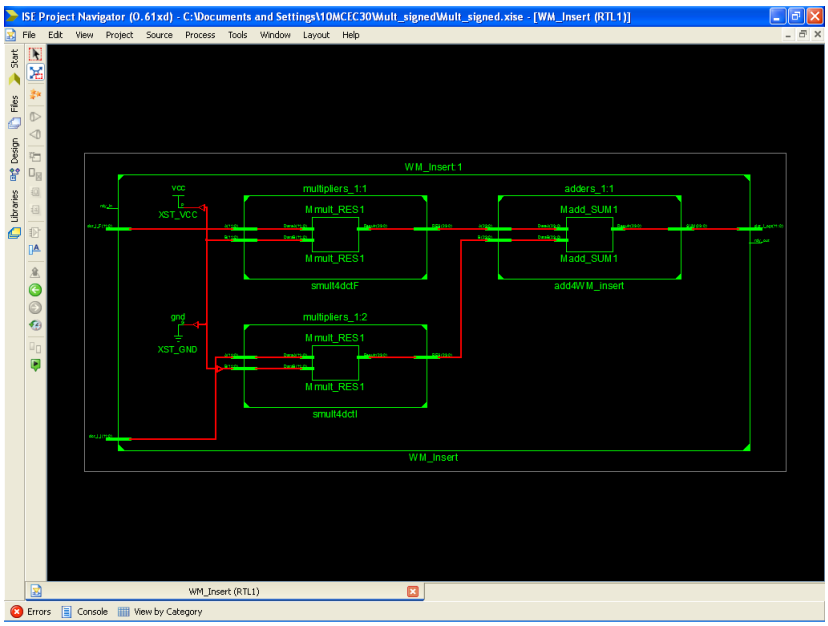

Figure 4.5: Detailed RTL diagram of Watermark Insertion module

4.3 Integration of ISE Modules

In xilinx XPS tool, it is possible to integrate a module developed in ISE as a peripheral to microblaze processor. Instead of adding each module in XPS, all the modules are being integrated in xilinx ISE and the integrated system is to be integrated in XPS. Figure 4.8 and 4.9 is the RTL level diagram of integrated modules in xilinx ISE.

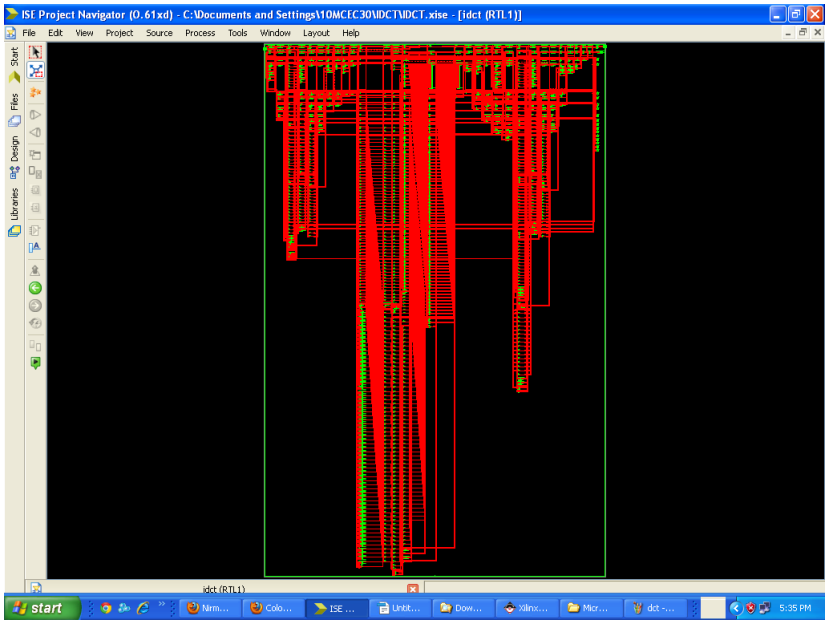

Figure 4.6: Detailed RTL diagram of inverse discrete cosine transform module

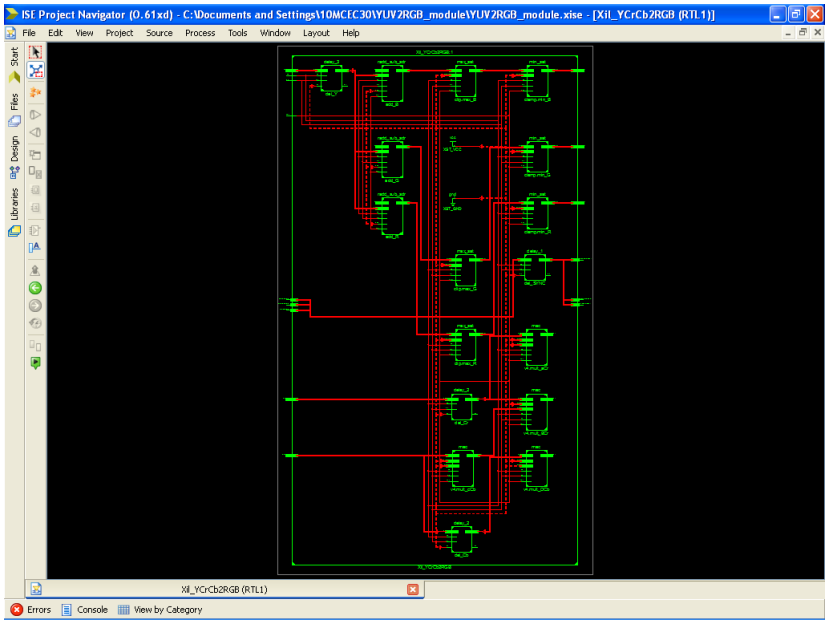

Figure 4.7: Detailed RTL diagram of YUV to RGB conversion module

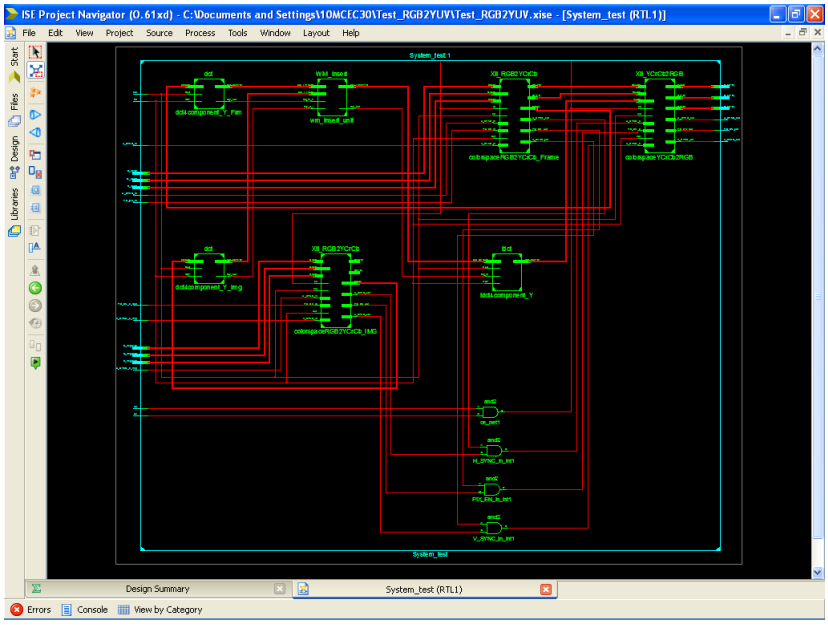

Figure 4.8: Unexpanded RTL diagram of integrated system

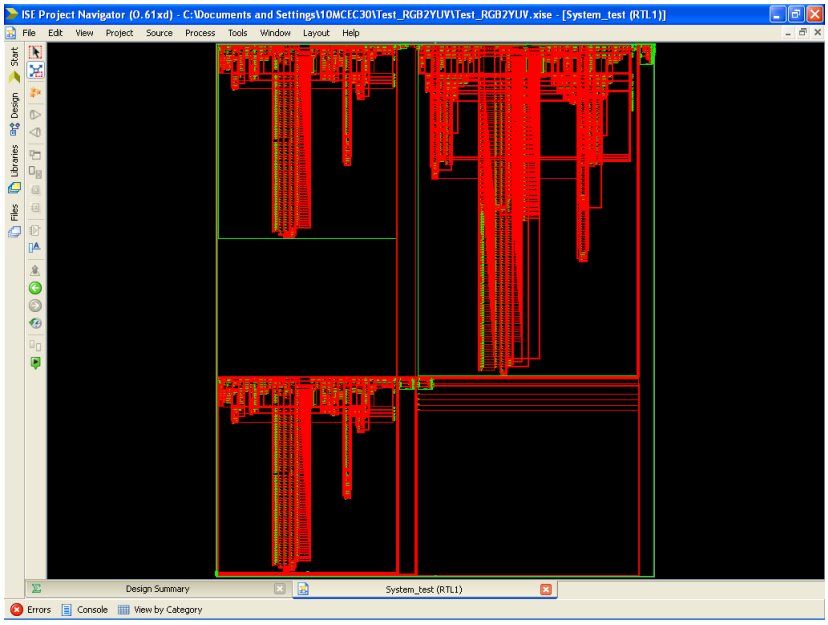

Figure 4.9: Expanded RTL diagram of integrated system

Chapter 5

Results and Findings

This section includes implementation results of the project work. Resource utilization is measured for each modules with the help of design summary utility provided by xilinx Embedded Development Kit(EDK) .

5.1 Resource Utilization Summary

Below is the resource utilization summary and time consumption of each modules.

Modules	**No. of LUTs used (%of total LUTs)**	**Number of registers used (%of total registers)**
RGB2YUV	132(1%)	174(1%)
DCT	1,073(3%)	1,517(2%)
IDCT	1,732(6%)	2,444(4%)
YUV2RGB	73(1%)	120(1%)
ISE Integration	4,106(15%)	5,506(10%)
Microblaze	9,027(33%)	8,110(14%)

Table II: Resource Utilization Table

As per the above resource utilization summary, total 51.27% LUTs of total LUTs in XC6SLX45T-3FGG484 FPGA device is being utilized during this implementation.

With the help of design summary utility and isim modulator below worst slack time is noted for each module.

Modules	**Worst time slack (in neno second)**
RGB2YUV	0.315
DCT	920
IDCT	1020
YUV2RGB	0.413
Total time Slack	1940.728

Table III: Worst time slack summary of modules implemented

Depending upon above results and the fact that the operating frequency of implemented system is set to 100MHz the overall performance for various MPEG standards is compared as shown in table IV.

	MPEG-1	**MPEG-2**	**MPEG-4**
Maximum Video Resolution	352x288	1920x1152	720x576
Frames Processed per Second	325	15	79

Table IV: Performance comparison for various MPEG standards

The project work concentrates on real-time needs of video watermarking. MPEG-4 being the common standard for real-time videos, 79 frames per second is a good performance achievement.

The system processes 15 frames per second for high quality videos of MPEG-2 standards. This can be improved to a better level by increasing operating frequency up

to 150MHz of current system.

There are research efforts available in same direction by various researchers using FPGA technology and other supporting techniques to increase the performance. Comparison with the research works is depicted in the next section.

5.2 Comparison with other research works

1 Comparison with paper reference [20]

Referred Paper Data:

a. Video standard: MPEG-4

b. Resolution Selected: 320x240

c. Performance Obtained: 43 frames per second

Our Experiment Results:

a. Video standard: MPEG-4

b. Resolution Selected: 320x240

c. Performance Obtained: 43 frames per second(Calculated with worst time slack)

2 Comparison with paper reference [14]

Referred Paper Data:

a. Video standard: MPEG-2(Watermark logo insertion)

b. Resolution Selected: 1920x1080

c. Performance Obtained: 30 frames per second

Our Experiment Results:

a. Video standard: MPEG-2(Full frame luminance watermarking)

b. Resolution Selected: 1920x1080

c. Performance Obtained: 16 frames per second(Calculated with worst time slack)

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The conclusion statements are as below:

a. The goal of hardware assisted video watermarking is to achieve low power usage, real-time performance, reliability, and ease of integration with existing consumer electronic devices.

b. FPGA technology is a viable solution for video watermarking algorithms. This has been proven with taking example of frequency domain watermarking algorithm.

c. Verification of logic and verification of design in FPGA is time taking processes. To figure out boundary conditions and test cases is again complex and time taking jobs.

d. To implement complex solutions, first designer needs to break down the problem into simpler forms. Again the way to decompose the design often effects performance of the system.

e. FPGAs helps in achieving better performance by bypassing the fetch-decode-execute overhead of general purpose processors.

6.2 Future Scopes

1 Optimization scope through higher frequency:

All the experiments during this project work is carried out using system clock frequency 100MHz. This frequency can be increased till 150MHz which needs timing and connection optimizations in the various circuits. One may try for these optimization to achieve better performance.

2 Optimization through handling memory:

Spartan 6 SP605 board supports various memory types. All the memories are having speed vs. storage specifications. There is a good scope to effectively utilize all these memories to optimize time benefits.

3 Optimization through parallelization of design:

Further improvements can be achieved by identifying parallel execution paths and implement the same in FPGA

Chapter 7

Appendix A: Publications of project work

A part of this project work is being presented at *'National Conference in Emerging Vitas of Technology in* 21^{st} *Century'* and has been published under special issue *'Revolutionary Trends in IT(ISSN 0971-3034)'* of Indian Journal of Technical Education(IJTE) promoted by Indian Society for Technical Education(ISTE).

Chapter 8

References

1 FPGA-Based System Design, By Wayne Wolf, 2005.

2 http://www.xilinx.com/company/gettingstarted.

3 Stephen Brown, Jonathan Rose. "Architecture of FPGAs and CPLDs: A Tutorial". University of Toronto,IEEE Design & Test of Computers, July 1996.

4 Ed Klingman. " FPGA programming step by step", Embedded System Programming (Mar 2004), ArticleID-18201956.

5 Richard Wain, Ian Bush, Martyn Guest, Miles Deegan, Igor Kozin and Christine Kitchen. " An overview of FPGAs and FPGA programming; Initial experiences at Daresbury ", CCLRC Daresbury Laboratory, Warrington, Cheshire, WA4 4AD, UK, November 2006, Version 2.0.

6 Xilinx Online Document Support, "Getting Started with the Xilinx Spartan-6 FPGA SP605 Evaluation Kit", UG525 (v1.1) June 7, 2010.

7 J. Bannur and A. Varma. A VLSI implementation of a square root algorithm, In IEEE Symposium on Comp. Arithmetic, pages 159-165. IEEE Computer Society Press, Washington D.C., 1985.

8 Daniel Ziener and Jurgen Teich, Evaluation of Watermarking methods for FPGA-based IP-cores, Hardware/Software Co-Design, University of Erlangen-Nuremberg, Germany, Co-Design-Report 01-2005, March 1, 2005, http://www12.informatik.uni-erlangen.de

9 Daniel Ziener, Moritz chmid,Jurgen Teich. 'Robustness Analysis of Watermark Verification Techniques for FPGA Netlist Cores', Published in: "Design Methodologies for Secure Embedded Systems", LNEE 78, pp.105-127, Springer-Verlag Berlin Heidelberg, 2010

10 John Lach, William H. Mangione-Smith, Miodrag Potkonjak. 'Robust FPGA Intellectual Property Protection Through Multiple Small Watermarks', UCLA EE Department, Los Angeles, CA 90095, 310-794-1630, International Conference on Multimedia Computing and Systems, 1996.

11 Adarsh K. Jain, Lin Yuan, Pushkin R. Pari, Gang Qu. 'Zero Overhead Watermarking Technique for FPGA Designs', University of Maryland, College Park, GLSVLSI03, April 28-29, 2003, Washington, DC, USA

12 Dr. M. Madhavi Latha, G. Kesavan pillai, K. Anitha Sheela, 'Watermarking based Content Security and Multimedia Indexing in Digital Libraries', JNT University, Hyderabad, Andhra Pradesh, India, International conference on Enabling Technologies for Smart Appliances(ETSA) IEEE, Hyderabad.Jan 2005.

13 Eugene T. Lin, Christine I. Podilchuk, Ton Kalker, Edward J. Delp.'Streaming video and rate scalable compression: what are the challenges for watermarking?', Video and Image Processing Laboratory, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, Philips Research, NL-5656-AA, Eindhoven, The Netherlands, June 2000.

14 In-Koo Kang, Dong-Hyuck Im, Heung-Kyu Lee, Young-Ho Suh. 'Implementation of Real-Time Watermarking Scheme for High-Quality Video', Korea Ad-

vanced Institute of Science and Technology Guseong-dong, Yuseong-GuDeajeon, Republic of Korea, MM&Sec06, September 2627, 2006, Geneva, Switzerland.

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

16 V Muthukumar, Daggu Venkateshwar Rao. 'Image Processing Algorithms on Reconfigurable Architecture using HandelC', University of Nevada Las Vegas, Las Vegas, NV 89154. Proceedings of the EUROMICRO Systems on Digital System Design (DSD04), 0-7695-2203-3/04.

17 FPGA programming step by step, By Ed Klingman, Courtesy of Embedded Systems Programming, Mar 4 2004 (14:00 PM), URL: http://www.design-reuse.com/exit/?url=http://www.embedded.com/showArticle.jhtml.

18 Deepak Kumar Tala. 'Verilog Tutorial', 25-Oct-2003, Website : http://www.deeps.org.

19 Saraju P. Mohanty, Elias Kougianos. 'Real-time perceptual watermarking architectures for video broadcasting',NanoSystem Design Laboratory (NSDL), University of North Texas, Denton, TX 76207, USA. The Journal of Systems and Software 84 (2011) 724738.

20 Xilinx Online Document Support, "Spartan 6 FPGA Configuration", UG380 (v2.3) July 6, 2011

21 http://www.xilinx.com/tools/xps.htm

22 ug081(v9.0)-Microblaze Processor Reference Guide, EDK 10.1i

23 OS and Libraries Document Collection, EDK 10.1, Service Pack 3,September 19, 2008