FAST INTRA/INTER MODE DECISION FOR A REAL-TIME H.264 STREAMING SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜ ALAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Gözde Bozdağı Akar
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Aydın Alatan (METU,EE) _____

Assoc. Prof. Dr. Gözde Bozdağı Akar (METU,EE) _____

Assoc. Prof. Dr. Temel Engin Tuncer (METU,EE) _____

Assist. Prof. Dr. Çağatay Candan (METU,EE) _____

Ertuğrul Kolağasıoğlu(M.S.) (OPTISIS INC.) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özgü Alay

Signature :

ABSTRACT

FAST INTRA/INTER MODE DECISION FOR H.264 CODED VIDEO STREAMING SYSTEM

Alay, Özgü
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Gözde Bozdağı Akar

July 2006, 88 pages

Video compression is a key technology used in several multimedia applications. Improvements in the compression techniques together with the increasing speed and optimized architecture of the new family processors enable us to use this technology more in real time systems. H.264 (also known as MPEG-4 Part 10 or AVC - Advanced Video Coding), is the latest video coding standard which is noted for achieving very high data compression. While H.264 is superior to its predecessors, it has a very high computational complexity which makes its costly for real time applications. Thus, in order to perform video encoding with satisfactory speed there is an obvious need for reducing the computational complexity. New algorithms were developed for this purpose. The developed algorithms were implemented on Texas Instrument TMS320C64x family to be able to fulfill the requirement in optimized signal processing hardware with low power consumption which arises from the computational complexity and the need for portable devices in video processing technology. With the new algorithms developed, a computation reduction of 55% was achieved without loosing perceptual image quality. Furthermore, the algorithms were implemented

on a DSP along with the networking functionality to obtain a video streaming system. The final system may be used in a wide range of fields from surveillance systems to mobile systems.

Keywords: H.264, Video Compression, Mode Selection, Real Time Coding, Digital Signal Processor

ÖZ

GERÇEK ZAMANLI H.264 DURAKSIZ VİDEO İLETİMİ İÇİN HIZLI İÇSEL/ÇERÇEVELER ARASI KİP SEÇİMİ

Alay, Özgü
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Gözde Bozdağı Akar

Temmuz 2006, 88 sayfa

Video sıkıştırma birçok çoklu ortam uygulamalarında kullanılan kilit bir teknolojidir. Sıkıştırma tekniklerinin gelişmesi bunun yanısıra hızlandırılmış ve geliştirilmiş yeni işlemci aile yapıları, bu teknolojinin gerçek zamanlı sistemlerde daha çok kullanımını mümkün kılmıştır. H.264 (MPEG-4 Bölüm 10 veya AVC – Gelişmiş Video Kodlama olarak da bilinir), çok yüksek veri sıkıştırma değerlerine ulaşmış, bilinen en son video kodlama standardıdır. H.264 öncel çalışmalara birçok yönden üstünlük kurarken, çok yüksek hesaplama karmaşıklığı bu ürünün gerçek zamanlı uygulamalarda kullanılmasını maliyetli kılmamaktadır. Dolayısıyla, tatmin edici hızlarda video kodlamak için hesaplama maliyetinin azaltılmasına olan ihtiyaç açıktır. Bu amaçla yeni algoritmalar geliştirilmiştir. Pratikte kullanabilmek için geliştirilen algoritmanın düşük güç harcayan, taşınabilir ve kişisel bilgisayarlardan daha güçlü bir donanıma aktarılması gerekmektedir. Bu amaçla geliştirilen algoritma Texas Instrument firmasının TMS320C64x ailesine uygulanmıştır. Geliştirilen algoritma ile, algılanan imge kalitesi korunarak, hesaplama süresinde %55 e varan bir düşüş sağlanmıştır. Buna ilaveten bu algoritmalar SSİ üzerine uygulanmıştır ve ağsallık özelliği ile

birlikte duraksız video iletim sistemi oluşturulmuştur. Oluşturulan bu sistem güvenlik sistemlerinden taşınabilir iletişim sistemlerine kadar bir çok alanda kullanılabilir.

Anahtar Kelimeler: H.264, Görüntü Sıkıştırma, Kip Seçimi, Gerçek Zamanlı Kodlama, Sayısal Sinyal İşlemci.

To my parents,
who let me go my own way while providing all the support and help
whenever I needed

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor Assoc. Prof. Dr. Gözde Bozdağı Akar for her guidance, criticism, support, encouragement, insight and friendship throughout the research.

This thesis work is supported by Turkish Prime Ministry State Planning Organization (DPT) under "Unmanned Aerial Vehicles" project. I would like to acknowledge the project coordinator Prof. Dr. Nafiz Alemdaroğlu for his support and understanding throughout the project.

Thanks go to Multimedia Research Group (MMRG) members for their technical support. Besides, I spent great time in all our activities in the last couple of years.

I would like to thank my dear friend Özlem Pasin. It is a great feeling to know that somebody cares about you and will be by your side in every situation.

Finally, Dr. Emrah Erduran. Despite thousands of miles in between, your presence has given me a lot of courage and strength throughout the most stressful periods of this thesis. When I need help you were the first one to ask for, when I am exhausted and upset you were the first one to make me smile. Thank you for your invaluable love and peace.

My deepest thanks and love go to my family for their endless love, support and encouragement throughout my whole life

Nothing in life is impossible, so go for it and have no regrets…

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Program Interface
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CCS	Code Composer Studio
CSL	chip support library
DCT	Discrete Cosine Transform
DHCP	Dynamic Host Configuration Protocol
DS	Diamond Search
DSP	Digital Signal Processors
DVD	Digital Video Disc
EDMA	Enhanced Direct Memory Access
EMIF	External Memory Interfaces
FS	Full Search
FVID	frame video module
GPIO	general-purpose input/output
HAL	hardware abstraction layer
HDTV	High Definition TV
ISDN	Integrated Services Digital Networks
JVT	Joint Video Team
LAN	Local Area Network
MAC	Multiply and Accumulate
MAD	mean of absolute differences
ME	Motion Estimation
MIPS	Million Instructions per Second

MPEG	Motion Picture Experts Group
MVD	motion vector difference
NAL	Network Abstraction Layer
NDK	Network Developer's Kit
OpenCV	Open Source Computer Vision
PSTN	Public Switching Telephone Networks
RDO	Rate Distortion Optimization
RF5	Reference Framework 5
RTDX	Real-Time Data Exchange
SAD	sum of absolute difference
SSD	sum of square differences
TDM	time–division multiplexed
TI	Texas Instrument
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VLC	Variable Length Coding
VLIW	very long instruction word

CHAPTER 1

INTRODUCTION

1.1 General

Innovations in communication systems resulted in real time transmission of high quality multimedia. Nevertheless, the available bandwidths do not allow the transmission of uncompressed multimedia data. Hence, researchers have focused on developing compression techniques to enable real-time transmission of visual services such as, Video Conferencing, Videophone, Video E-mail, and Video Streaming over Internet, High Definition TV (HDTV), and Digital Video Disc (DVD).

ITU-T and ISO/IEC are the two institutes that contributed to the developments in the video compression techniques. In 1990, ITU-T introduced the video compression standard H.261. In order to avoid the limitations of H.261, H.263 was issued in 1995 which was further improved and released as H.263+ in 1998. Meanwhile, the researchers in ISO/IEC have introduced the first version of the well known video compression format, MPEG1 in 1991. In 1994, MPEG-1 was improved to become widely used MPEG2 which is also utilized in DVD's. Upon these developments, these two institutes initiated a new project to combine the strengths of these two families of standards. The first draft of the final product of this joint research, H.264 was released in 2003.

The advantages of the H.264 standard over the predecessors are, but not limited to, compression and rate distortion efficiency along with network friendliness. However, one of the major drawbacks of H.264 is its computational complexity which makes it costly for real-time applications. Some enhancements in the standard are necessary to overcome this drawback. Moreover, this computational complexity results in extensive computational capability requirements. Even a PC with high speed CPU may not be sufficient to handle these requirements in real-time. A smaller, portable optimized signal processing hardware with low power consumption will be more preferable for the consumers. Digital Signal Processors (DSP) seems to be a perfect fit for this purpose.

Recently, H.264 based boards are released in the market and since these boards are of great interest new boards will be released in near future. Elecard [1], WWCom [2] and Ateme [3] are the leading companies that conduct research on H.264 implementation on DSP where TI's TMS320C6000 family and H.264 baseline profile are utilized in most of the products. Most of these products require non-disclosure agreement (NDA) and H.264 encoders in them are not open source which prevent them to be modified to be able to keep up with the advancements in H.264 standard. There is an open source H.264 encoder released by Texas Instruments which costs almost 60000$.

1.2 Scope of the thesis

One of the two main objectives of this study is to develop algorithms that will reduce the computational complexity while maintaining similar PSNR and bit-rate values compared to reference software [4]. Upon achieving this objective, further studies were carried out to implement this enhanced software on Texas Instrument's (TI's) TMS320C6000 DSP platform in order to have a portable, low power, real-time video streaming system. One of the most critical aspects of this implementation is that the commercially available softwares that encode H.264 are not open source which prevent them to be modified to be able to keep up with the advancements in H.264 standard.

As being supported by Turkish Prime Ministry State Planning Organization (DPT) under "Unmanned Aerial Vehicles (UAVs)" project, the system in this thesis is planned to be used in UAVs. Since H.264 standard is planned to be added into the next edition of STANAG 4609 (NATO's Digital Motion Imagery Format), the implemented sytem in this thesis will be compatible with the recent military standards.

1.3 Outline of the dissertation

The structure of this thesis closely follows the order in which the work was undertaken in response to the aims as they were initially conceived. It consists of five further chapters.

Chapter 2 briefly summarizes the history and general structure of the H.264 video compression standard. Main building blocks that are of interest to the scope of the study are explained in detail.

Third chapter focuses on algorithms developed to overcome the computational complexity drawback of the H.264 standard. Moreover, the results of the proposed algorithm were also presented in this chapter.

Chapter 4 summarizes implementation details of the enhanced software on the TI's TMS320C6000 DSP platform and system features.

Fifth chapter discusses the results obtained from the use of proposed algorithms implemented on DSP platform on sample and real-time video sequences.

Finally, Chapter 6 summarizes the work done within the scope of this thesis and discusses the conclusions drawn from the work carried out. It also addresses the recommendations for the similar works that are intended to be done in the future.

CHAPTER 2

H.264 RECOMMENDATION

2.1 Overview

The growing interest in digital video applications has led researchers to focus on developing compression techniques to meet the requirements of various applications. ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG) are two organizations that develop video coding standards. MPEG-1 and MPEG-2 are standards which are developed for coding video and audio by ISO/IEC. ITU-T developed the first widely used video telephony standard, H.261 and its successor, H.263. A joint project was initiated to combine these two families of standards into an international standard, H.264, by both ISO/IEC and ITU-T.

MPEG-1 [5] is the first MPEG standard which is developed for video storage on CD-ROM's in 1991. Some of the features in MPEG-1 video are: block-based motion compensation, Discrete Cosine Transform (DCT) and the quantization. It is optimized for bit rates of about 1.5 Mbits/s.

To improve the video quality of MPEG-1, MPEG-2 [6] was developed in 1994. It is based on the MPEG-1 with some additional features such as: efficient coding of interlaced videos, new prediction modes for I frame, 16x8 block size motion compensation, improved Variable Length Coding (VLC) tables and different

scalability modes. Thus the final MPEG-2 is a fully generic system for audiovisual interactive services. MPEG-2 has a wide range of applications, from low-resolution to high-resolution, from low picture quality to high picture quality and from low bit-rate to high bit-rate. These scalability features allow it to be used in cable TV, DVD video and high definition TV.

The ITU-T's H-261 [7] standard was developed in 1990 to support video telephony and video conferencing over Integrated Services Digital Networks (ISDN). These networks operate at multiple of 64 Kbps ant the standard supports bit rates between 64 kbps to 2 Mbps. The standard has the following key features such as: integer pixel accuracy for inter prediction, differentially encoded motion vectors, adaptive loop filter, 8x8 DCT for residual coding and run-level coding.

To enhance the compression performance of H.261 for Public Switching Telephone Networks (PSTN), H.263 [8] was developed in 1995. These networks operate at bit rates about 20 kbps. Comparing to H.261, H.263 offers half bit rate for similar picture quality. The standard has the following additional features such as: half pixel accuracy motion compensation, different arithmetic and variable length coding and advanced prediction modes.

H.264 [9] is the newest video coding standard which is developed by the Joint Video Team (JVT) whose members are ITU-T and ISO/IEC. H.264 Recommendation has a number of features that provide significant improvement in terms of compression and rate-distortion efficiency when compared with the other standards. Furthermore a network friendly representation is also aimed in H.264 through a conceptual separation between a Video Coding Layer (VCL) which generates the high-compressed video stream and Network Abstraction Layer (NAL) which formats the VCL representation of the video and provides header information appropriate for transmission and storage.

2.2 H.264 Encoder Structure

In Figure 2-1[10] the block diagram of H.264 is given. Each input frame is split into macro blocks and each macro block is coded in intra or inter mode. Intra and inter coding produces a prediction block which is subtracted from the original macro block. This difference macro block is called the residual macro block. The residual block is transformed, quantized and entropy coded together with side information and finally the compressed bit stream, which is passed to NAL for transmission or storage, is formed.

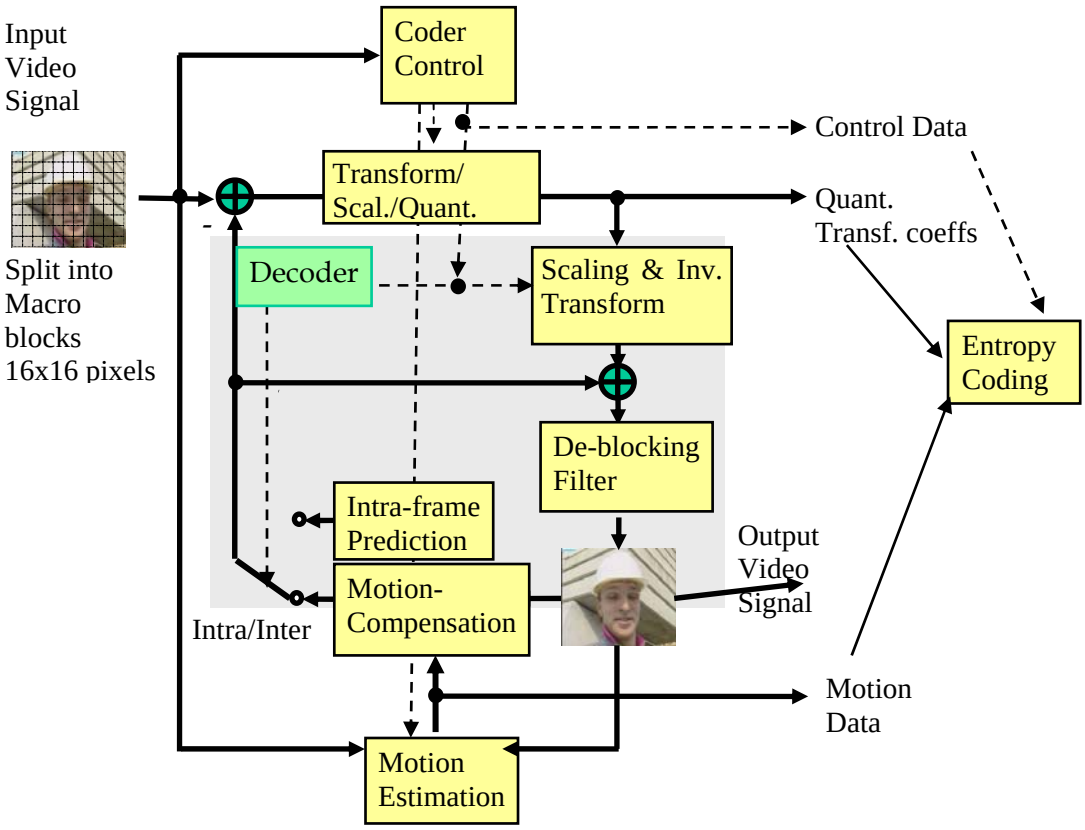

Figure 2-1 Encoder Structure

2.3 Video Format and Structure

2.3.1 Video Format

Progressive or interlaced video with default 4:2:0 sampling format is supported in H.264. In the default sampling format, which is shown in Figure 2-2 [11], chrominance (Cb and Cr) samples are aligned horizontally with every second luminance sample and are located vertically between two luminance samples.

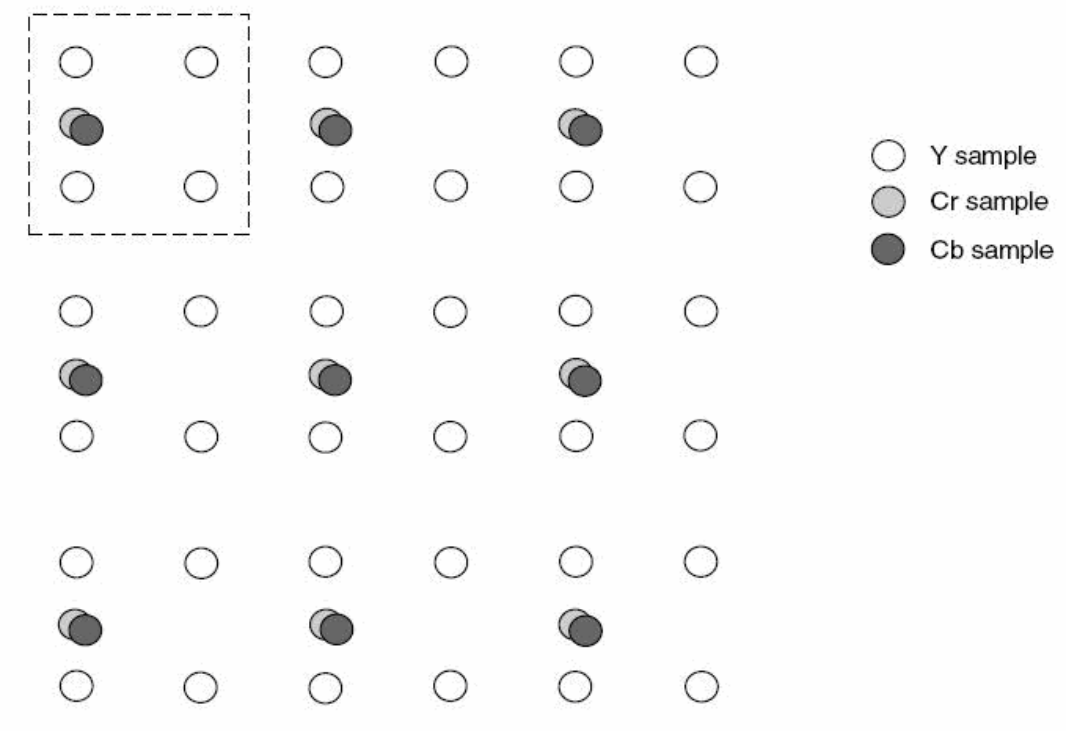

Figure 2-2 4:2:0 Sampling Pattern

2.3.2 Video Frame Structure

A frame consists of a number of slices each containing an integral number of macro blocks from 1 (1 MB per slice) to the total number of macro blocks in a frame (1 slice per frame). Example QCIF in Figure 2-3 has two slices where each

slice is represented by two different colors. Each slice consists of regions called macro blocks. For luminance, each macro block consists of 16x16 luminance pixels whereas for chrominance each macro block consists of two 8x8 chrominance pixels.

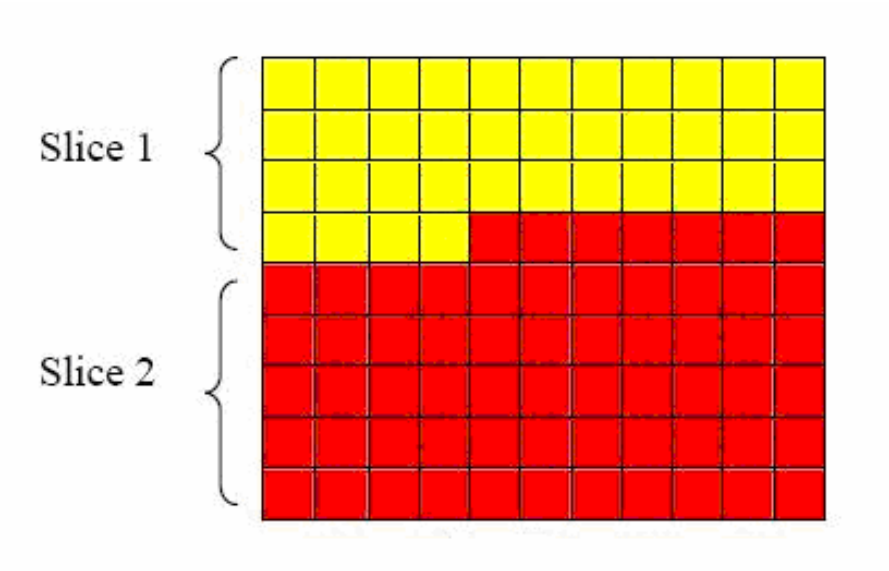

Figure 2-3 QCIF Frame with two slices

There are five types of slices in H.264 such as: I Slice (Intra), P Slice (Predicted), B Slice (Bi -predictive), SP Slice (Switching P), SI Slice (Switching I) and a frame can contain a mixture of these types. The main idea with slices is that there is minimal inter-dependency between coded slices which can help to limit the propagation of errors.

2.4 Intra Coding

In H.264, intra coding is conducted in spatial domain where previously coded left and/or top blocks are used for intra prediction [11]. A low energy residual is formed by subtracting the prediction from the original macro block. Intra4x4 and intra16x16 together with chrominance prediction are different kinds of intra

coding. Intra4x4 prediction is used to encode parts of the picture with significant details whereas intra16x16 is used to encode smooth areas of the picture. In the intra4x4 prediction, each 4x4 block is predicted from spatially neighboring samples as illustrated on the left-hand side of Figure 2-4. There are 8 directional prediction modes and a DC prediction mode as shown on the right-hand side of Figure 2-4. In the intra16x16 prediction, there are 4 modes which are shown in Figure 2-5. Each 8x8 chrominance block has four types of prediction modes which are very similar to the 16x16 luminance prediction modes.

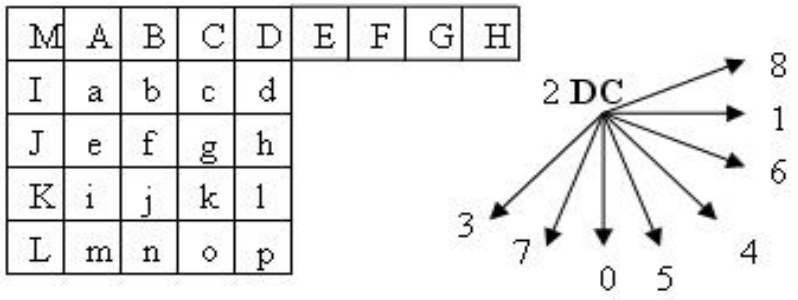

Figure 2-4 Left: Using samples A-M, intra4x4 prediction is conducted for 4x4 block. Right: Eight prediction modes for intra4x4 prediction.

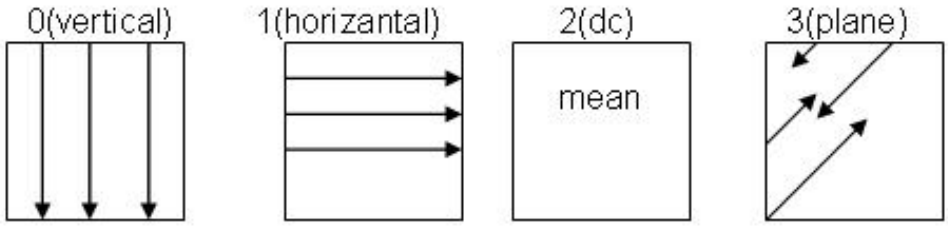

Figure 2-5 Four prediction modes for intra16x16 prediction

2.5 Inter coding

In video coding systems, there is a great amount of temporal correlation between adjacent frames. These temporal redundancies should be reduced to achieve higher coding efficiency. That is, a frame is selected as a reference, and subsequent frames are predicted from the reference using a technique known as motion estimation. The process of video compression using motion estimation is also known as inter coding.

In H.264 block-based motion estimation is utilized. Important differences from earlier standards include multiple reference frame usage, support for different block sizes down to 4x4 and fine sub-pixel motion vectors.

2.5.1 Multiple Reference Frames

In H.264, more than one previously coded picture can be used for motion estimation. Although more memory is required in the encoder/decoder, higher compression performance is achieved, since it is not certain that the best match will be obtained in the most previous frame.

2.5.2 Variable Block size

Traditionally, motion estimation is performed only on the macro block level. Each 16x16 block will be assigned one motion vector with minimum block matching error. However, when the macro block contains multiple objects and every object moves in different directions or when the macro block lies on the boundary of a moving object, only one motion vector for the whole block will not be enough to represent true motions and it will result in serious prediction error. In H.264, in order to improve the prediction accuracy, variable size blocks are used for motion estimation/compensation. There are conceptually 7 different block sizes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4). These different block sizes actually form a one or two level hierarchy inside a macro block. Comprising only the first level, the block size can be 16x16, 16x8, or 8x16. In the case of two levels, the macro block is

specified as P8x8 type, of which each 8x8 block can be one of the subtypes such as 8x8, 8x4, 4x8 or 4x4. The four macro block type sizes and four macro block subtype sizes are shown in Figure 2-6.

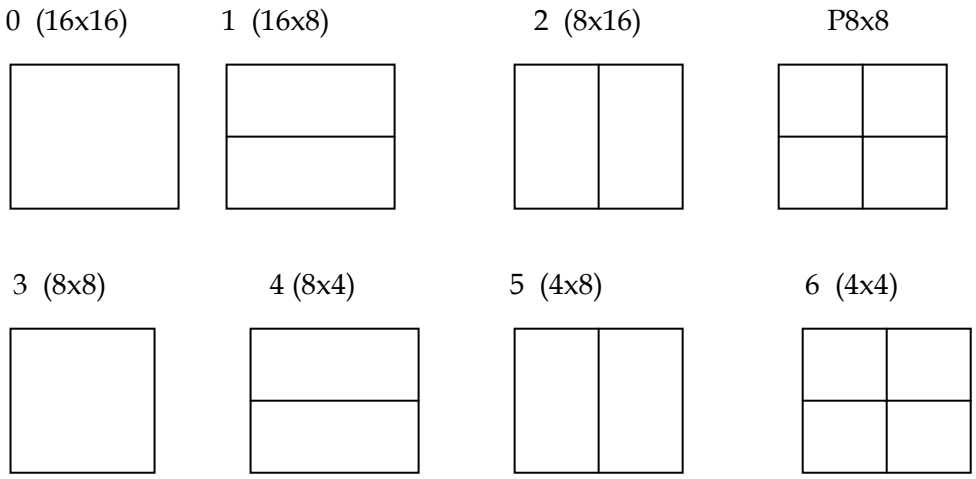

Figure 2-6 Variable size blocks

For each part of the frame the best partition size which minimizes the coded residual and motion vectors is selected. The appropriate macro block partitions for each area are shown in Figure 2-7 [11]. For the regions where there is a small motion, larger macro block partitions are chosen. In regions of detailed motion, smaller macro block partitions are chosen.

2.5.3 Sub-pixel Motion Search

A moving object does not always move exactly an integer number of pixels from one frame to another. Therefore, sub-pixel motion estimation down to a quarter pixel accuracy is supported in H.264. If the motion vector points to an integer pixel position, the prediction samples are the matching samples of the reference frame. On the other hand, if not it is necessary to create sub-pixel positions using interpolation from nearby image samples. Sub-pixel motion compensation can

provide significantly better compression performance than integer-pixel compensation, at the expense of increased complexity. Quarter-pixel accuracy outperforms half-pixel accuracy.

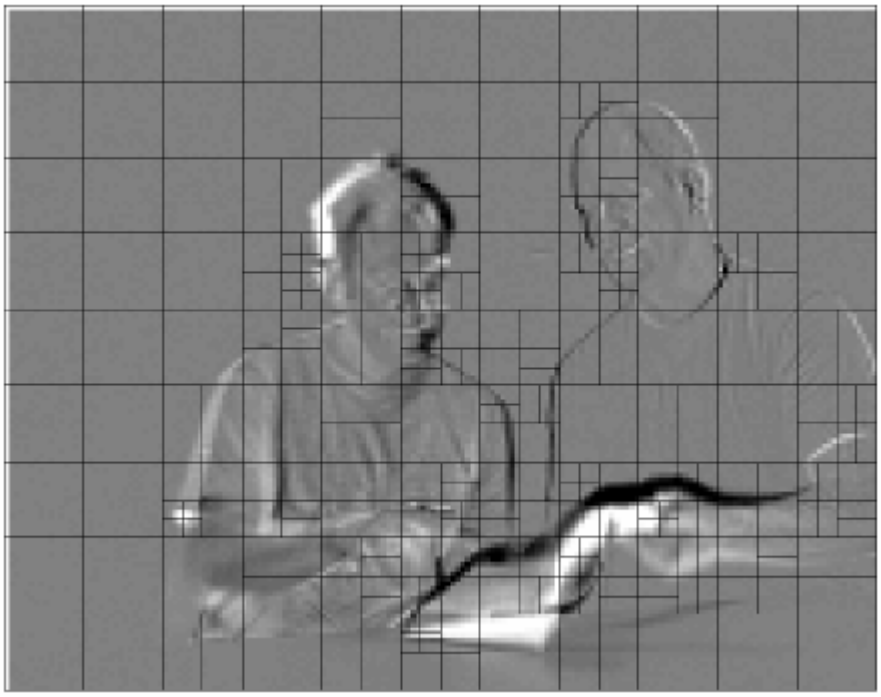

Figure 2-7 Macro block partitions

The prediction values at half-sample positions are obtained by applying a one dimensional 6-tap FIR filter whereas prediction values at quarter sample positions are produced by averaging samples at integer and the half sample positions. For the chrominance components, the prediction values are generated by bi-linear interpolation. Figure 2-8 shows the relationship between full-pixel, half-pixel and quarter-pixel.

2.5.4 Motion Vector Prediction

Encoding a motion vector for each partition can cost a significant number of bits, particularly if small partition sizes are chosen. Thus predictive coding where each motion vector is predicted from vectors of nearby previously coded partitions, is

used for coding motion vectors since motion vectors for neighboring partitions are often highly correlated. For this purpose, a predicted motion vector (MVp), is formed based on previously calculated motion vectors. Then the difference between the current vector and the predicted vector, motion vector difference (MVD), is encoded and transmitted. The prediction method depends on the partition size and on the availability of nearby vectors.

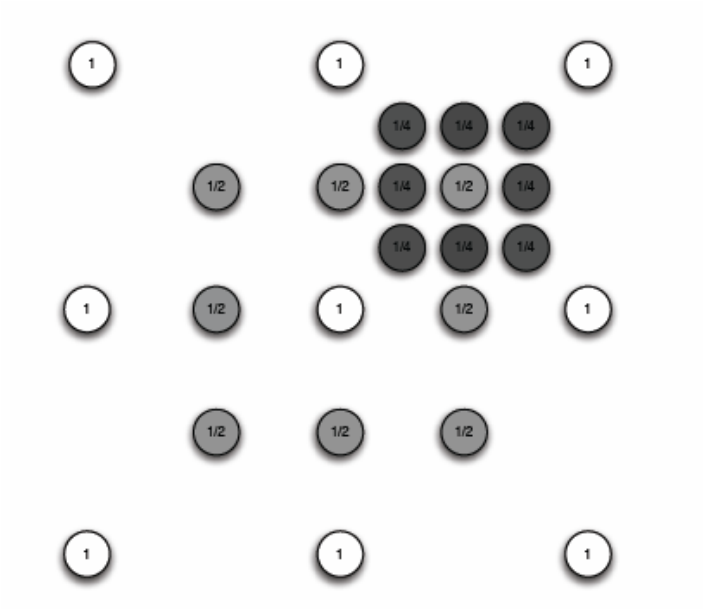

Figure 2-8 Sub-pixel interpolation

2.6 Mode Decision

As explained above there are 9 modes for intra_4x4, four modes for intra_16x16 and 7 modes for inter coding. In order to determine the best mode, a cost function which includes both the distortion and the rate, is defined in H.264. The aim is to minimize the cost:

$$Cost_{\mathrm{mod}e} = Distortion(MB) + \lambda_{\mathrm{mod}e} Rate(MB) \quad (2.1)$$

where λ_{mode} is an exponential function of the quantization parameter.

In the reference software [4], the Full Search (FS) algorithm is used to examine all the possible prediction modes to find the best mode(s). The major steps for the FS intra prediction mode(s) selection process can be summarized as:

Step1: Find the Best Intra-16x16 Prediction Mode

Generate four prediction blocks according to the four intral6x16 modes and then calculate their residual macro blocks. For each residual macro block:

- Generate four prediction blocks according to the four intral6x16 modes
- Calculate the sum of absolute difference (SAD) between the original and the predicted block
- Compute the cost of the block
- Find the mode with minimum cost, $Cost_{16x16}$

Step2:Find the Best Intra-4x4 Prediction Modes

Divide the macro block into sixteen 4x4 non-overlapped blocks. For each 4x4 block find the best mode as follows:

- Generate nine 4x4 blocks based on the nine Intra-4x4
- Calculate the sum of absolute difference (SAD) between the original and the predicted block
- Compute the cost of the block
- Find the mode with minimum cost, $Cost_{4x4}$
-

Step3:Find the Best Intra Prediction Mode(s)

If $Cost_{16x16} > \sum Cost_{4x4}$, intra4x4 is selected otherwise intra16x16 is selected.

The major steps for the FS inter prediction mode(s) selection process used in the reference software can be summarized as:

- Perform motion estimation for each 16x16, 16x8 and 8x16 size in a macro block
- Encode the macro block with the different sizes and compute the costs
- For each 8x8 block in a macro block, perform motion estimation for each 8x8, 8x4, 4x8 and 4x4 sizes
- Encode each 8x8 block with different sizes and compute the costs
- Choose the mode that gives the minimum cost among 16x16, 16x8, 8x16 and P8x8 types

2.7 Deblocking Filter

Block borders are usually reconstructed with less precision than inner samples and blocking is considered to be one of the most obvious artifacts. Thus, an adaptive in-loop de-blocking filter is used to decrease the visibility of these artifacts without much influencing the sharpness of the content. Therefore the subjective quality is significantly enhanced. De-blocking filter is used before motion compensation and this improves the compression performance since the filtered image is more similar to the original frame than the unfiltered image.

2.8 Transform and Quantization

2.8.1 Transform Coding

There is a main difference in transform coding in H.264 compared to the other common codecs. In H.264, instead of 8x8 DCT, an integer transform which is an approximation to DCT, is applied to 4x4 blocks. There are several advantages of using integer transform. First of all, one of the problems with DCT is inverse

transform mismatch. Integer transform diminishes this problem because there are no rounding errors resulting from division and multiplication of DCT coefficients. Furthermore, integer transform requires no multiplications, only additions and shifts which make it easier to implement on conventional processor types.

The transformation whose matrix is given in Equation 2.2 is the integer approximation to the 4x4 DCT and has almost identical compression performance to the DCT.

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{2.2}$$

Another transform used in H.264 is Hadamard transform which is applied to 4x4 DC coefficients in intra macro block predicted in intra_16x16 mode and to 2x2 chrominance DC coefficients in any macro block. This additional transform is used to obtain more accurate results in smooth areas. The Hadamard transform matrices are shown in Equation 2-3.

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) \quad Y_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.3}$$

2.8.2 Quantization

In H.264, a scalar quantizer is used. To avoid division which is a computationally expensive operation, multiplications and arithmetic shifts are utilized. In H.264 a new quantization scheme is developed where redundant precision on high frequency coefficients is eliminated. For a single macro block, total of 52 quantization parameters can be chosen adaptively. This introduces better quantization steps because higher transform coefficient levels are quantized with higher step sizes. The wide range of quantizer step size enables the encoder to manage the trade-off between bit-rate and quality.

2.9 Entropy Coding

There are two entropy coding techniques supported in H.264; Context-based Adaptive Variable Length Coding (CAVLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC). In the default mode CAVLC is used to encode the residual data and Exp-Golomb (Exponential Golomb) codes are used to encode syntax elements such as sequence and picture syntax elements, macro block type, quantization parameter, reference frame index and motion vector.

2.10 Network Abstraction Layer

Different from the previous standards, H.264 introduces a new approach for the video data representation where coding specific features and transport specific features are distinguished. The Network Abstraction Layer (NAL) is basically designed to supply network friendliness.

The output of the encoding process is Video Coding Layer (VCL) data where compressed video data is efficiently represented. Then VCL data are mapped to Network Abstraction Layer (NAL) units prior to transmission or storage. Each

NAL unit contains a Raw Byte Sequence Payload (RBSP), a set of data corresponding to coded video data or header information. A coded video sequence is represented by a sequence of NAL units that can be transmitted over a packet-based network or a bit stream transmission link or stored in a file. Every unit in NAL contains integer number of bytes. This reduces overhead of processing for communication and content storage or data retrieval. NAL unit standardizes an interface for packet oriented and bit-stream systems.

2.11 Profiles

There are four profiles in H.264: Baseline, Main, Extended and High [2.7].

The baseline profile handles intra coding and P-slice inter coding. Entropy coding is performed through CAVLC, Context-based Adaptive Variable Length Coding. The primary application of the baseline profile is low delay wireless communication.

In the main profile inter coding using B-slices is supported. Interlaced video can be handled and entropy coding can be performed using CABAC. The main profile is suitable for television broadcasting and video storage.

Interlaced video and CABAC are not supported in the extended profile, however it has improved error resilience and uses efficient switching between coded bit streams (SP- and SI-slices). These features make this profile useful for streaming.

Finally, the high profile supports sampling formats 4:2:2 and 4:4:4.

CHAPTER 3

FAST MODE DECISION ALGORITHMS

3.1 Introduction

The new techniques introduced in H.264 provide significant improvement in coding efficiency, average bit rate reduction of 50% given fixed fidelity comparing with the previous standards, as stated in Chapter2. However these techniques also increase the computational complexity drastically which is a drawback for real-time applications.

One of the most time consuming part of the H.264/AVC standard is the Mode Decision algorithms. In the H.264 reference software [4], there are two kinds of mode decision algorithms in the complexity sense: low and high complexity mode decision. Both methods choose the best mode by considering the cost function given in Equation 2.1.

In the low complexity mode decision, distortion is evaluated by sum of absolute difference (SAD) between the predicted frame and the original one. The rate is the number of bits required to code the mode information. On the other hand, in high complexity mode decision which is also known as Rate Distortion Optimization (RDO), distortion is evaluated by sum of square differences (SSD) between the original and reconstructed frame. In this case, the rate is the number of bits required to encode the mode information and the residue. Thus, for real time

applications, the low complexity mode decision method is a better solution since the high complexity mode requires more computation in order to achieve the best performance. However, even in the low complexity mode decision, there is still much computation since Full Search (FS) mode selection process is used.

FS mode selection algorithm finds best mode by exhaustively searching all the possible modes which corresponds to searching 9 intra4x4 and 4 intra16x16 mode for intra coding and 7 variable block size for inter coding where for each position in the search window, a large number of motion estimation is performed to find the motion vector for each variable-sized block. Obviously, computational complexity of this process is extremely high. Therefore it is highly desirable to reduce the complexity of mode decision for both intra and inter coding especially for practical applications such as real time video streaming.

In this chapter, how to reduce the computation complexity of H.264 encoder will be discussed. First, the fast mode decision algorithms in the literature will be given. Then, the proposed fast mode decision algorithm for intra and inter mode decision will be discussed and the experimental results will be presented.

3.2 Fast Mode Decision Algorithms

In literature, a number of efforts have been reported to explore new algorithms for fast mode decision. These algorithms will be examined in two ways: fast intra mode decision algorithms and fast inter mode decision algorithms.

3.2.1 Fast Intra Mode Decision

In Figure 3-1, it is shown that, among the functional modes introduced in H.264 transform, cost generation, mode decision and intra prediction almost takes %80 percent of the computational time required for intra coding [12]. Thus, the works carried out within the scope of this study are concentrated mainly on these blocks.

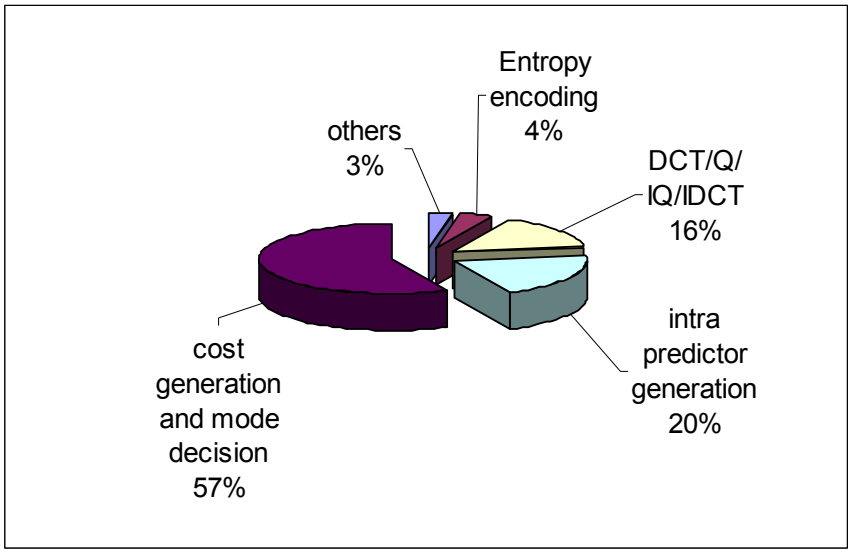

Figure 3-1 Run time percentages of intra prediction

In literature, generally there are two ways to reduce the complexity of intra prediction mode selection: simplify the cost function or examine fewer modes. A fast intra prediction mode selection algorithm is proposed in Pan [13], where Pan pointed out that the pixels along the direction of the local edge are normally of similar values for both luminance and chrominance components. In Pan's algorithm the edge information is calculated with the Sobel operator. Then the edge direction histograms are calculated for 4x4 luminance, 16x16 luminance and 8x8 chrominance blocks, respectively. In intra prediction for 4x4 blocks, the histogram cell with the maximum amplitude and the two adjacent cells are selected together with DC prediction mode as candidates for the best 4x4 prediction mode. In intra prediction for 16x16 blocks, the histogram cell with the maximum amplitude and DC prediction mode are considered as candidates of the best 16x16 prediction mode. In chrominance prediction the histogram cell with the maximum amplitude in U component, the one in V component and DC mode are selected as candidates for the best chrominance prediction mode. Although Pan's algorithm has reduced much complexity, it still needs much computation since the computation of the edge direction histogram by Sobel operator increases the computational complexity. Another approach is proposed

by Yong [14] where local edge information is extracted by a simplified method. In this method, two edge parameters are defined; vertical and horizontal. These parameters represent intensity differences between the left and right parts and between the upper and lower parts of the block. According to two parameters, one can obtain the edge direction information within the current 4x4 block. This approach reduces the computational complexity, however for some cases (QP=32) increase in bits is a drawback. Furthermore the approach is used just for the 4x4 mode selection. One of the best results in computational complexity is achieved [12]. In this algorithm sub-sampling contributes most to the computation reduction while context-based skipping of unlikely intra4x4 candidates with periodic insertion of full search blocks and adaptive threshold checking can further reduce more computation with little quality loss. However this method has also the disadvantage of increased bits.

3.2.2 Fast Inter Mode Decision

In inter coding, the main time consuming block is the motion vector search. This block contains the Motion Estimation (ME) for 7 different block-sizes (modes) and intra prediction. In order to reduce the complexity of inter coding it is logical to either simplify the motion estimation algorithm or reduce the number of modes which is also called fast mode decision. There are also hierarchic methods that combine these two. In this dissertation, the focus will mainly be on fast mode decision algorithms.

There are several methods proposed in the literature for fast inter mode decision. A fast inter mode selection algorithm is proposed [15], where Lim assigned proper potential modes for each macro block and its sub-partitions according to their homogeneity properties. Here Lim [15] makes use of the edge information and determines the homogeneity of the block by comparing the amplitude of the edge vector with a threshold. One of the drawbacks of this method is that edge information using Sobel operators is computationally complex. In Lee [16], the algorithm decides to skip mode at early stage and by calculating the boundary

error it decides to omit all intra modes. However, omitting all intra modes do not contribute much to the computational complexity. Another method depends on the absolute difference between consecutive frames[17]. The large amplitudes were assumed to appear on the moving edges or the boundaries of moving objects. There is an extra computation of absolute difference in this method and the complexity reduction is not satisfactory. A different approach to mode decision is proposed where the original and the reference macro block are firstly down-sampled to small images of ½ low resolution [18]. Then, the mode decision algorithm is applied to these macro blocks and a subset of modes is determined. Then using the modes in subset, the original macro block is encoded. Since the prediction is performed on down-sampled images, a successful computation complexity reduction is achieved. In another method developed by Kuo [19], the original frame is sub-sampled first. Then, motion estimation is performed and the result is used to predict the initial motion vector and the mean of absolute differences (MAD). With the MAD information, an initial search mode is determined using the estimated encoded bits and some predefined threshold. After the initialization, the motion vector of an MB is refined by performing motion search with the initial block size. The number of encoded bits is then estimated again using the MAD of the refined motion vector. The updated encoded bits are used to verify the mode decision and determine the next mode to search. In addition, motion search will be terminated if the number of estimated encoded bits is smaller than an adaptive threshold.

3.3 Proposed Fast Mode Decision Algorithms

As stated above, the fast intra mode decision algorithms in literature either increases the bit rate significantly or decreases the computational complexity slightly. Fast inter mode decision algorithms in the literature suffer from the similar disadvantages with fast intra mode decision algorithms, furthermore some of these algorithms use additional computation, i.e. MAD computation in Kuo [19]. In this dissertation, fast intra and inter mode decision algorithm for H.264 is proposed to overcome these drawbacks.

3.3.1 Fast Intra Mode Decision

In this section, a fast intra mode selection method which is able to reduce the computational complexity considerably while maintaining similar PSNR and bit-rate compared to the FS algorithm is proposed. This method is based on the following observations:

- Best prediction mode of a block is most likely in the direction of local edge within that block
- DC prediction mode has the higher possibility to be the best mode
- 4x4 blocks in a 16x16 block gives a general idea on 16x16 block

Considering above observation, the intra mode selection problem was approached in two ways. For intra4x4 mode selection, the number of possible modes is reduced by local edge extraction. The modes which are most likely to be the best candidate are found by the computed edge parameters. The search is done among these modes. For intra16x16 mode selection, depending on the distribution of intra4x4 modes in 16x16 block, the decision is made on whether to perform intra16x16 prediction or not. If the result of the decision is to perform the intra16x16 prediction, the number of intra16x16 candidate modes is reduced according to the information gathered from best intra4x4 modes in the 16x16 block.

Note that, there are 9 flags for each directional mode for intra4x4 prediction and 4 flags for intra16x16 prediction. When a flag is set, it denotes that the corresponding mode is a candidate and will be used in the search.

3.3.1.1 Local Edge Extraction

Two different types of edge information are calculated for 4x4 intra-prediction mode selection. There are 9 directional modes for intra4x4 prediction thus vertical, horizontal and diagonal edges are sought after to decide on the best intra4x4 candidates. For the vertical and horizontal edge extraction, the vertical

and horizontal parameters are used as proposed in [14] with different scale values. Furthermore, diagonal edge information is calculated for more accurate mode selection.

Before the local edge parameters are calculated, a 4x4 block is divided into four 2x2 blocks as shown in Figure 3-2, where I_{ij} denotes the intensity of a pixel. Here A, B, C and D respectively denotes the sum of intensity of pixels in the corresponding 2x2 blocks such as:

$$A = I_{11} + I_{12} + I_{21} + I_{22} \ . \tag{3.1}$$

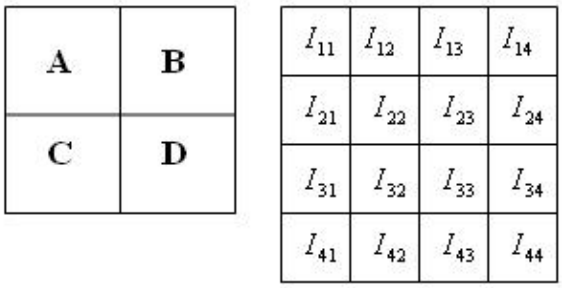

Figure 3-2 Division of 4x4 block

3.3.1.1.1 Vertical and Horizontal Edge Information

For vertical and horizontal edges, two local edge parameters, P_v and P_h, are used where;

$$P_v = \text{abs}((A+C-B-D)/S)$$
$$P_h = \text{abs}((A+B-C-D)/S) \tag{3.2}$$

and S is a scaling factor which will be discussed in Section 3.3.1.4.

3.3.1.1.2 Diagonal Edge Information

For the diagonal edge information two new local edge parameters, P_{dl} and P_{dr}, are introduced where;

$$P_{dl} = \text{abs}((A + I_{13} + I_{31} - D - I_{24} - I_{42})/\text{S});$$
$$P_{dr} = \text{abs}((C + I_{21} + I_{43} - B - I_{12} - I_{34})/\text{S}); \quad (3.3)$$

These parameters, diagonal left and diagonal right edge parameters, are shown in Figure 3-3 respectively.

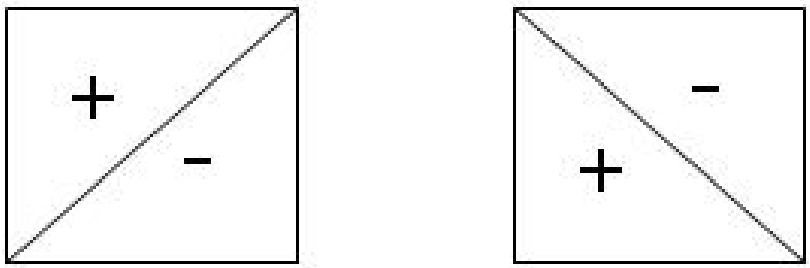

Figure 3-3 Diagonal left and diagonal right edge parameters

3.3.1.2 4x4 Intra prediction Mode Selection

The proposed algorithm for intra4x4 prediction can be depicted as follows:

Step1) Determine the modes whose flags are set based on the algorithm shown in Figure 3-4 for a 4x4 block.
Step2) Compute the cost for all the modes whose flags are set and choose the one with minimum cost
Step3) For each 4x4 block in a 16x16 block, perform steps 1 through 2
Step4) Compute the cost of the macro block, $Cost_{4x4}$

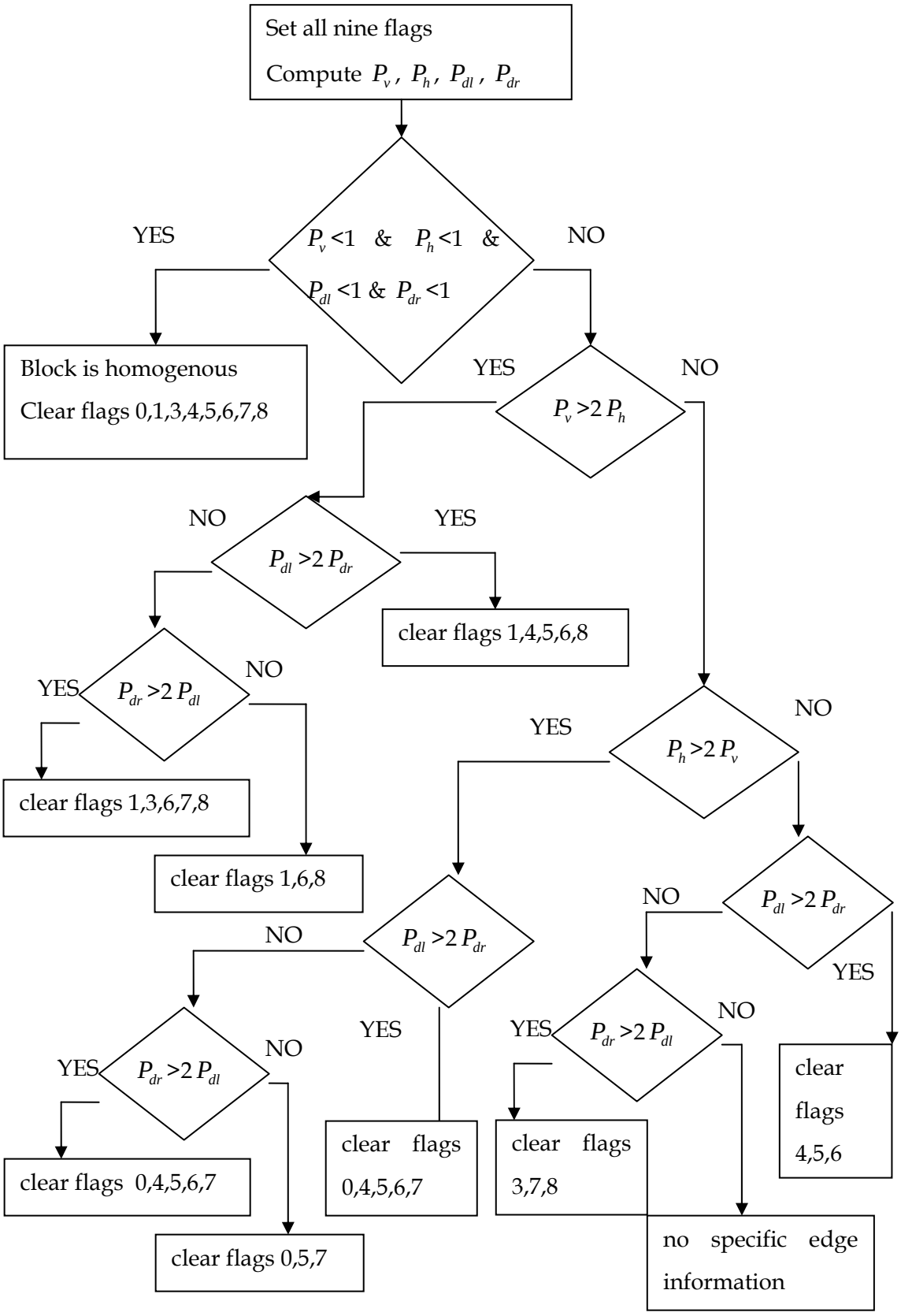

Figure 3-4 Intra 4x4 mode selection

3.3.1.3 16x16 Intra Prediction Mode Selection

The proposed intra16x16 mode selection algorithm is based on the following observations:

- intra4x4 distribution gives an idea on smoothness of a 16x16 block
- intra4x4 distribution contains edge information of a 16x16 block

The following examples elaborate these observations and may help the reader picture them. Assume that best intra4x4 prediction modes are found for each 4x4 block in a 16x16 block. If the best intra4x4 prediction modes are different from each other, i.e. some of them are DC some of them are horizontal and some of them are diagonal down left, it is obvious that the block contains lots of details. On the other hand, if the best intra prediction modes are similar to each other and one of the modes is dominant, i.e. number of vertical intra4x4 modes in a 16x16 block is larger than the number of other modes, then it is least probably that the 16x16 block has horizontal characteristics. These observations constitute a basis for our algorithm.

The proposed algorithm for intra16x16 prediction can be summarized as follows:

Step1) Compute the number of vertical, horizontal and dc modes of 16 intra4x4 in a 16x16 block
Step2) Find the mode with the highest count, C. If C<6 Proceed to step 3
Step2.1) If C is the # of vertical modes, clear flags 1, 3 and go to step4
Step2.2) Else if C is the #of horizontal modes, clear flags 0, 3 and go to step4
Step2.3) Else if C is the #of dc modes, clear flag 3 and go to step4
Step3) No need to compute 16x16 modes since the block is not homogenous. Thus equate the cost of 16x16 block to a large number and go to step5
Step4) Compute the cost for all the other modes whose flags are set and chose the one with minimum cost.
Step5) If $Cost_{16x16} > Cost_{4x4}$, intra 4x4 is selected otherwise intra 16x16 is selected.

3.3.1.4 Scaling factor Decision

Scaling factor decision is an important step in the proposed algorithm. Scaling factor basically determines whether there is a strong edge or not. In other words it is being used to decide on the homogeneity of the block. Choosing a higher value of scaling factor omits the strong edges. On the other hand choosing a small value increases the computation because in that case, though the block is homogenous the algorithm will intend to search unnecessary modes as if there is an edge. Thus it is a tradeoff between computational complexity and number of bits.

Optimum value for the scaling factor is chosen based on the experiments. The experiments are carried out for two QCIF sequences; Foreman and Container and for each sequence three quantization parameters are used. In Figure 3-5, the reduction in computational complexity is evaluated for quantization parameters; QP=16, QP=24, QP=32. Similarly, in Figure 3-6 total number of bits versus scaling factor is evaluated for the same quantization parameters. As illustrated in the Figure 3-5 and Figure 3-6, increasing scaling factor increases the saved computation along with the total number of bits which is not desired. Furthermore, the experimental results show that the quantization parameter plays an important role in the choice of the scaling factor, S. Thus, an optimum scaling factor is chosen to be used in the proposed algorithm as follows:

$$S = \begin{cases} 4 \ldots\ldots\ldots .. QP < 20 \\ 8 \ldots\ldots\ldots 20 < QP < 30 \\ 16 \ldots\ldots\ldots 30 < QP < 40 \end{cases} \tag{3.4}$$

3.3.1.5 Results

Before giving the experimental results of the proposed algorithm, experiments are carried out to show the effectiveness of the H.264 intra prediction compared to the previous standard, H.263. The experiments are conducted for 6 QCIF sequences; Foreman, Container, Coastguard, Carphone, News and Silent.

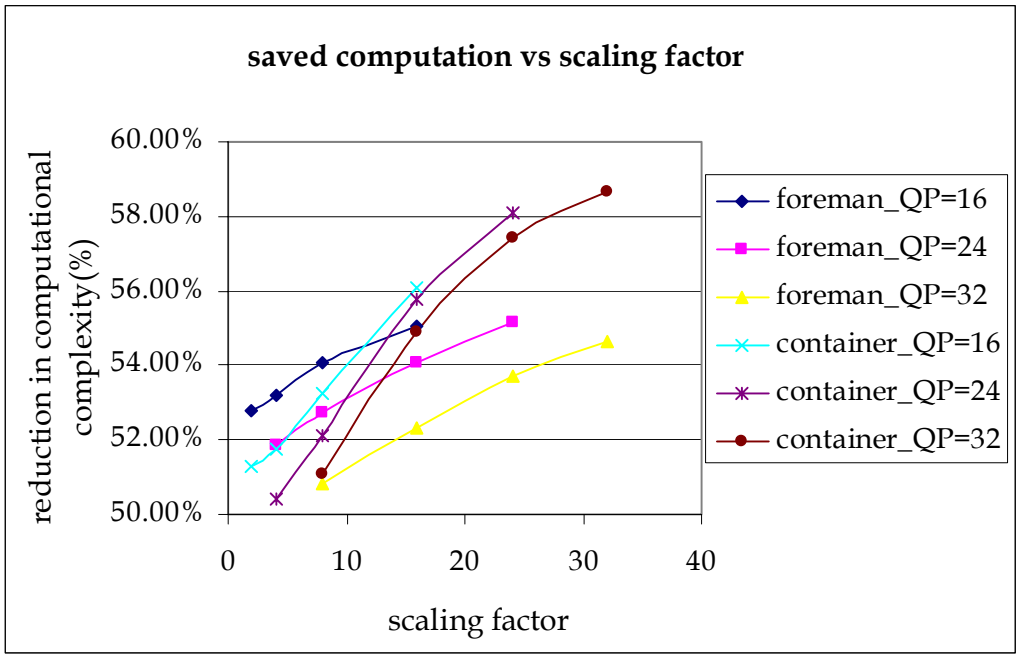

Figure 3-5 Saved computational complexity versus scaling factor

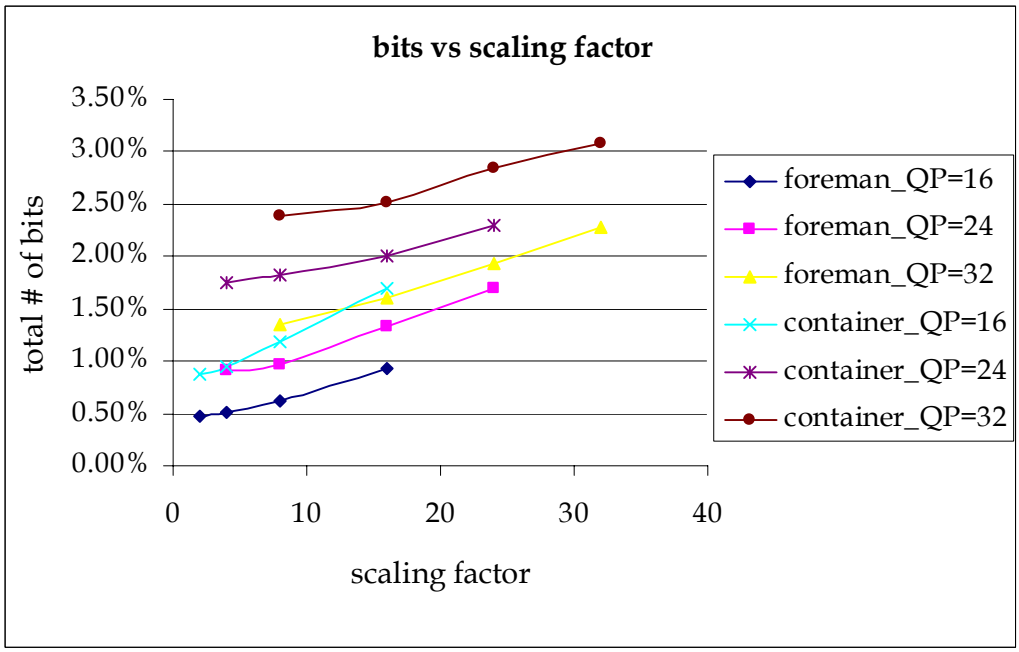

Figure 3-6 Total number of bits versus scaling factor

For each sequence, 50 frames are only intra encoded. For H.263 performance evaluation, only DC prediction mode is used. On the other hand all the directional modes are searched for the H.264 performance evaluation. Table 3-1 shows the experimental results where PSNRY which is the Peak Signal to Noise Ratio of color component Y is derived based on the equation below:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{3.1}$$

where MSE denotes the Mean Square Error.

In the table, ΔPSNRY denotes the difference in the PSNRY and ΔBits is the percent increase in bit using the H.264 intra prediction algorithm as compared with using H.263 intra prediction. As seen in the table, H.264 intra prediction outperform the H.263 both in PSNR and number of bits.

Table 3-1 Comparison with H.263

QP=32	ΔPSNRY(dB)	Δbits
foreman	0,34	-25,86%
container	0,08	-13,03%
coastguard	0,22	-9,25%
carphone	0,29	-21,85%
news	0,15	-14,01%
silent	0,27	-15,76%

The proposed algorithm was implemented into Nokia's open source H.264 encoder [4]. It is tested on 5 QCIF sequences; News, Silent, Coastguard, Container and Foreman. For each sequence, 50 frames are only intra encoded for different quantization parameters.

Table 3.2, Table 3.3 and Table 3.4 show the experimental results on five QCIF sequences for various QP. Here, ΔPSNRY is the difference in the Peak Signal to Noise Ratio of color component Y using the proposed algorithm as compared with using FS. ΔBits is the percent increase in bit using the proposed algorithm as compared with using FS. The average reduction in computational complexity with respect to FS algorithm is defined in Equation 3.6 where ΔComputation is the percentage of saved computation for intra mode decision, Avg_4x4 is the average number of searched intra4x4 modes and similarly Avg_16x16 is the average number of searched intra16x16 modes. The number of chrominance pixels is half of the number of luminance pixels, thus the ratio of the computational complexities is 9:4:2, intra4x4, intra16x16 and chrominance respectively. The computational load of obtaining the local edge parameters P_v, P_h, P_{dl} and P_{dr} are almost half of that of one search and in the experiments it is regarded as half of one search.

$$\Delta computation = (1 - (\frac{Avg\_4x4 + Avg\_16x16 + 2}{15})) \tag{3.6}$$

In Table 3-5, the experimental results of Yong's fast intra mode decision algorithm [14] are given for QP=32. As seen in the table, the algorithm achieves %60 computational reduction whereas the increase in number of bits is 4.75% on the average. Experimental results show that the proposed algorithm has better performance with a 2% increase in total number of bits for comparable computational complexity reduction.

Table 3-2 The simulation results for QP=16

QP=16	ΔPSNRY(dB)	ΔComputation	ΔBits
news	-0,02	51,35%	1,01%
silent	0,01	53,18%	0,49%
coastguard	-0,01	51,58%	0,67%
container	0,00	51,23%	0,86%
foreman	-0,01	52,83%	0,49%

Table 3-3 The simulation results for QP=24

QP=24	ΔPSNRY(dB)	ΔComputation	ΔBits
news	0,00	51,70%	1,68%
silent	0,00	53,40%	1,09%
coastguard	0,00	51,51%	1,29%
container	-0,01	50,88%	1,30%
foreman	-0,01	51,92%	0,95%

Table 3-4 The simulation results for QP=32

QP=32	ΔPSNRY(dB)	ΔComputation	ΔBits
news	0,00	51,98%	2,97%
silent	-0,03	52,31%	2,16%
coastguard	-0,05	50,68%	1,44%
container	0,00	52,91%	2,24%
foreman	0,00	50,70%	1,57%

Table 3-5 Experimental results for Yong's algorithm

QP=32	ΔPSNRY(dB)	ΔComputation	Δbits
news	-0.04	63.00%	5.25%
silent	-0.05	58.80%	5.19%
coastguard	-0.02	62.10%	3.56%
container	-0.02	64.00%	4.98%

3.3.2 Fast Inter Mode Decision

In this section, a fast inter mode selection method, which is able to reduce the computational complexity considerably while maintaining similar PSNR and bit-rate compared to the FS algorithm is proposed. This method is based on the following observations:

- 16x16 block size is used for temporally stationary blocks
- A small value of cost for 16x16 block size indicates that the macro block is most likely to be a temporally stationary block.
- Homogenous blocks have similar motion, since homogenous regions tend to move together
- Intra properties and distribution gives information on homogeneity
- If the cost of the 8x8 block is smaller than the cost of the 4x4 block, then the 8x8 block is most probably homogenous.

Considering above observations, if temporal stationary regions can be guessed at an early stage, all the block size except size 16x16 can be skipped. Moreover, if the degree of the smoothness of the block is known, the candidate modes for inter prediction can be reduced. Intra4x4 properties and distribution together with the intra16x16 properties are used to decide on the degree of the smoothness of the block. Furthermore, intra16x16 properties give an idea on the edge information which is also used to reduce the number of candidate modes. There is also another early skip stage. This is applied to the sub-blocks. When the cost of the 8x8 block is smaller than the cost of the 4x4 block, the sub block is most probably a homogenous block. The candidate modes can be reduced by skipping 8x4 and 4x8 blocks.

Note that, there are 7 flags for each block size in inter prediction. When a flag is set, it denotes that the corresponding mode is a candidate and will be used in the search.

In the following subsection, determination of the degree of smoothness will be presented first. Then, the proposed algorithm will be discussed in detail. Finally, the results of the proposed algorithm, which shows the effectiveness of the algorithm, will be summarized.

3.3.2.1 Degree of Smoothness

The degree of smoothness can be determined by intra prediction properties. The degree of the smoothness is classified into three.

- Level 1 (L1): Block is homogenous
- Level 2 (L2): Block has detailed parts
- Level 3 (L3): Block is totally composed of detailed parts

In order to determine these levels, the intra4x4 and intra16x16 properties were utilized. In the proposed fast intra mode decision algorithm, intra4x4 prediction is performed first and the pre-decision step decides whether intra16x16 prediction will be performed or not. This information is used to determine the levels. If the best intra mode is intra16x16, the conclusion is that the block is homogenous, i.e. L1. If the best intra mode is intra4x4 but the result of the pre-decision step is to perform the intra16x16 prediction, the conclusion is that even the best prediction mode is intra4x4, the block has some homogeneity, i.e. L2. Finally if the best prediction mode is intra4x4 and the result of the pre-decision step is not to perform intra16x16 prediction, the block has lots of details, i.e. L3.

3.3.2.2 Inter Prediction Mode Selection

The proposed algorithm for inter prediction can be depicted as follows:

Step1) Perform fast intra prediction
Step2) Determine the modes whose flags are set based on the algorithm shown in Figure 3-7 for a macro block.

Step3) Compute the cost for 16x16 block. If the cost is smaller than a threshold, the block is most likely a stationary block. Then, skip checking all the remaining modes and go to Step6. If the cost is larger than the threshold, continue.

Step4) For the sub-macro blocks, compute the cost for 8x8 and 4x4 macro blocks first. If the cost of 8x8 block is smaller than the cost of the 4x4 block, the sub-macro block is homogenous, thus there is no need to compute the 8x4 and 4x8 blocks.

Step5) Compute the cost for all the other modes whose flags are set and chose the one with minimum cost

Step6) Repeat the process Step1 through Step 5 for each 16x16 macro block

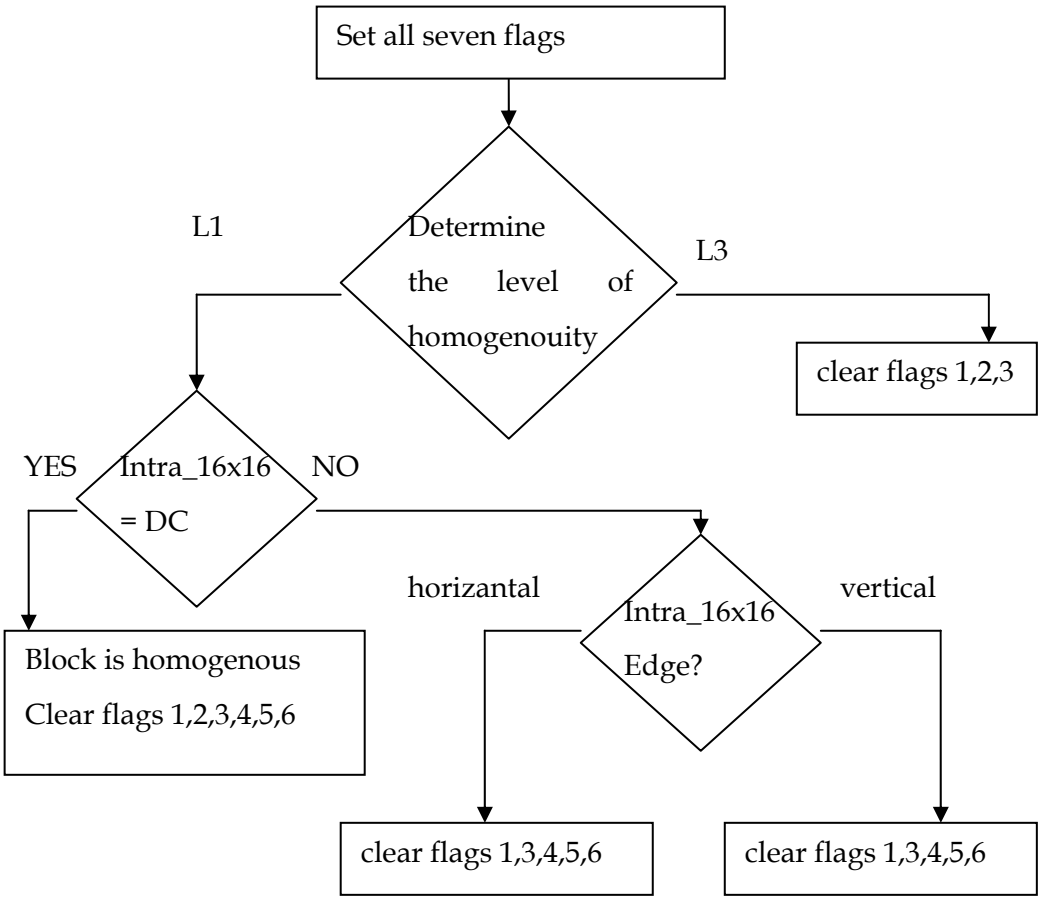

Figure 3-7 Inter mode decision

3.3.2.3 Results

The proposed algorithm was implemented into Nokia's reference software [4]. The simulations consider only the H.264 baseline profile that does not contain CABAC nor B slice. The option for Hadamard transform is turned on but the Rate Distortion Optimization is turned off. Intra frequency is adjusted to 10. The experimental simulations are performed on 6 well-known QCIF sequences with different characteristics; Akiyo(80 frames), Carphone(290), Coastguard(290 frames), Container(290 frames), Foreman(290 frames) and News(290 frames). The simulation results in terms of PSNRs, total number of bits and saved computational complexity are tabulated in Table 3-6 to Table 3-8 for three quantization parameters; QP=16, 24, 32. In these tables, ΔPSNRY, ΔPSNRU and ΔPSNRV are the the difference in PSNR of color component Y, U, and V respectively. ΔComputation is the percent reduction in computational complexity and ΔBits is the percent increase in total number of bits using the proposed algorithm as compared with using Full Search algorithm.

Table 3-6 The simulation results for QP=16

QP=16	ΔPSNRY	ΔPSNRU	ΔPSNRV	Δbits	ΔComputation
akiyo	-0.03	0.01	0.00	0.82%	58.72%
carphone	-0.06	-0.02	0.00	1.02%	48.75%
coastguard	-0.04	0.00	0.01	0.17%	42.35%
container	-0.01	0.00	0.00	0.15%	49.44%
foreman	-0.03	-0.02	-0.01	1.15%	39.60%
news	-0.04	-0.02	-0.01	1.23%	55.38%

Table 3-7 The simulation results for QP=24

QP=24	PSNRY	PSNRU	PSNRV	Δbits	ΔComputation
akiyo	-0.05	0.04	-0.02	0.94%	62.88%
carphone	-0.10	-0.04	-0.03	1.88%	57.31%
coastguard	-0.04	0.00	0.00	0.65%	43.97%
container	-0.01	0.00	0.00	0.92%	58.71%
foreman	-0.07	-0.03	-0.01	2.59%	50.06%
news	-0.06	-0.02	-0.02	1.59%	59.34%

Table 3-8 The simulation results for QP=32

QP=32	PSNRY	PSNRU	PSNRV	Δbits	ΔComputation
akiyo	-0.08	0.01	-0.01	0.91%	64.54%
carphone	-0.16	-0.04	-0.04	1.59%	62.79%
coastguard	-0.02	-0.03	0.00	1.33%	49.47%
container	-0.02	-0.01	-0.01	1.24%	58.92%
foreman	-0.15	-0.02	-0.04	2.42%	60.95%
news	-0.09	-0.04	-0.09	2.04%	60.30%

In, the experimental results of Lim's fast inter mode decision algorithm [15] are given on the average. As seen in the table, the algorithm achieves %35 computational reduction whereas the increase in number of bits is 1% on the average. Experimental results show that the proposed algorithm has better performance with 55% reduction in computational complexity for comparable increase in total number of bits.

Table 3-9 Experimental results of Lim's algorithm

	ΔPSNRY(dB)	ΔComputation	Δbits
container	-0.01	36.25%	0.30%
foreman	-0.06	25.18%	1.28%
news	-0.07	42.62%	1.18%

3.4 Discussion

In this chapter two different methods for reducing the computational complexity in mode decision of intra and inter coding were presented.

Experimental results of the proposed fast intra mode decision algorithm show that, 52% computation reduction on the average is achieved with negligibly PSNR degradation (less than 0.05dB) and slight increase in total number of bits (1.5% on the average) with respect to FS algorithm.

For the fast inter mode decision, the experimental results show that, compared to FS, the proposed algorithm achieves 55% computation reduction on the average

while maintaining similar PSNR values with a bit rate increase of 1.3% on the average. The proposed fast inter mode decision algorithm has better performance in sequences containing smooth areas and in sequences where the motion is limited since there is an early skip stage if temporal stationary and homogenous blocks exist. In these types of sequences, i.e. Akiyo, almost 65%the computation reduction is achieved. On the other hand, the proposed algorithm achieves a computation reduction of 43% in the worst case where the sequence contains both high motion and details, i.e. Coastguard.

These results indicate that the proposed algorithm outperforms the previous approaches and it is proper for real-time video streaming considering the low computational complexity.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Introduction

In this chapter a real-time video streaming system is proposed. The system composes of encoder and decoder parts which are combined with networking functionality. The structure of this chapter closely follows the order in which the work was undertaken. First, an overview on TMS320C64x DSP Family will be given. Secondly, the system architecture will be presented. Then, the H.264 encoder running on TMS320C64x platform will be discussed in detail. Finally, the decoder structure which is responsible for decoding and displaying the frames will be presented.

4.2 TMS320C64x Overview

With the emergence of new products that feature wireless digital communication, image and video processing, speech recognition, medical, and Internet telephony, designers are relying on Digital Signal Processors (DSPs) to handle real-time processing power requirements. The main advantage that separates DSPs from general-purpose microprocessors is the ability to efficiently process streams of vectors. DSPs can perform one or more Multiply and Accumulate (MAC) instructions in a single machine cycle, which is not supported by commercial processors.

In 1997 Texas Instruments (TI) developed TMS320C62x and TMS320C67x cores which use VelociTI architecture. An extension to these cores, the TMS320C64x architecture [20], which is a member of C6000 family is introduced to be used in applications that require high processing power such as video and speech processing. This newest member brings the highest performance level by use of advanced very long instruction word (VLIW) architecture. VLIW contains multiple execution units running in parallel, which allow multiple instructions to run in a single clock cycle. For instance, a 600 MHz TMS320C64x DSP offers 4800 MIPS (Million Instructions per Second).

TMS320C64x has the following enhancements:

- increased orthogonality and parallelism of instruction set
- extended data load and store paths
- increased number of register files
- packed data processing
- increased clock speed

Considering all these features TMS320C64x platform is chosen for the system implementation.

4.3 System Architecture

The proposed real-time video streaming system composes of encoder and decoder parts which are combined with networking functionality. Encoder, which is responsible for real-time H.264/AVC encoding of video, is designed to run on Texas Instrument's (TI) TMS320C64x DSP Family. TI's DM642 Evaluation Module is used for the implementation and performance evaluation. On the other hand, the decoder which runs on a PC is responsible for the real-time decoding of the bit stream and displaying the decoded bit stream. These two parts, the encoder and the decoder, communicate over Local Area Network (LAN). The overall system architecture can be seen in Figure 4-1.

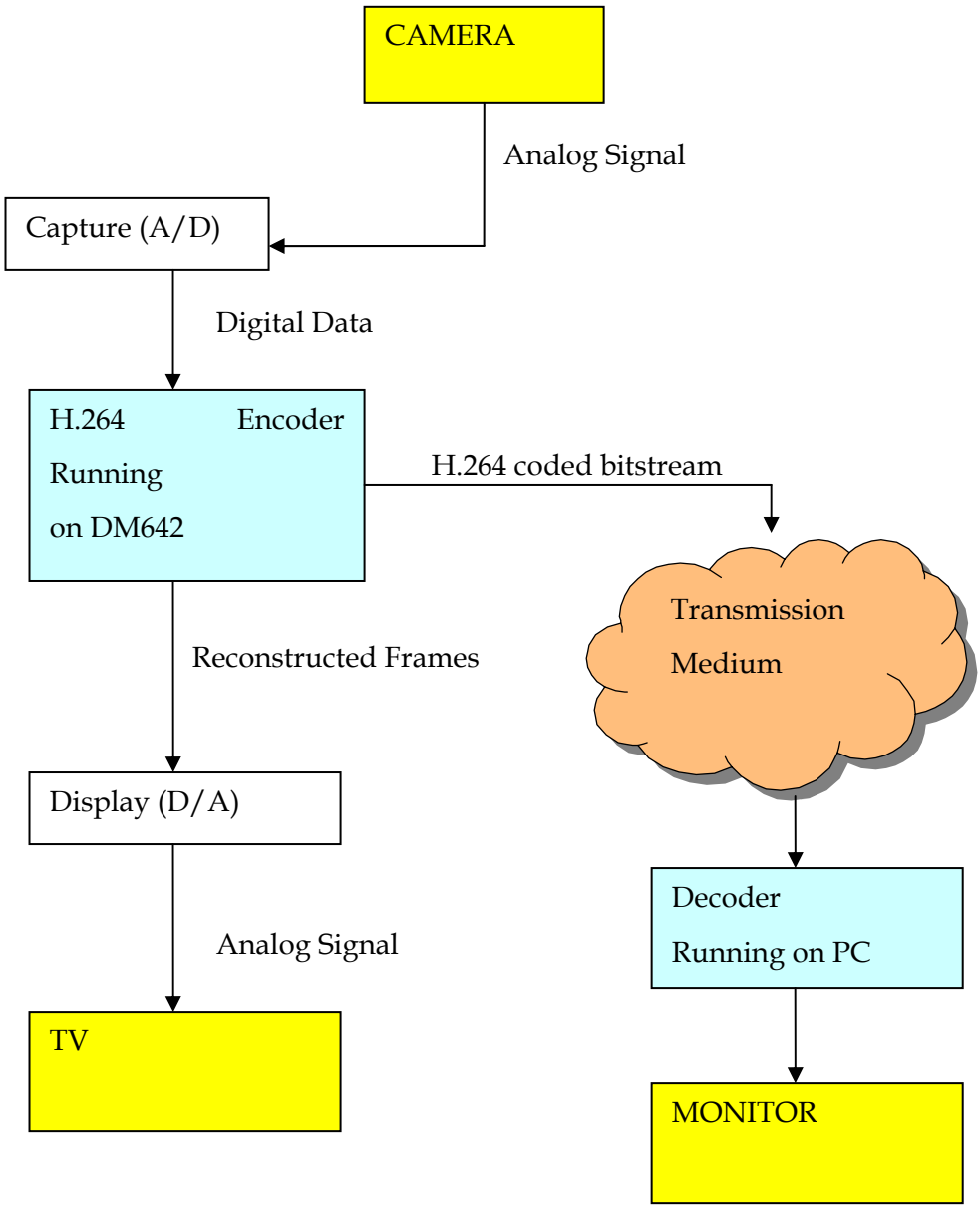

Figure 4-1 System Architecture

4.4 Encoder

This chapter mainly discusses the implementation of a real-time H.264 codec on TMS320C64x DSP Family. First, software architecture and data flow diagram of the encoder system will be given. Secondly, the required system initialization steps will be discussed. After that, encoder implementation details will be presented where the main tasks running on the system will be discussed. Then, memory management procedure will be given. Lastly, optimizations regarding TIC6000 Compiler will be discussed.

4.4.1 Overview

In the design of the proposed system, TI's DSP/BIOS [21] is used. DSP/BIOS is mainly a scalable real-time kernel which is designed to be used by applications that require real-time scheduling and synchronization, host-to-target communication, or real-time instrumentation. DSP/BIOS provides preemptive multi-threading, hardware abstraction, real-time analysis, and configuration tools.

Multi-threading feature of DSP/BIOS allows users to run multiple tasks simultaneously. However, in order to use the processor efficiency, inter-task communication is also required. For this purpose, Reference Framework 5 (RF5) [22] is used, which provides a module called SCOM, for simple single direction, zero-copy data passing among tasks.

Furthermore, being a real time video encoder, the system requires frame-by-frame video capture and display device drivers with the following features:

- A video display driver must always be displaying video data.
- Likewise, a video capture driver should always return the latest "captured" video data. This requires the driver to retain at least one frame buffer until a buffer exchange can occur.

TMS320DM642 video capture and video display mini drivers with a frame video module (FVID) are used for this purpose [23].

Moreover, we also get use of the TI Chip Support Library (CSL) [24]. CSL provides an application programming interface (API) used for configuring and controlling the DSP on-chip peripherals.

4.4.2 Software Architecture and Data Flow

The proposed system is a multi-task system where the tasks can be classified as "Input Task", "Output Task", "Networking Task" and "Processing Task". The tasks are shown in the data flow diagram in Figure 5-2 which reflects the following sequence:

- After the initializations, the system runs into multithreading system
- The processing task asks for a new input frame from the input task
- Once the request is retrieved from the processing task, a frame is captured in the input task. The acquired frame data is re-sampled and scaled to the required size. Finally the frame is fed to the processing task.
- In the processing task, a H.264 encoded bit stream is generated and then the bit stream is sent to the networking task.
- The networking task produces network data from H.264 coded bit stream and transmits these data over IP. In parallel, the reconstructed H.264 frame is generated and fed to the output task.
- In the output task, the reconstructed frame is re-sampled and is sent to the monitor.

Following sections describe these building blocks in detail.

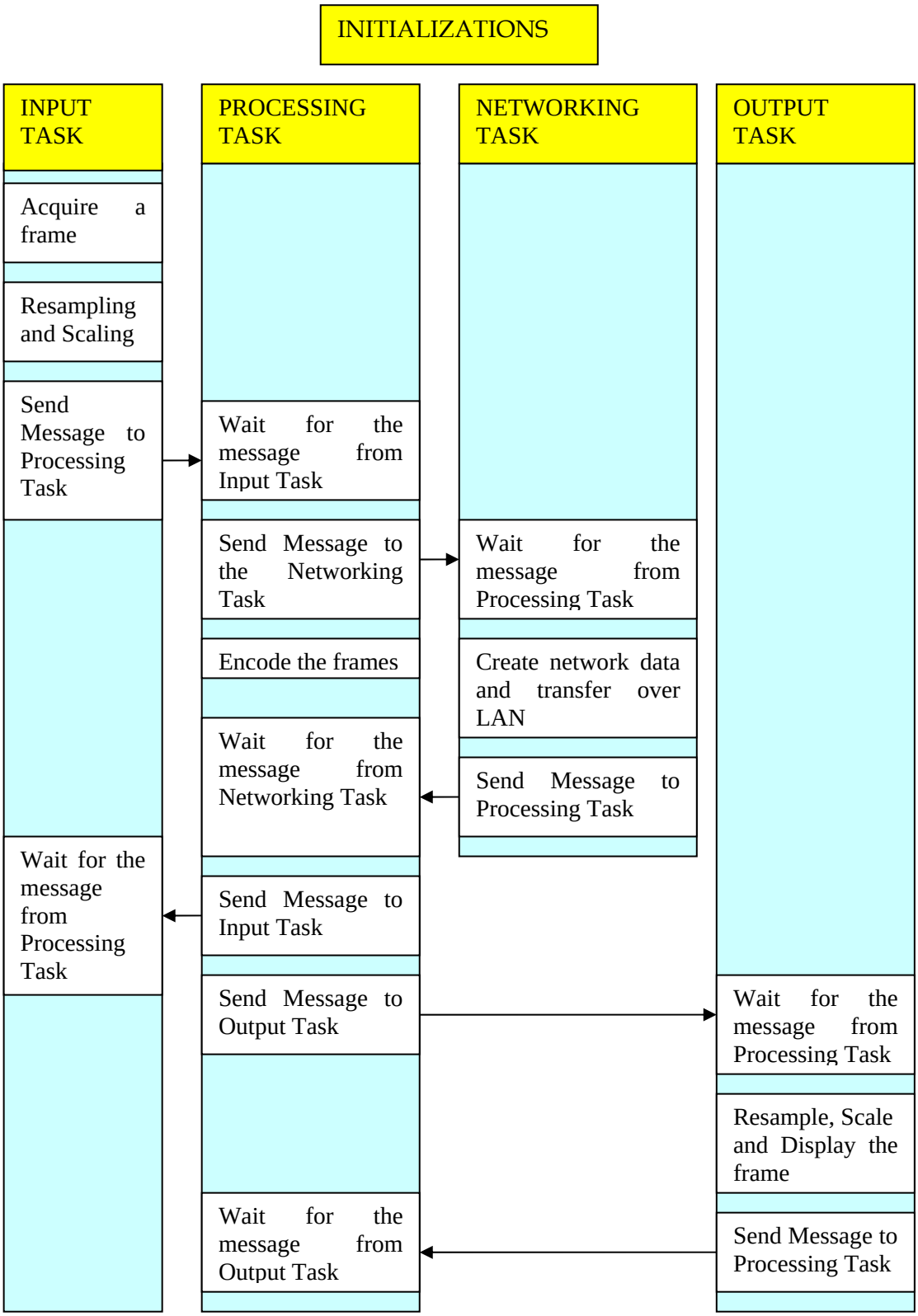

Figure 4-2 Data Flow Diagram

4.4.3 Initializations

In a DSP/BIOS program, there are two types of initialization: C runtime library initializations and user initializations. The first one takes place before the first line of main (). Before the main() is executed, the program starts running the C runtime library which initializes features like the stack, constants and initial values of global variables. After the C runtime is complete, the main () function is called. Here, main () is used simply as a place to put user initialization code. In our system, the user initialization codes includes: configuring the cache, configuring the capture and the display channels, initialization of RF5 and CSL, initialization of the network.

After all the initializations are completed, i.e. when the main () function returns, DSP/BIOS will be fully initialized and the on-chip peripherals will be set up. At the end of the process, the scheduler will be enabled in order to run the tasks. It is important to note that the main () function is used primarily as a place to do initialization in a DSP/BIOS program. The real work is expected to be placed in tasks and threads.

Below are the details of the user initialization codes that were used.

4.4.3.1 Board Specific

In this step, the DSP/BIOS and CSL is initialized first. Then, the cache is enabled and cache memory configuration is initialized. Lastly, SCOM Module in RF5 is initialized and the SCOM message objects are created.

4.4.3.2 Capture and display channels

The initialization step covers the initialization of capture channel parameters and display channel parameters.

4.4.3.3 Network Initialization

The network initialization boots up the networking environment where TMS320C6000 TCP/IP Network Developer's Kit (NDK) [25] is used as a reference.

In order to achieve IP communication, the IP address and the port number should be known. There are two ways to configure the network with an IP address. First, if an IP address is specified, IP address can be manually configured. Secondly, if an IP address is not specified, system gets an IP address from the network using Dynamic Host Configuration Protocol (DHCP). In the system developed in this study, DHCP was used to obtain the IP address and the port number was specified as 3333. After configuring the network, the last step of the initialization takes place where networking task is created.

4.4.4 Input Task

The input task whose data flow diagram is shown in Figure 4-3 acquires the frames from the input device. The input task has the following key features:

- Capturing the frames
- Communication with processing task
- Re-sampling and scaling

4.4.5 Output Task

The output task whose data flow diagram is shown in Figure 4-4 displays the frames on the output device. The output task has the following key features:

- Displaying the frames
- Communication with processing task
- Re-sampling

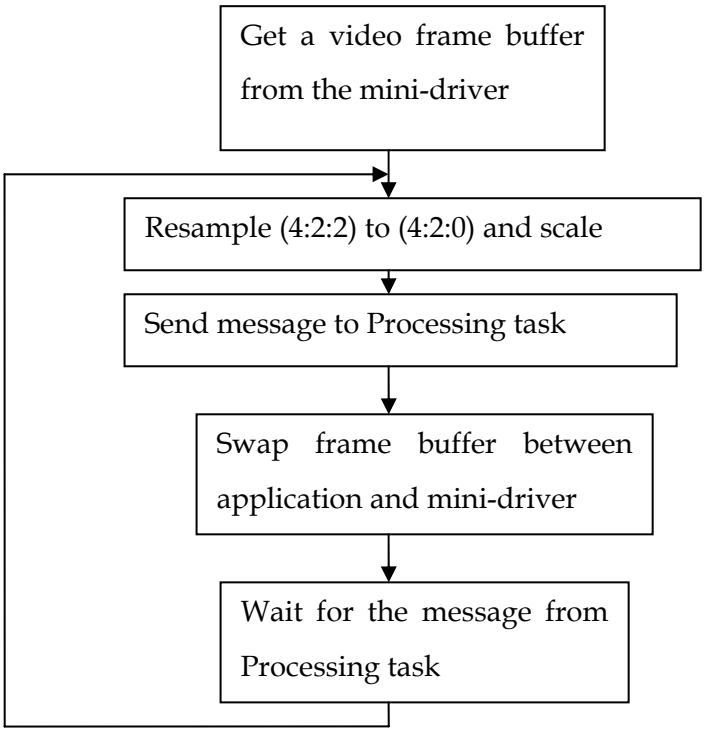

Figure 4-3 Input Task Flow Chart

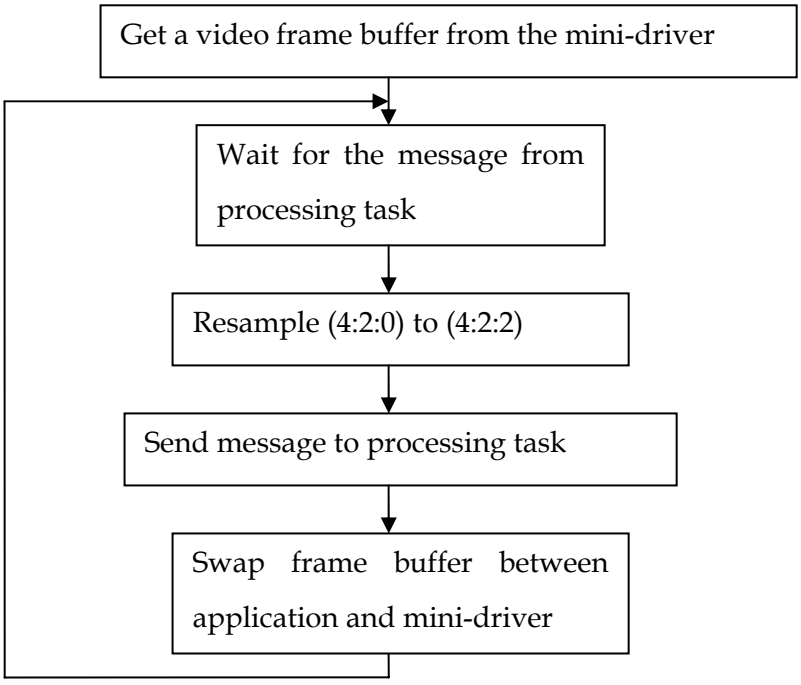

Figure 4-4 Output Task Flow chart

4.4.6 Processing Task

The processing task is responsible for encoding the frame, passing the bit stream to the networking task, reconstructing the frame, and then passing it to the output task. The data flow chart is shown in Figure 4-5.

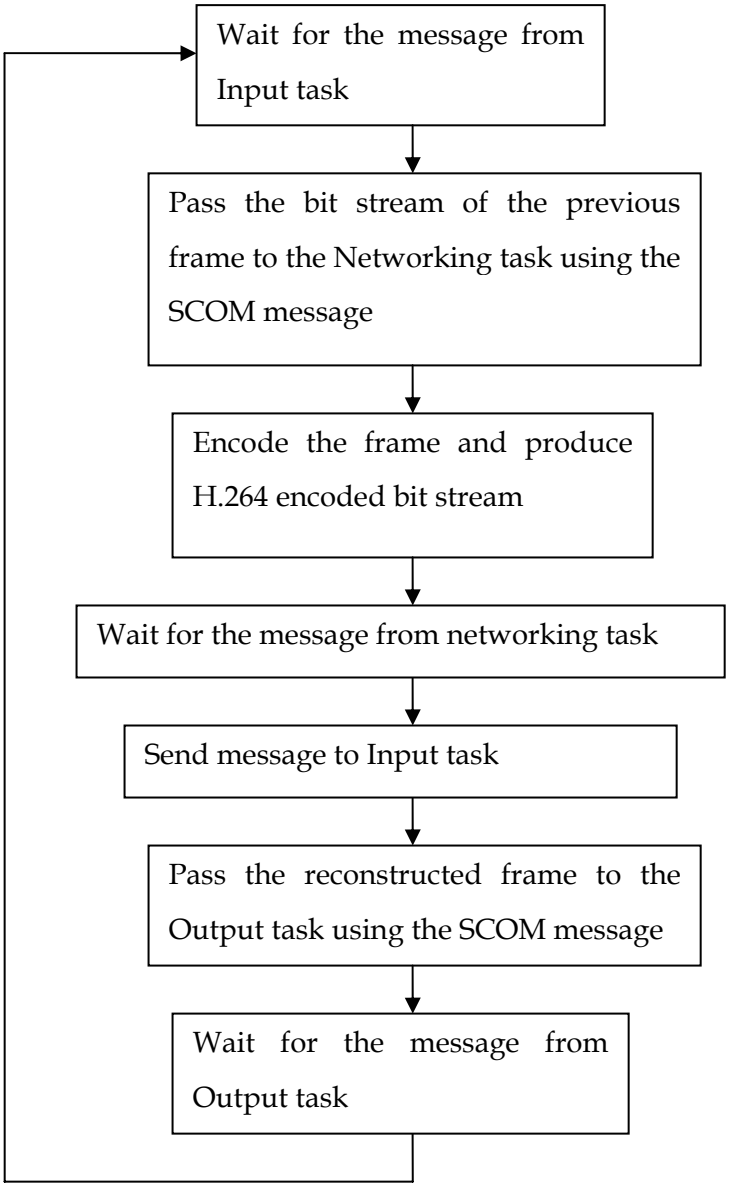

Figure 4-5 Processing task flow chart

Following sections describe H.264 encoder structure and explain briefly the basic building blocks of H.264 codec implementation.

4.4.6.1 Encoder Architecture

Common with the previous standards, H.264 Recommendation does not define the encoder but rather the syntax of encoded bit stream together with the method of decoding this bit stream is specified. The encoder specific parts, such as best matching criterion for motion estimation, inter prediction and intra prediction, are left to the one who implements the encoder.

In the following sections, the implementation details of the encoder will be discussed. The system developed herein is a real-time system. Therefore the main constraint in the implementation is computational complexity. For this reason, the baseline profile of H.264 standard has been used. In Figure 4-6, the implemented baseline profile encoder architecture is shown.

4.4.6.1.1 Sum of Absolute Difference (SAD)

In order to find best matching prediction, some comparison between predicted and original block should be made. For this implementation Sum of Absolute Differences (SAD) method is used for best matching criterion.

4.4.6.2 Intra Encoder

Intra Encoding is performed based on the proposed algorithm in Chapter 4.

4.4.6.3 Inter Encoder

4.4.6.3.1 Mode Selection

Mode selection in inter coding is performed based on the proposed algorithm in Chapter 4.

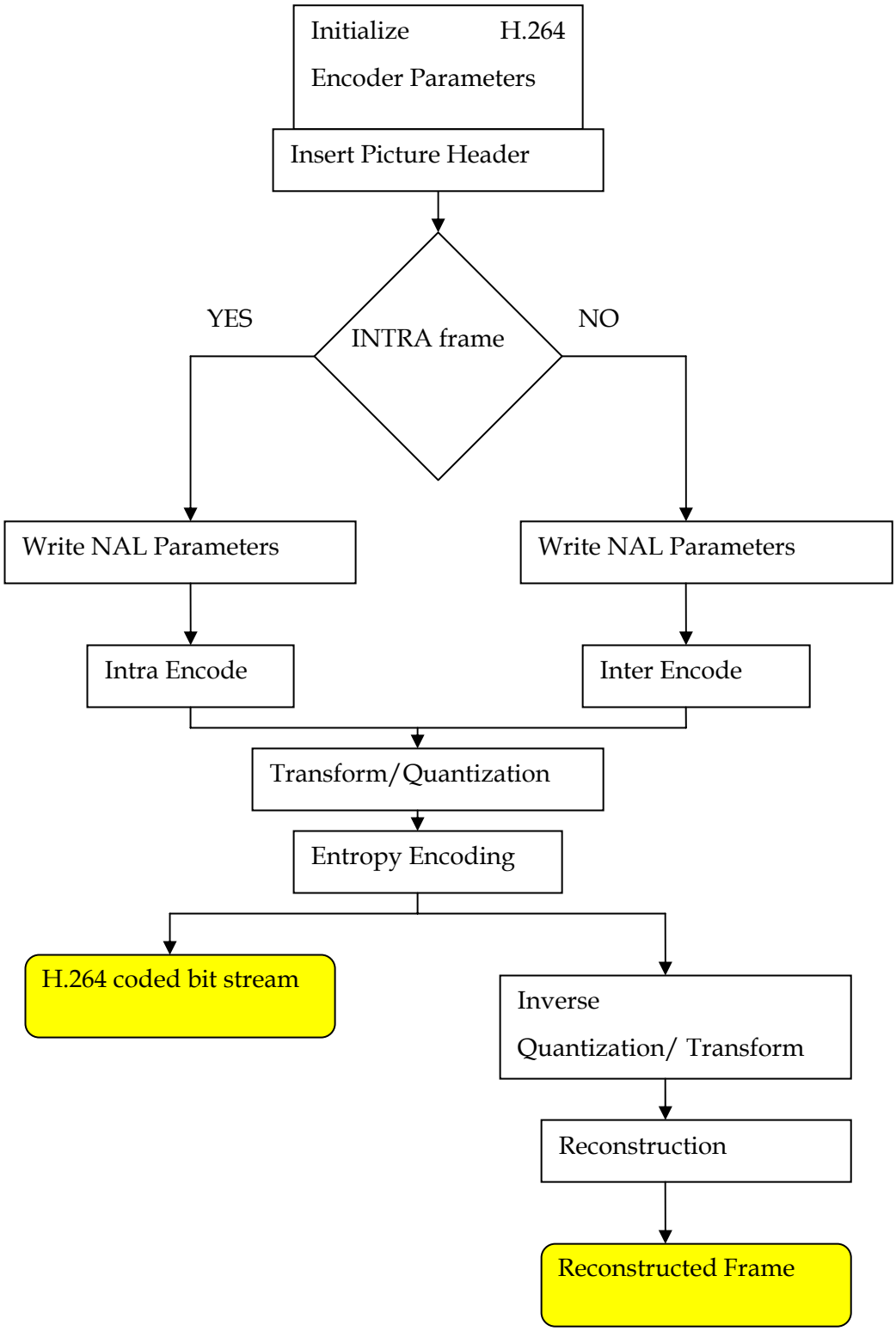

Figure 4-6 Encoder architecture

4.4.6.3.2 Motion Estimation

In video coding systems, motion estimation plays an important role in inter coding. There is a significant amount of temporal correlation between adjacent frames. Motion estimation is aimed to reduce these temporal redundancies. One of the most successful and popular methods is Block Matching (BM). The simplest BM algorithm is the Full Search Algorithm (FS) which exhaustively searches all the candidate blocks within the search area. Obviously, FS Algorithm's computational complexity is a handicap for real-time applications. In order to overcome this handicap, many fast BM algorithms are developed such as three-step search [26], new three-step search [27], four step-search [28] and block based gradient descent search [29].

A simple, robust and efficient block matching search algorithm, Diamond Search (DS) [30] is used in this implementation.

4.4.6.4 Transform and Quantization

In this dissertation the low computational complexity transform and quantization based on Malvar [31] was implemented, which is explained in detail in Section 3.8. The transforms are multiplier-free; they require only additions and a minimum number of shifts. A simplified quantization structure which reduces the size of the quantization tables was also implemented.

4.4.6.5 CAVLC

Entropy coding is the last stage of the encoder. Context Adaptive Variable Length Coding (CAVLC) was implemented based on the reference software.

4.4.7 Networking Task

The networking task performs any network functionality required in the system. The data flow diagram is shown in Figure 4-7.

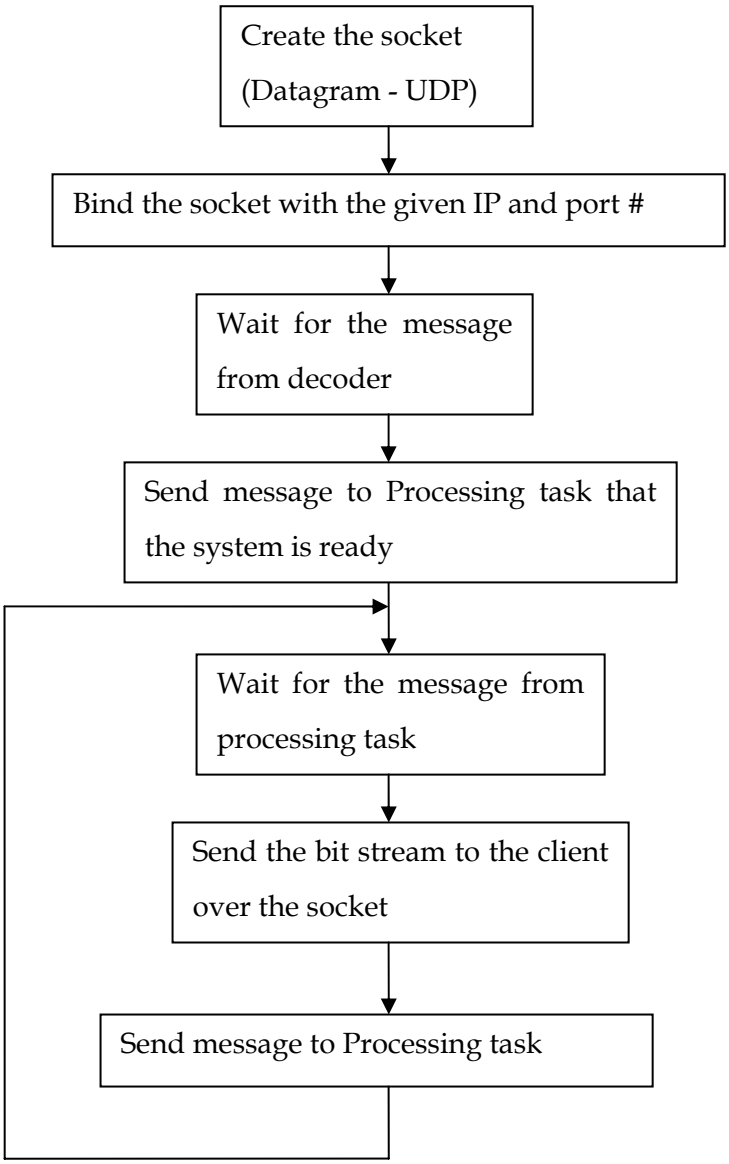

Figure 4-7 Networking Task Flow Chart

It is important that networking be handled by a separate task since it spends most of its time in a blocked state by its nature. Thus, during the transmission of a block of network data, the DSP is mostly free. Putting networking in a separate task allows those free MIPS to be utilized by other processes.

4.4.8 Memory Management

Memory management is an important issue in applications like video processing since the memory requirements is huge. The encoder memory requirements in this application is mainly due to large storage of frames, transform coefficients quantization tables and CAVLC tables. Thus, rather then using the on-chip memory which is only 256k, 1024M external memory was used. It is a known fact that external memories are slower than the on-chip memories. This problem was managed by using cache architecture which is described in Chapter 3.2.1. On the other hand, the external memory is also divided into smaller memories whose lengths and base addresses are given in Table 4-1. In this memory map, ISRAM is the internal memory, DATA_RAM is the external memory reserved for data such as variables, PROG is the external memory reserved for the program text and finally SDRAM is the general purpose external memory utilized for data such as input/output frame buffer.

memory	Base	length
ISRAM	0x00000008	0x0001FFF8
SDRAM	0x80000000	0x01600000
DATA_RAM	0x81600000	0x00400000
PROG	0x81A00000	0x00500000

Table 4-1 Memory Map

4.4.9 Optimizations Regarding TMS320C6000 Compiler

In the previous sections, software optimization techniques, fast mode decision algorithms, were presented. Moreover, fast motion estimation algorithm called Diamond Search (DS) was presented. These are encoder specific optimizations. Compiler and platform independent optimizations, such as stack management,

are also important part of the implementation. Stack management is an essential issue for all platforms. Local arrays and/or recursive functions are main reasons for the stack overflow. In a DSP system stack management is more important since there is a fix stack size different from PCs where stack size is adjusted dynamically. Thus, global variables are used for data exchange between functions which allows us to get rid of unnecessary stack allocations.

So far, all the optimization techniques proposed were either encoder specific or platform independent. Hereafter, TMS320C6000 platform dependent optimization techniques will be discussed.

The TI compiler tools can perform many optimizations that improve the execution speed and reduce the size of C and C++ programs by performing tasks such as simplifying loops, software pipelining, rearranging statements and expressions, and allocating variables into registers [32]. The C/C++ compiler is able to perform various optimizations. High-level optimizations are performed in the optimizer and low-level, target-specific optimizations occur in the code generator. High-level optimizations must be used to achieve optimal code. Figure 4-8 illustrates the execution flow of the compiler with the optimizer and code generator.

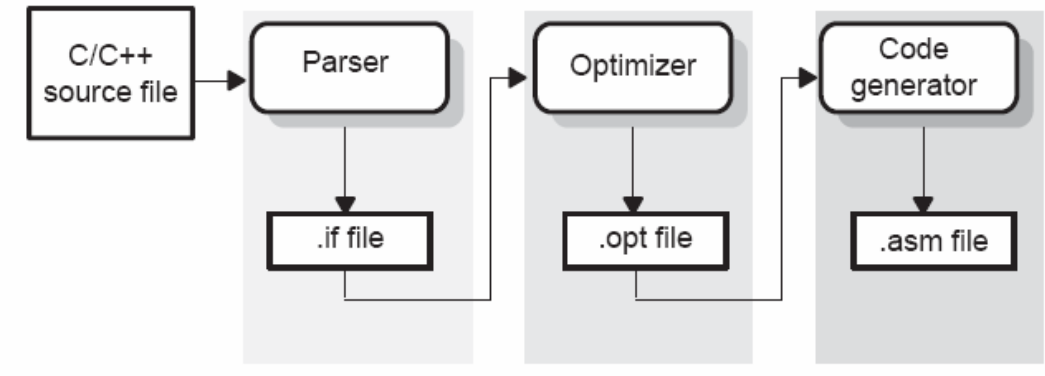

Figure 4-8 Compiling a C/C++ Program With Optimization

Cl6x compiler program is used to invoke the optimizations. There are four level of optimization, which controls the type and degree of optimization. Third level optimization is chosen which has the following key features:

- Simplifies expressions and statements
- Eliminates unused code and assignments and removes all functions that are never called
- Performs loop rotation and loop unrolling
- Performs software pipelining

Loop unrolling is an optimization method that expands small loops so that each iteration of the loop appears in the code. Although loop unrolling increases code size, it can improve the efficiency of the code. This optimization is used in the implementation in order to take advantage of increased execution performance.

Among above optimizations, one of the most important ones is software pipelining. Software pipelining is a technique used to schedule instructions from a loop so that multiple iterations of the loop are executed in parallel. The code performance increases as the number of parallel instructions increases. Figure 4-9 illustrates a software pipelined loop where A, B, C, D, and E represents the stages of the loop. The shaded area in the figure represents the loop Kernel where all five stages execute in parallel.

It is sometimes difficult for the compiler to understand independent instructions. Independent instruction means that one instruction can be scheduled in parallel. One possible solution to this problem is to give information to the compiler about independencies. The “restrict” keyword was used for this purpose in this implementation. This keyword guarantees that, within the scope of the pointer declaration, the object pointed to can be accessed only by that pointer. Since the aliasing information was provided, compiler easily optimizes the code by increasing the parallelism.

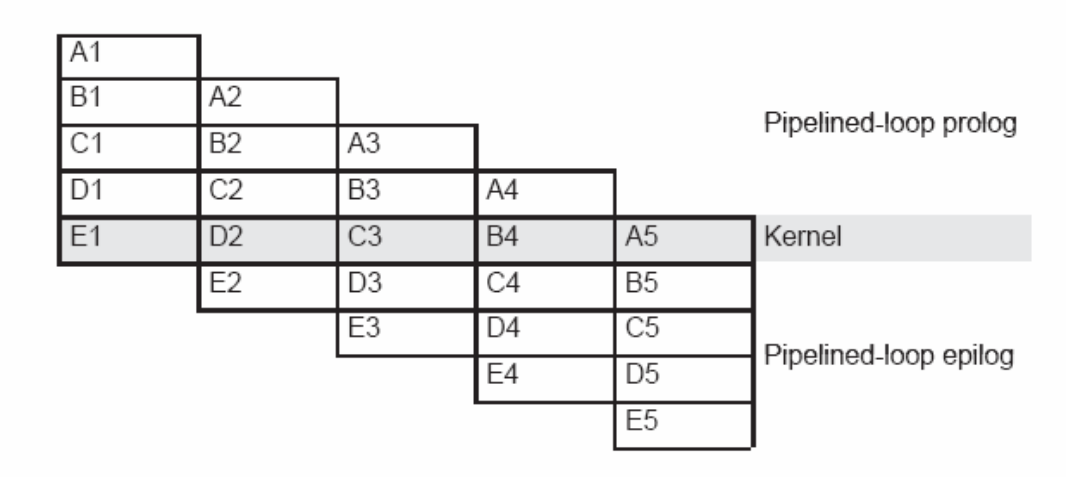

Figure 4-9 Software pipelined loop

Another keyword that has been used is "const". In general the variables are stored in RAM. However if you define a variable as "const" then you ensure that its value is not altered. Thus, you can allocate large tables into system ROM.

4.5 Decoder

This section mainly discusses the real-time H.264 decoder and the player running on a PC. First, software architecture and data flow diagram of the decoder system will be given. Secondly, the decoder system details will be presented where the main tasks running on the system will be discussed.

4.5.1 Software Architecture

The system is running on PC and uses the reference software's standard H.264 decoder which is modified as a real-time player. The decoder system has 3 main functions:

- get the network data over IP
- decode the H.264 coded bit stream
- play the decoded frames

The data flow can be summarized as:

1. The system first configures the networking environment
2. The decoder sends a message to the encoder, declaring that it is waiting for data
3. Once the request is retrieved from the encoder, the network data is sent. The acquired network data which is H.264 coded bit stream is buffered until one full frame is received.
4. When the bit stream is ready, it is decoded and the decoded frames are displayed.

Following sections describe these building blocks in detail.

4.5.2 Networking Environment

We use the windows socket functions [33] to configure the networking environment. The IP address is already known and the port number necessary to configure the network. Here it may be useful to recall that the port number in the encoder side is specified by the user and the encoder gets an IP address using DHCP and print it on the screen. Thus using this IP address and the port number we initialize the system's networking environment. The data flow chart of network is shown in Figure 4-10

4.5.3 H.264 Decoder and Display

The decoder is responsible for creating the decoded frames and displaying them on the screen. The encoded bit stream can be decoded by any H.264 decoder. However, in the proposed system, the reference software which is combined with a module that is capable of communicating over LAN is used. For displaying issues, Intel Open Source Computer Vision Library (OpenCV) [34] is used. The flow chart for this operation is shown in Figure 4-11.

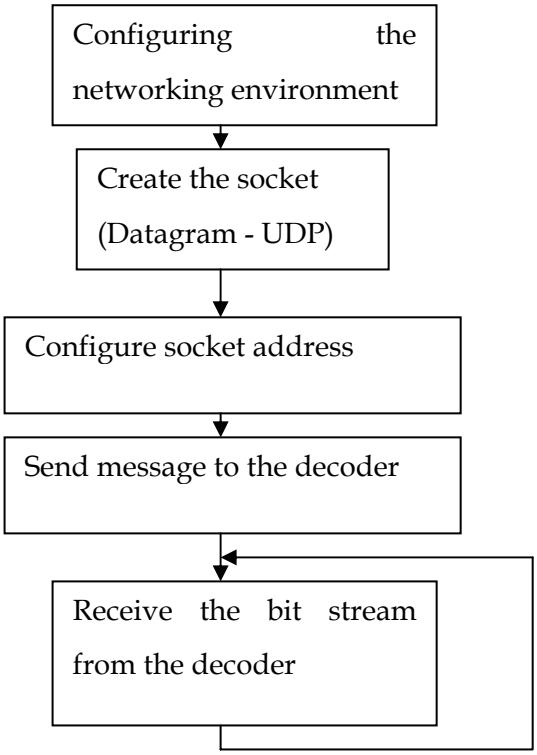

Figure 4-10 Decoder network structure

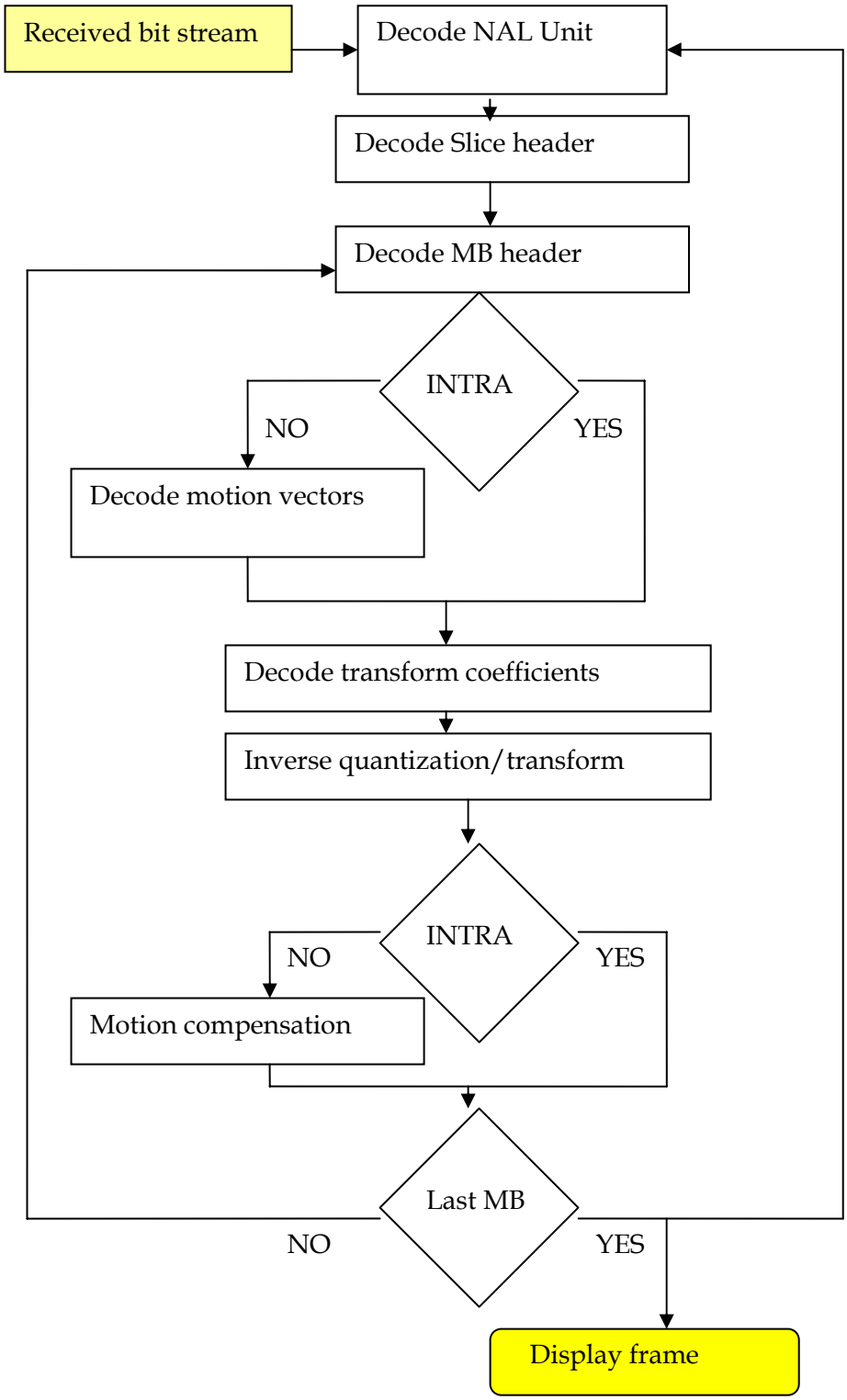

Figure 4-11 Decoder Flow Chart

CHAPTER 5

RESULTS

5.1 Encoder Performance

The proposed real time system is capable of processing colored frames with resolutions 144x176 and 288x352, which are known as QCIF and CIF frame resolution respectively.

The performance of the encoder is evaluated in two ways: off-line and real-time. The offline performance evaluation is used in order to compare the encoder performance with a software codec. Well known QCIF sequences are encoded for different quantization parameters and intra frequencies for this purpose. Secondly, we encode real-time sequences captured from the camera. For the real time case, the performance is evaluated by using two motion sequences. One of the sequence is a low motion sequence and the other one is a high motion sequence.

While evaluating the performance of the encoder, the total number of bits, PSNR and the frame-rate are measured. To calculate the frame rate, the encoding time is measured. The DSP/BIOS's CLK Module, which provides a real-time clock that can be used to measure the passage of time, is used to measure the encoding time.

5.1.1 Off-line Performance Evaluation

In the off-line performance evaluation five well known QCIF sequences with different characteristics are used; Akiyo(90 frames), Carphone(200 frames), Coastguard(200 frames), Container(200 frames) and News(200 frames). These sequences are encoded for different quantization parameters and intra frequencies. The results in this section are for QP=16,24,32 and intra frequency of 10. The simulation results in terms of PSNRs, average bits per frame and frame rate for different optimization levels are tabulated in Table 5-1, Table 5-2 and Table 5-3 for QP=32, 24, 16 respectively.

In these tables, PSNRY, PSNRU and PSNRV are the Peak Signal to Noise Ratio of color component Y, U, and V respectively. Frame/sec is the average encoded frame per second and kbits/frame is the average number of bits per frame.

To be able to observe the effects of the optimizations used, the experiments are conducted for different optimization levels. These levels are as follows:

- OPT0: The results are evaluated without any optimizations
- OPT1: The results are evaluated with only compiler based optimizations
- OPT2: The results are evaluated with the proposed software optimizations together with the compiler basedoptimizations

The experimental results show that, the encoding rate of the proposed system is 25 frames per second on the average. From the tables below, it can be inferred that the system has better performance in sequences containing smooth areas and in sequences where the motion is limited since both the proposed algorithm and the fast motion estimation algorithm gives better results for these sequences. In these types of sequences, i.e. Akiyo, encoding performance, up to 34 frames per second, is achieved. On the other hand, for sequences containing both high motion and details, i.e. Coastguard, 16 frames per second is achieved in the worst case.

Table 5-1 Test results with QP=32

QP=32		PSNRY	PSNRU	PSNRV	frames/sec	kbits/frame
akiyo	OPT0	35,71	38,95	40,48	2,0	1,706
	OPT1	35,71	38,95	40,48	7,9	1,706
	OPT2	35,57	38,96	40,47	34,7	1,723
carphone	OPT0	34,52	39,50	39,64	1,8	3,278
	OPT1	34,52	39,50	39,64	7,1	3,278
	OPT2	34,30	39,47	39,63	28,9	3,353
coastguard	OPT0	31,33	41,52	43,53	1,1	4,951
	OPT1	31,33	41,52	43,53	5,4	4,951
	OPT2	31,27	41,50	43,53	20,3	4,970
container	OPT0	33,79	39,71	39,42	1,8	2,147
	OPT1	33,79	39,71	39,42	7,2	2,147
	OPT2	33,75	39,70	39,41	28,7	2,165
news	OPT0	33,97	38,60	39,12	1,9	3,079
	OPT1	33,97	38,60	39,12	7,6	3,079
	OPT2	33,83	38,55	38,98	30,1	3,145

Table 5-2 Test results with QP=24

QP=24		PSNRY	PSNRU	PSNRV	frames/sec	kbits/frame
akiyo	OPT0	41,30	43,22	44,32	2,0	3,978
	OPT1	41,30	43,22	44,32	7,7	3,978
	OPT2	41,21	43,32	44,31	31,4	4,000
carphone	OPT0	39,99	42,66	43,12	1,6	9,165
	OPT1	39,99	42,66	43,12	6,5	9,165
	OPT2	39,81	42,61	43,08	22,6	9,457
coastguard	OPT0	37,28	44,57	45,95	1,2	17,446
	OPT1	37,28	44,57	45,95	5,4	17,446
	OPT2	37,22	44,57	45,94	16,7	17,601
container	OPT0	39,04	43,58	43,61	1,6	5,690
	OPT1	39,04	43,58	43,61	6,6	5,690
	OPT2	39,01	43,58	43,61	25,6	5,738
news	OPT0	39,91	42,60	43,26	1,9	7,021
	OPT1	39,91	42,60	43,26	7,2	7,021
	OPT2	39,83	42,57	43,23	27,2	7,183

Table 5-3 Test results with QP=16

QP=16		PSNRY	PSNRU	PSNRV	frames/sec	kbits/frame
akiyo	OPT0	47,04	48,22	48,93	2,1	9,867
	OPT1	47,04	48,22	48,93	7,7	9,867
	OPT2	47,00	48,23	48,91	27,3	9,956
carphone	OPT0	45,74	46,66	47,10	1,6	24,361
	OPT1	45,74	46,66	47,10	6,2	24,361
	OPT2	45,67	46,66	47,07	16,4	24,988
coastguard	OPT0	44,30	48,06	48,68	1,3	44,726
	OPT1	44,30	48,06	48,68	5,5	44,726
	OPT2	44,26	48,05	48,66	15,4	45,080
container	OPT0	45,11	48,15	48,22	1,8	16,943
	OPT1	45,11	48,15	48,22	6,7	16,943
	OPT2	45,07	48,15	48,21	19,7	17,022
news	OPT0	46,09	47,37	48,05	1,9	15,798
	OPT1	46,09	47,37	48,05	7,2	15,798
	OPT2	46,05	47,37	48,03	23,9	16,082

5.1.2 Real-time Performance Evaluation

For real time performance evaluation, two tests are carried out. Both of the sequences are encoded with QP=32 and intra frequency=10. For the first test a low motion sequence, moving the head, is used.Two of the encoded frames are shown in Figure 5-1.

Figure 5-1 Low motion sequence

For the high motion test, waving hand sequence is used.Two of the encoded frames are shown in Figure 5-1.

Figure 5-2 High motion sequence

Numerical test results for high and low motion sequences are shown in Table 5-4.

Table 5-4 Test results for different motion sequences

Type of the sequence	PSNR	framerate	bitrate
low motion sequence	34.58	29.2	60164.84
high motion sequence	33.55	24.8	95684.09

5.1.3 Comparison with software codec

The tests are performed for the same QCIF sequences used in off-line mode in order to make a comparison. The PC used for the tests has 1024 MB RAM, an Intel P4 3.0 GHz CPU and Microsoft XP operating system. The software encoder works 7.5 fps in average for QCIF resolution with QP=32. The proposed system outperforms the software codec with 25 fps on the average.

5.2 Decoder Performance

The decoder performance is also evaluated. Here the time needed to decode a bitstream and play the decoded frame is measured. For QCIF and CIF frame size the decoder running on a PC (1024 MB RAM, an Intel P4 3.0GHz CPU and Microsoft XP operating system) has the capability of decoding the bitstream and displaying the decoded frame in real-time.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

There are two main objectives of this study. First one is to develop algorithms that will reduce the computational complexity of H.264 standard while maintaining similar PSNR and bit-rate values compared to reference software [4]. Upon achieving this objective, further studies were carried out to implement this enhanced software on Texas Instrument's (TI's) TMS320C6000 DSP platform. In order to further speed up the encoding process, a fast motion estimation algorithm is implemented and the features of DSP are utilized. The results show that a video streaming system which meets real-time requirements with satisfying picture quality is achieved.

Two fast mode decision algorithms are proposed: intra and inter. For the fast intra mode decision, the computational complexity is reduced in two ways. For intra4x4 mode selection, local edge extraction is used to reduce the number of possible modes. On the other hand, for intra16x16 mode selection, there is pre-decision step to perform intra16x16 prediction depending on the distribution of intra4x4 modes in a 16x16 block. If the result of the decision is to perform the intra16x16 prediction, the number of intra16x16 candidate modes is reduced according to the information gathered from best intra4x4 in the 16x16 block. The experimental results show that, the proposed fast intra mode decision algorithm achieves 52% computation reduction on the average while maintaining similar PSNR values with a bit rate increase of 1.5% on the average.

For the fast inter mode decision, the computational complexity is reduced in three ways. First, if exists, the temporal stationary regions are tried to determine at an early stage. If a region is temporally stationary, all the block size except size 16x16 is skipped. Moreover, the smoothness of the block is defined and intra4x4 properties and distribution together with the intra16x16 properties are used to decide on the degree of the smoothness of the block to reduce the candidate modes for inter prediction. Furthermore, intra16x16 properties give an idea on the edge information which is also used to reduce the number of candidate modes. Finally, another early skip stage is used which is applied to the sub-blocks. If the cost of the 8x8 block is smaller than the cost of the 4x4 block, the block is most probably a homogenous block. The candidate modes can be reduced by skipping 8x4 and 4x8 blocks. Experimental results of the proposed fast inter mode decision algorithm show that, 55% computation reduction on the average is achieved with negligibly PSNR degradation and slight increase in total number of bits (1.3% on the average) with respect to FS algorithm.

In both algorithms above, since the fast algorithms do not consider all the possible candidates, there is a slight increase in the bit rate and degradation in picture quality while there is a radical decrease in the computational complexity. Moreover, it is observed that the encoder performance is highly correlated with mode decision thresholds and scaling factor. Optimum value for these parameters is decided after large number of trials and they are fixed in this implementation. Obviously all performance evaluation results may change by changing these critical parameters. Further improvement can be achieved by changing these parameters according to channel parameters and/or video sequence properties.

Motion search algorithm also significantly affects the encoder performance. Diamond Search Motion Estimation algorithm is implemented in this thesis. The algorithm concentrates at some points by using two predefined search pattern where it is more likely to find a good match. Thus perfect match for a macro block can not be achieved for every instance which results in a slight increase in bit rate and degradation in the picture quality while computation time is radically

decreased. Different search algorithms such as three step search or exhaustive search may be used as an alternative motion search algorithms.

DSP features are also utilized to speed up the encoder performance. For instance, multithreading feature of DSP/BIOS provided an effective utilization of tasks. Furthermore compiler based optimization techniques reduced the computation time significantly.

Flexible encoder architecture is created while implementing the encoder since modularity was an important issue which will enable easy further manipulations of the code and encoder structure.

The baseline profile of the H.264 recommendation is implemented in the proposed system. Any H.264 decoder can decode the coded bit stream. However, since the proposed system has also the networking functionality, the required functions are embedded to H.264 decoder.

Additional features, such as support for field pictures, generation of B, SP, SI frames can be implemented in the future work. In addition, error resilient methods can be implemented to the system in order to manage packet losses over the LAN. Furthermore, the system can be extended for use in stereoscopic video streaming.

REFERENCES

[1] Ateme Inc. , http://www.ateme.com, updated 2006, visited 20 June 2006

[2] Wireless and Wireline Communications Inc., http://www.wwcoms.com, updated 2006, visited 20 June 2006

[3] Elecard Inc., http://www.elecard.com, updated 2006, visited 20 June 2006

[4] Nokia H.264, ftp://standards.polycom.com/IMTC_Media_Coding_AG, updated September 25 2005, visited 20 June 2006

[5] ISO/IEC International Standard 11172; "Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbits/s", November 1993

[6] ISO/IEC International Standard 13818, "Generic coding of moving pictures and associated audio information", November 1994

[7] ITU-T Recommendation H.261, "Video Codec for Audiovisual Services at px64 kbit/s", 1993

[8] ITU-T Recommendation H.263, "Video Coding for very Low Bit rate Communication", 1996

[9] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC)", JVT-G050, March 2003.

[10] R. Schafer, T. Wiegand, H. Schwarz, "The Emerging H.264/AVC Standard", EBU Technical Review, January 2003.

[11] Iain E.G. Richardson, H.264 and MPEG-4 Video Compression, UK: Wiley & Sons, 2003

[12] Yu-Wen Huang; Bing-Yu Hsieh; Tung-Chien Chen; Liang-Gee Chen, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder", IEEE Trans. On Circuits and Systems for Video Technology, Volume 15, Issue 3, March 2005 Page(s):378 – 401

[13] Pan F., Lin X., Susanto R., Lim K.P., Li Z.G., Feng G.N., Wu D.J., and Wu S., "Fast Mode Decision for Intra Prediction",JVTG013, 7th Meeting: Pattaya , March, 2003

[14] Z.Yong,D.Feng,L.Shou, "Fast 4x4 Intra-Prediction Mode Selection for H.264",IEEE ICME 2004

[15] Lim K. P., Wu S., Wu D.J., Rahardja S., Lin X., Pan F., Li Z. G., "Fast Inter mode selection", Doc. I020, Sep. 2003

[16] Lee J., Jeon B., "Fast mode decision for H.264", in ICME 2004

[17] Jing X., Chau L.P., "Fast approach for H.264 inter mode decision", Electronic Letters, Vol. 40, No. 7, August 2004

[18] Dai Q., Zhu D., Ding R.,"Fast Mode decision For Inter Prediction In H.264", IEEE ICIP 2004

[19] Kuo C., Shen M. and Kuo C.-C. J., "Fast inter-prediction mode decision and motion search for H.264", IEEE ICME 2004

[20] Texas Instruments, “TMS320C64x Technical Overview, Literature Number: SPRU395B”, January 2001

[21] Texas Instruments, “TMS320 DSP/BIOS User’s Guide, Literature Number: SPRU423C”, April 2003

[22] Texas Instruments, “Reference Frameworks for eXpressDSP Software: RF5, An Extensive, High-Density System, Literature Number: SPRA795A”, April 2003

[23] Texas Instruments, “The TMS320DM642 Video Port Mini-Driver, Literature Number: SPRA918A”, August 2003

[24] Texas Instruments, “TMS320C6000 Chip Support Library API Reference Guide, Literature Number: SPRU401”, March 2001

[25] Texas Instruments, “TMS320C6000 DSP TCP/IP Network Developer’s Kit”, August 2001

[26] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” in Proc. Nat. Telecommun. Conf., New Orleans, LA, Nov. 29–Dec. 3 1981, pp. G5.3.1–5.3.5.

[27] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” IEEE Trans. Circuits Syst. Video Technol., vol. 4, pp. 438–442, Aug. 1994.

[28] L. M. Po and W. C. Ma, “A novel four-step search algorithm for fast block motion estimation,” IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 313–317, June 1996.

[29] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 419–423, Aug. 1996.

[30] Zhu S. and Ma K-K, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Trans. On Image Processing, vol. 9, no. 2, , pp. 287–290, February 2000

[31] Malvar, H.S.; Hallapuro, A.; Karczewicz, M.; Kerofsky, L., "Low-complexity transform and quantization in H.264/AVC", Circuits and Systems for Video Technology, IEEE Transactions on Volume 13, Issue 7, July 2003 Page(s): 598 – 603

[32] Texas Instruments, "TMS320C6000 Optimizing Compiler User's Guide, Literature Number: SPRU187L", May 2004

[33] Windows socket functions, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/winsock_functions.asp, updated January 2006, visited 20 June 2006

[34] Intel OpenCV Library, http://sourceforge.net/projects/opencvlibrary/, updated July 21 2005, visited 20 June 2006

[35] Texas Instruments, "TMS320C64x DSP Two-Level Internal Memory Reference Guide, Literature Number: SPRU610B", August 2004

[36] Texas Instruments, "TMS320C64x EDMA Architecture, Literature Number: SPRA994", March 2004

[37] Texas Instruments, "TMS320C6000 DSP External Interface (EMIF), Literature Number: SPRU266E", August 2004

[38] Texas Instruments, “Code Composer Studio White Paper, Literature Number: SPRA520”, May 1999

APPENDIX A

TMS320C6000 DSP FAMILY

In this appendix TMS320C64x processor family together with the code development environment will be introduced. First, an architecture overview of C64x will be given. Secondly, chip level features of C64x will be discussed. Then code development environment will be introduced. Finally the libraries and the development kits which are used in the implementation will be given.

A.1. C64x Architecture Overview

TMS320C64x CPU core which is composed of two register files, eight functional units and two data paths is shown in Figure A-1[20].

There are two register files, Register File A and Register File B, in TMS320C64x CPU Core. Each of these register files have 32 32-bit general-purpose registers which can be used for arithmetic or conditional operations.

There are eight functional units which can be divided into two groups. Each group has the same functionality. There are four different types of functional groups whose names are L, S, M, D, which are also shown in Figure A-1 [20].

Two register cross paths between two groups of functional registers exist in TMS320C64x. These cross paths allow functional units from one data path to

access a 32-bit operand from the opposite side's register file as shown in Figure A-2 [20]. This increases orthogonality, thus compiler efficiency.

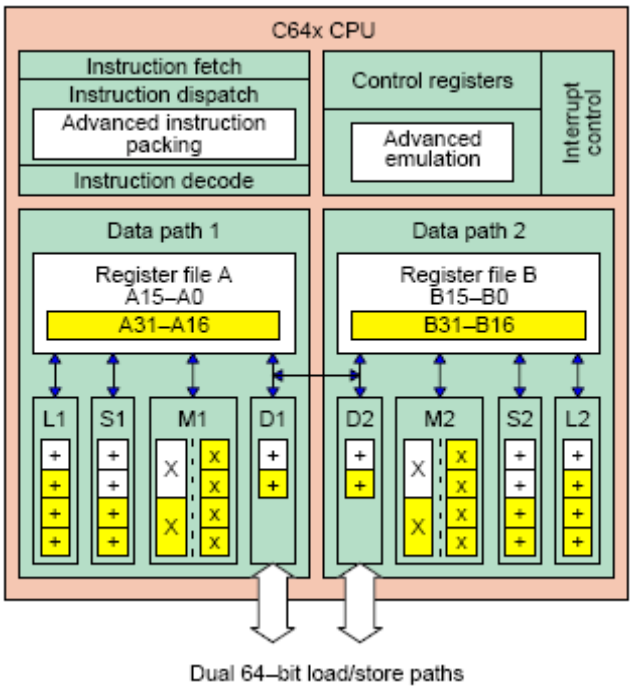

Figure A-1 C64x CPU

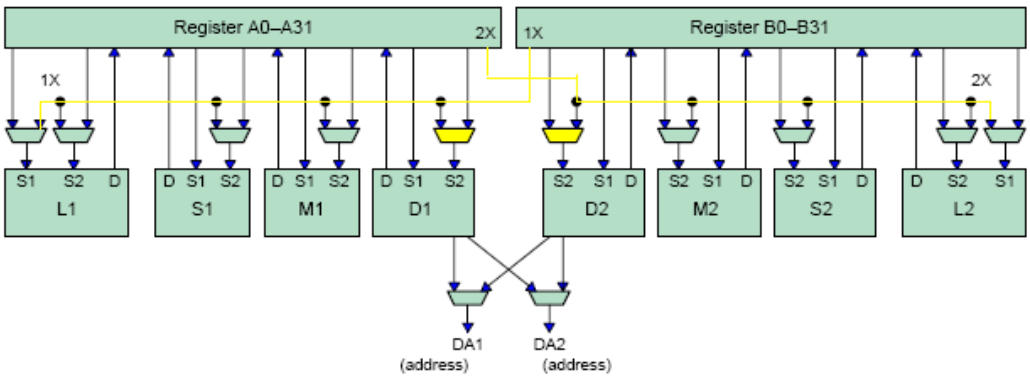

Figure A-2 C64x Data Cross Paths

The C64x supports 32 bit load and store operations. There are four paths to registers: two load and two store paths as shown in Figure A-3 [20]. In C62x and C67x architectures word or double word need alignment to 32-bit or 64-bit boundaries. However, C64x can access words or double words using non-aligned loads or stores which result in increased parallelism, thus, performance improvement.

A.2. Chip Level Features

In Figure A-4 [35], the block diagram of C64x is shown which composes of two-level memory, Enhanced Direct Memory Access (EDMA) controller, External Memory Interfaces (EMIFs) and peripherals.

A.2.1. Memory Structure

As mentioned above, a 600 MHz TMS320C64x DSP offers 4800 MIPS. Fast memory which is directly connected to the CPU (Central Processing Unit) is required to process data at this extremely high rate. However, the increase in memory speed could not catch the increase in processor speed which results in a bandwidth dilemma. Therefore, the memory to which the CPU is connected often becomes a processing bottleneck where a possible solution is caches.

Caches, which lie between the CPU and slower system memory, provide code and data to the CPU at the speed of the processor, while automatically managing the data movement from the slower memory. TMS320C64x has a two level memory structure for program and data as shown in Figure A-5 [35]. L1P, level one data cache, services data accesses from the CPU. On the other hand, L1D, level one data cache, services program fetches from the CPU. Both the program and the data memory share the second level memory, L2, which services the cache misses from both L1P and L1D.

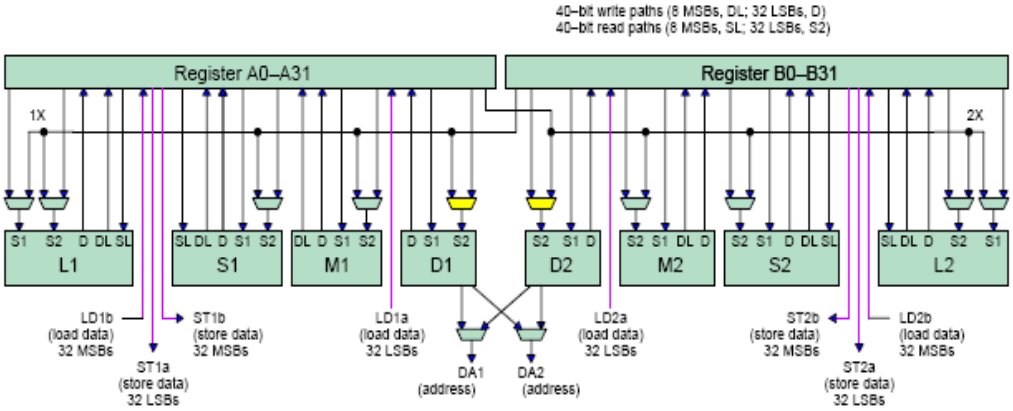

Figure A-3 C64x Memory Load and Store Paths

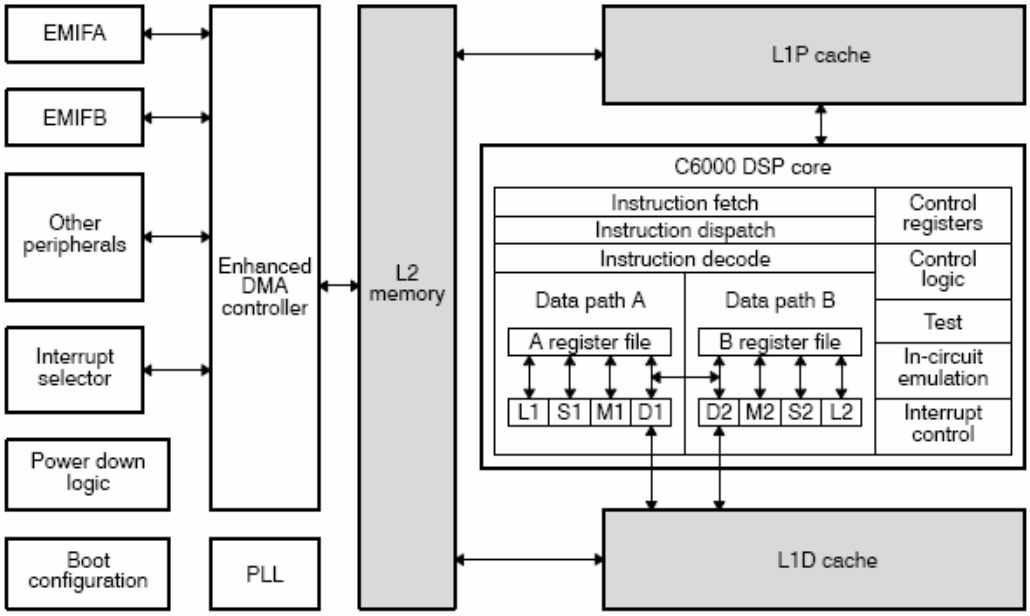

Figure A-4 TMS320C64x DSP Block Diagram

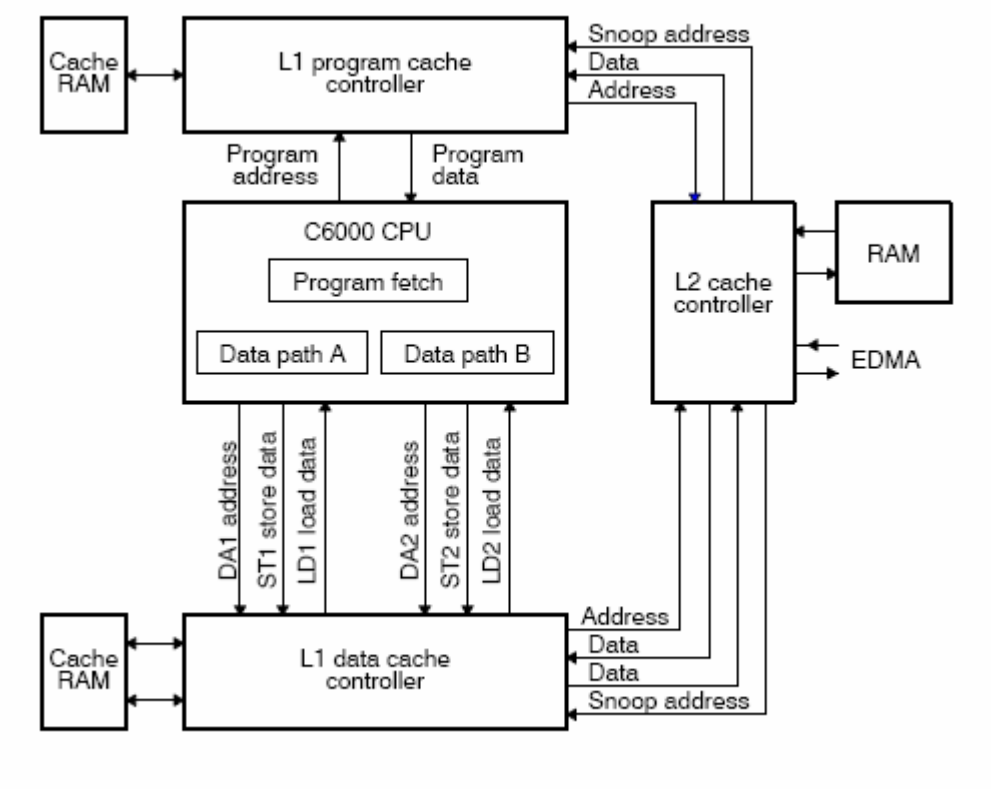

Figure A-5 TMS320C64x Two Level Internal Memory Block Diagram

A.2.2. EDMA Controller

All data transfer between the on-chip level-two (L2) memory, external memory, and the device peripherals are performed by Enhanced Direct Memory Access (EDMA) [36]. These data transfers include CPU-initiated and event-triggered transfers, master peripheral accesses, cache servicing, and non-cacheable memory accesses. The EDMA architecture has many features designed to service multiple high-speed data transfers simultaneously.

The C64x EDMA can provide 64 channels for independent data transfer. Both one and two–dimensional transfers are supported. Sub-frames of an image as well as automatically interleave or de–interleave time–division multiplexed (TDM) digital streams can be transferred using 1–D and 2–D. Byte, word, half-word, and double-word data sizes are supported.

A.2.3. External Buses

C64x processors support 3 parallel external buses in order to fulfill the high I/O bandwidth requirements. There are two external memory interfaces (EMIFs) [37] which are EMIFA and EMIFB and one host port interface (HPI). 64–bits wide EMIFA is utilized for direct connection to high speed synchronous memory, whereas 16–bit EMIFB is utilized for external I/O peripherals. The two EMIFs are identical except for their width, allowing for a variety of system designs.

32-bit HPI supports communication interface between other processors of industrial type. In some models of C64x, HPI is replaced by PCI interface. PCI bus supplies interface for PCI devices.

A.2.4. General Purpose I/O

The general-purpose input/output (GPIO) peripheral provides pins that can be configured as either inputs or outputs. The state of the input which is reflected in an internal register can be detected when configured as an input. On the other hand, the state of the output can be controlled, when configured as an output. There are a total of 16 GPIO pins some of which are multiplexed with other device pins. Furthermore, the GPIO peripheral can produce CPU interrupts and EDMA events.

A.3. Code development

Programmable DSPs provide software engineers the tools to reduce time to market along with an optimized solution to the application challenge. Sophisticated and easy to use development tools are necessary in order to focus on innovation, product differentiation, and time to market. Historically, there were two distinct DSP tools: code generation (compilers, assemblers, and linkers) and code analyzing (source code debuggers, and profilers). Since they were distinct there was no automatic sharing of data, requiring the developer to constantly switch between different applications. Today development tools

enable quick movement through the DSP-based application design process - from concept, to code/build, through debug analysis, tuning, and on to testing.

Code Composer Studio (CCS) [38] is a development environment designed for the Texas Instruments (TI) high performance digital signal processor (DSP) platforms. CCS has the following capabilities:

- Integrated development environment with editor, debugger, project manager, profiler, etc...
- C Compiler, Assembly Optimizer and Linker
- Instruction Set Simulator
- Real-Time Foundational Software (DSP/BIOS)
- Real-Time Data Exchange Between Host and Target (RTDX)
- Real-Time Analysis and Data Visualization

The development flow of most DSP- based applications consists of four basic phases: Application Design, Code Creation, Debug, and Analyze/Tune. The code development cycle of CCS is illustrated in Figure A-6 [38].

A.4. DSP/BIOS

DSP/BIOS [21] is a kernel where run-time services are provided for developers to build DSP applications and manage application resources. It is designed for applications that require real-time scheduling and synchronization, host-to-target communication, or real-time instrumentation. The DSP/BIOS provides easy-to-use powerful program development tools with the following components:

DSP/BIOS Real-Time Analysis Tools are used together with windows within Code Composer Studio to view the program as it executes on the target in real-time.

Figure A-6 Code development cycle

DSP/BIOS Application Program Interface (API) which lets the user to utilize C or assembly language functions to access and configure DSP/BIOS functions by calling any of over 150 API functions. The Embedded Target for TI C6000 DSP uses the API to let the user access DSP/BIOS from MATLAB. DSP/BIOS Configuration Tool enables the user to add and configure any and all DSP/BIOS objects that is used to instrument the application. This tool is used to configure interrupt schedules and handlers, set thread priorities, and configure the memory layout on the DSP. Select and configure the foundation modules and kernel objects required by the application with the DSP/BIOS Configuration Tool

Furthermore DSP/BIOS minimizes the memory and CPU requirements on the target in the following ways:

- All DSP/BIOS objects can be created in the Configuration Tool which reduces code size and optimizes internal data structures.
- The library is optimized to require the smallest possible number of instruction cycles, with a significant portion implemented in assembly language.
- Communication between the target and the DSP/BIOS Analysis Tools is performed within the background idle loop. This ensures that the DSP/BIOS Analysis Tools do not interfere with the program's tasks.
- Error checking that would increase memory and CPU requirements has been kept to a minimum.

DSP/BIOS also provides preemptive multi-threading. There are several thread types such as hardware interrupts, software interrupts, tasks, idle functions, and periodic functions. The priorities and blocking characteristics of threads can be controlled through the choice of thread types. Structures to support communication and synchronization between threads are also provided.

In Figure A-7 [21] the components of DSP/BIOS within the program generation and debugging environment of Code Composer Studio are shown which reflects the following sequence:

- Programs are written in C or assembly on the host PC
- The objects which will be used in the program is defined in the Configuration Tool
- Then the program is compiled and linked
- The DSP/BIOS Analysis Tools is used to test the program on the target device from Code Composer Studio while monitoring CPU load, timing, logs, thread execution, etc...

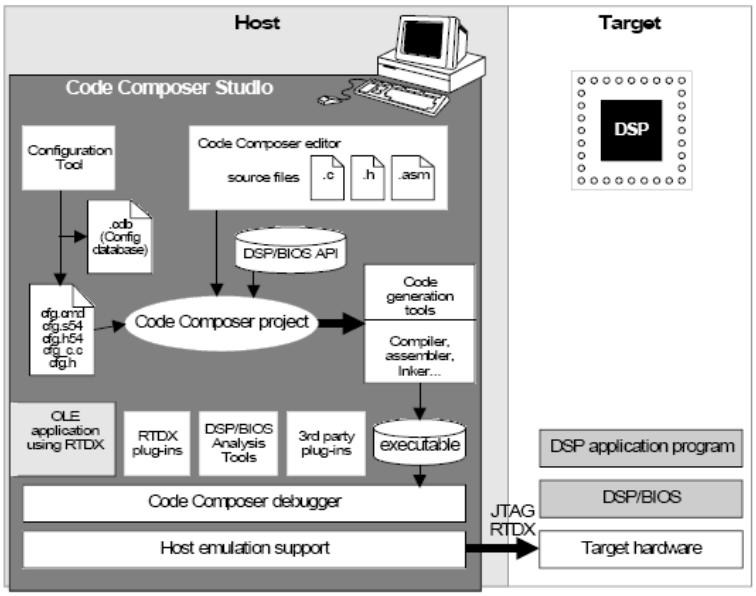

Figure A-7 DSP/BIOS Components

A.5. Chip Support Library

The chip support library (CSL) [24] is composed of discrete modules which are built and archived into a library file. CSL is written primarily in C with some assembly language where needed. Each module represents an individual application programming interface (API) and is referred to simply as an API module.

The list of CSL API Modules that are currently available is:

- CACHE cache module
- CSL top-level module
- DAT device independent data copy/fill module
- CHIP chip specific module
- DMA direct memory access module
- EDMA enhanced direct memory access module
- EMIF external memory interface module
- HPI host port interface module
- IRQ interrupt controller module
- MCBSP multi channel buffered serial port module
- PWR power down module
- STDINC standard include module
- TIMER timer module

CSL has a two layer architecture: the top layer is the service layer and the bottom layer is the hardware abstraction layer (HAL). The entire purpose of the HAL is to provide the service layer a symbolic interface into the hardware. On the other hand, the actual APIs are defined in the service layer which is the layer the user interfaces to.

A.6. Network Developer's Kit

The new TCP/IP Network Developer's Kit (NDK) [25] based on the TMS320C6000 DSP platform is a complete and easy-to-use development environment for integrating TI's TCP/IP stack with DSP applications. TI's TCP/IP stack increases system integration and simplifies the design for embedded systems needing network connectivity by running as an extra duty on the same C6000 DSP as the application. This allows designers to eliminate a separate network processor and use a more cost effective MAC/PHY device instead.

- TI's TCP/IP stack has the following features:
- NDK TCP/IP can be configured as client, protocol server or router, by adjusting the stack configuration and selecting the network services.
- Developers can program the applications using DSP/BIOS.
- The system provides nearly all the socket functions.