İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

REAL TIME FPGA IMPLEMENTATION OF A TRAINING BASED CONTENT ADAPTIVE VIDEO RESOLUTION UP-CONVERSION ALGORITHM

M.Sc Thesis by

Muzaffer Barış UYAR, B.Sc.

Department : Advanced Technologies in Engineering

Program : Computer Science

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

REAL TIME FPGA IMPLEMENTATION OF A TRAINING BASED CONTENT ADAPTIVE VIDEO RESOLUTION UP-CONVERSION ALGORITHM

M.Sc. Thesis by

Muzaffer Barış UYAR, B.Sc.

(704031013)

Date of submission : 7 May 2007

Date of defence examination: 11 June 2007

Supervisor (Chairman): Prof. Dr. Bülent ÖRENCİK

Members of the Examining Committee Prof. Dr. Ali ZEKİ

Asst. Prof. Dr. D. Turgay ALTILAR

JUNE 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

EĞİTİM TABANLI, İÇERİK UYARLAMALI BİR VİDEO ÇÖZÜNÜRLÜĞÜ DÖNÜŞTÜRME ALGORİTMASININ GERÇEK ZAMANLI OLARAK, SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ(SPKD(FPGA)) İLE GERÇEKLENMESİ

YÜKSEK LİSANS TEZİ

Müh. Muzaffer Barış UYAR

(704031013)

Tezin Enstitüye Verildiği Tarih : 7 Mayıs 2007

Tezin Savunulduğu Tarih : 11 Haziran 2007

Tez Danışmanı: Prof. Dr. Bülent ÖRENCİK

Diğer Jüri Üyeleri Prof. Dr. Ali ZEKİ

Yrd. Doç. Dr. D. Turgay ALTILAR

HAZİRAN 2007

ACKNOWLEDGEMENT

First I would like to thank my supervisor Prof. Dr. Bülent Örencik, for his guidance and support during this thesis study. I would also like to thank Toygar Akgün, and Murat Sayinta for their technical support, my manager in Vestek R&D, Ali Sayinta for his technical support, guidance and tolerance during this thesis work.

I would finally like to thank my grand mother and my family for their endless support and love.

May 2007 Muzaffer Barış UYAR

CONTENTS

TABLE LIST

FIGURE LIST

EĞİTİM TABANLI, İÇERİK UYARLAMALI BİR VİDEO ÇÖZÜNÜRLÜĞÜ DÖNÜŞTÜRME ALGORİTMASININ GERÇEK ZAMANLI OLARAK, SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ(SPKD (FPGA)) İLE GERÇEKLENMESİ

ÖZET

Bu çalışmada, eğitim tabanlı, içerik uyarlamalı bir video çözünürlük yükseltme algoritması için, iş hattı ve kaynak paylaşımı kullanan yüksek performanslı bir donanım mimarisi önerilmiş ve önerilen yapı, 480x720 çözünürlükteki videonun 720x1280 çözünürlükte videoya dönüştürülmesi uygulaması için düşük maliyetli bir sahada programlanabilir kapı dizisinde (SPKD (FPGA)) gerçeklenmiştir. İçerik uyarlamalı video çözünürlük yükseltme algoritmaları temel olarak alt örnekleme işlemi sürecinde video sinyalinde kaybolan yüksek frekans bileşenlerinin, geçmişte elde edilen istatistiksel bilgiden yararlanarak geri kazanılmasını hedefler. Bu çalışmada donanım yapısı önerilen ve gerçeklenen, modifiye edilmiş çözünürlük sentezi (MRS) algoritması, kaybolan yüksek frekans bilgisini geri kazanmak için geniş bir video görüntü kümesi üzerinde yapılan eğitim sürecinden faydalanır. MRS algoritması çıkış görüntüsünü oluşturan her piksel için 137 çarpma ve 120 toplama işlemi içeren kompleks bir algoritmadır. 480x720 çözünürlükteki standart çözünürlük (SÇ (SD)) videonun 720x1280 çözünürlükteki yüksek çözünürlük (YÇ (HD)) videoya dönüştürülmesi problemi, 27 Mhz giriş saat çevriminde üretilen piksel datası ile gerçek zaman kısıtları içerir. Önerilen donanım mimarisi, içerideki çekirdek blokların, girişteki piksel saat frekansının tam sayı katı bir frekansta çalıştırılması yöntemi ile kaynak paylaşımına olanak sağlar. Hedeflenen FPGA için, tasarım, giriş piksel saat frekansının dört katı olan 108 Mhz saat frekansında çalışacak biçimde iş hattı yapısı kurulmuştur. Bu sayede içerideki çarpma ve toplama işlemleri için kaynak paylaşımı yapılmış ve, iş hattındaki saklayıcılarda ve kontrol lojiğinde küçük bir artış ile çarpıcı ve toplayıcı sayısı dörtte birine indirilmiştir. Tasarım akışı sürecinde, donanım kısıtları ve algoritma performansı gözönüne alınarak, algoritmanın kayan noktalı yazılım modelinden, sabit noktalı yazılım modeli çıkarılmıştır. Önerilen yapının, saklayıcı transfer seviyesindeki tanımı, VHDL dili ile yazılmış; sabit noktalı C modeli ile VHDL modeli çıktıları karşılaştırılarak donanım yapısı doğrulanmıştır. Doğrulanan tasarım, Xilinx XC3S2000 FPGA çipi kullanılarak gerçeklenmiş ve likit kristal ekranlı TV üzerinde SD giriş videosunun HD videoya dönüştürülmesi uygulaması için test edilmiştir. Gerçeklenen tasarım, 1.6 ms zaman içerisinde, SD çözünürlükteki görüntüyü HD çözünürlükte görüntüye dönüştürerek video çözünürlük dönüştürme probleminin gerçek zaman kısıtlarını sağlamaktadır. Tasarım, FPGA içerisinde 3533 dilim ve yaklaşık 60 KB blok RAM yapısı kullanmaktadır. Tasarımın lojik kapı sayısı cinsinden karmaşıklığının, literatürdeki mevcut lineer video boyutlandırma algoritmaları ile yaklaşık aynı ölçekte olduğu görülmüştür. TV üzerinde gerçekleştirilen gerçek zamanlı testler sonucunda, uygulamanın geleneksel bikubik interpolasyon tabanlı video boyutlandırıcılardan daha başarılı detay koruma özelliğine sahip olduğu gözlenmiştir.

REAL TIME FPGA IMPLEMENTATION OF A TRAINING BASED CONTENT ADAPTIVE VIDEO RESOLUTION UP-CONVERSION ALGORITHM

SUMMARY

In this study, a high performance, pipelined, resource shared hardware architecture was proposed for a training based content adaptive video resolution up-conversion algorithm, and the proposed architecture was implemented in a low cost field programmable gate array (FPGA), for a video standards conversion application where the input resolution is 480x720 and the output resolution is 720x1280. Content adaptive video resolution up-conversion methods aim to recover the missing spectrum at the down sampled image, by using prior information obtained by statistical analysis. Modified resolution synthesis (MRS), which was implemented in this study is one such method, which makes use of statistical data obtained by training with large set of images. Modified resolution synthesis is a complex algorithm which requires 137 multiplications and 120 additions per output pixel. For 480x720 standard definition (SD) video to 720x1280 high definition (HD) video conversion, the design is constrained by the input pixel rate which is 27 Mhz. The proposed architecture can make use of resource sharing by running the core blocks at an integer multiple of the input pixel rate. For the targeted FPGA, the design was pipelined to work at 108 Mhz, which is four times the input pixel clock rate. Number of multipliers and adders were reduced by a factor of 4, with a minor increase in the pipeline stages and the control logic complexity. A fixed point model of the design was generated from the floating point model, by considering the hardware constraints imposed by the target FPGA, and by considering the performance of the algorithm with different bit precisions. Register transfer level (RTL) description of the proposed architecture was written in VHDL and RTL model was verified with fixed point C model outputs. The verified design was mapped to Xilinx XC3S2000 FPGA, and was tested on a 40 inch liquid crystal display (LCD) for 480p to 720p resolution up-conversion. The implemented design performs scaling of a frame in 1.6ms, which meets the real time constraints for target video up conversion application. The design uses 3533 slices, and 60KByte of block RAMS available in the FPGA. The logic gate count of the design is in the order of gate counts for bicubic scalers proposed previously. Real time tests on LCD TV shows that the performance of the algorithm is better in preserving the details when compared with conventional bicubic interpolation based scalers.

1. INTRODUCTION

1.1 Motivation

Recent advances in flat panel displays and the advent of high definition TV (HDTV) standard led to more emphasis on several video processing issues. Spatial resolution up-conversion of the input video is one such issue playing a more important role with current technological advancements. Flat panel displays, with the native resolution that supports high definition (HD) video input with resolution of 1080*1920 or 720*1280 are now starting to dominate the market, with many of the high-end TV companies, investing only on flat panel displays rather than Cathode Ray Tubes (CRT). However, those flat panel displays with HD resolution, lack from appropriate input source compatible with the native resolution of the TVs. Although the current TV sets are capable of displaying video signals on HD resolution, most of the existing content and current video broadcasting is still in SD resolution, and those high resolution flat panel displays are fed with SD video input. In such cases, the resolution of the SD input video source must be up-converted to the native resolution of the flat panel display using a mathematical method. The performance of the up-conversion method used has a considerable effect on the video quality and the effect is becoming more obvious with the fast growth rate of the size of the flat panel displays, forcing the TV set manufacturers to focus more on the issue. Apart from the video quality performance of the up-conversion algorithm, there are two other constraints, namely the real time requirements and the cost. Video processing standards are set considering the human visual system, and a pre-defined refresh rate is listed in the standards that constraints the throughput of the video system. The tight real time constraints and the need for a high performance, high complexity scaling algorithm requires high performance processing elements capable of fast parallel processing, eliminating the option of software based, sequential processing. Therefore the video up-conversion algorithms are mostly implemented in fixed hardware blocks, which are a part of the video processor inside the TV set. In this work, several high complexity, high performance video up-scaling algorithms are evaluated and Modified Resolution Synthesis (MRS) algorithm proposed in [1] is chosen to be implemented in a high end flat panel display. A novel, resource shared, pipelined hardware architecture is proposed to implement the algorithm in fixed

hardware and the proposed architecture is implemented in a low cost FPGA family, Xilinx XC3S2000 targeting the consumer electronics market.

1.2 Organization of Thesis

This thesis presents an optimized, pipelined, resource shared architecture for a training based content adaptive video up-conversion algorithm, and utilizes that architecture to implement a real time video up-conversion system integrated into a flat panel LCD display with 768*1366 native resolution.

Chapter 2 presents basics of digital video and current trends in design and implementation of video enhancement algorithms in flat panel displays.

Chapter 3 is a review of video up-conversion algorithms in the literature. Performance evaluation of the algorithms obtained from previous work on the subject is presented.

Chapter 4 describes the Modified Resolution Synthesis algorithm, which was selected to be implemented in this thesis.

Chapter 5 presents the hardware architecture proposed for the Modified Resolution Synthesis algorithm. Details of the proposed architecture are given for each sub block.

Chapter 6 presents the simulation and implementation results in FPGA platform. Real-time tests performed on TV are also presented.

Chapter 7 concludes the thesis giving a brief overview of the contribution of the thesis study, and possible feature work.

2. INTRODUCTION TO DIGITAL VIDEO

2.1 Digital Image Basics

Digital video is composed of digital image frames. The structure of a digital image is illustrated in Figure 2-1. The image taken from [2] is acquired by microwave radar from an orbiting space probe.

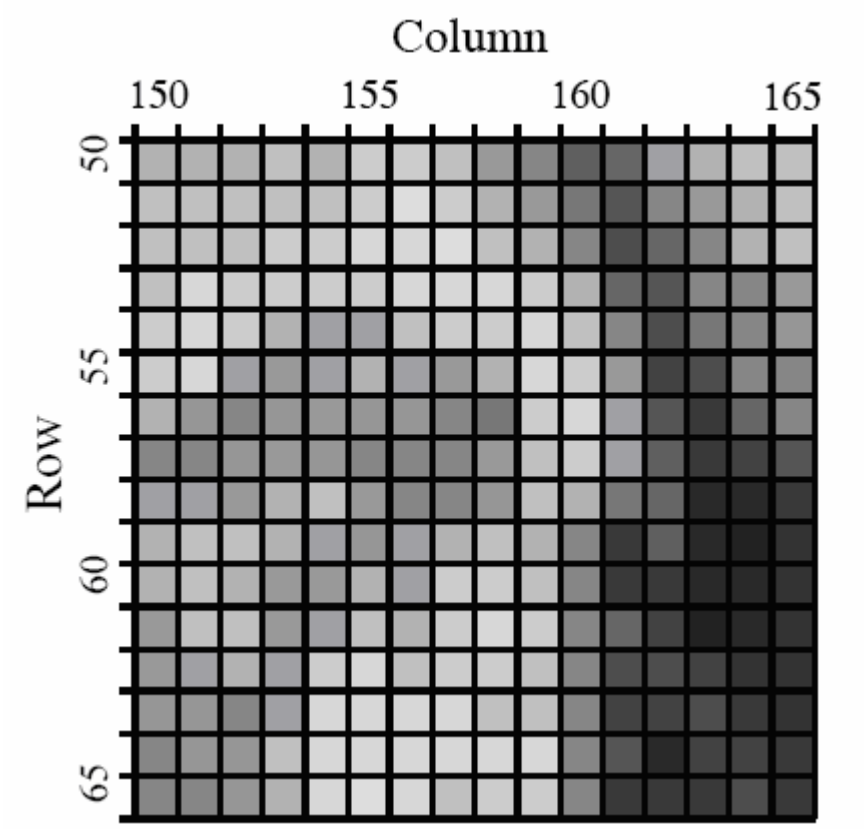

Figure 2-1 : Structure of a digital image

The image shown is represented by 256 samples arranged in a two-dimensional array of 16 columns by 16 rows. In imaging jargon, each sample is called a pixel, which is an abbreviation of picture element. The value of each pixel in the example is between 0 and 255. When displaying this as a visual image, the value of each pixel is converted into a grayscale, where 0 is black, 255 is white and intermediate values are grey levels. The mapping of the grey levels in Figure 2-1 to 8 bit numbers in the range of 0-255 is illustrated in Figure 2-2.

Row \ Column	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
50	183	183	181	184	177	200	200	189	159	135	94	105	160	174	191	196
	186	195	190	195	191	205	216	206	174	153	112	80	134	157	174	196
	194	196	198	201	206	209	215	216	199	175	140	77	106	142	170	186
	184	212	200	204	201	202	214	214	214	205	173	102	84	120	134	159
	202	215	203	179	165	165	199	207	202	208	197	129	73	112	131	146
55	203	208	166	159	160	168	166	157	174	211	204	158	69	79	127	143
	174	149	143	151	156	148	146	123	118	203	208	162	81	58	101	125
	143	137	147	153	150	140	121	133	157	184	203	164	94	56	66	80
	164	165	159	179	188	159	126	134	150	199	174	119	100	41	41	58
	173	187	193	181	167	151	162	182	192	175	129	60	88	47	37	50
60	172	184	179	153	158	172	163	207	205	188	127	63	56	43	42	55
	156	191	196	159	167	195	178	203	214	201	143	101	69	38	44	52
	154	163	175	165	207	211	197	201	201	199	138	79	76	67	51	53
	144	150	143	162	215	212	211	209	197	198	133	71	69	77	63	53
	140	151	150	185	215	214	210	210	211	209	135	80	45	69	66	60
65	135	143	151	179	213	216	214	191	201	205	138	61	59	61	77	63

Figure 2-2 : 8 bit coding of the image in Figure 2-1

A typical digital image is composed of about 500 rows by 500 columns. This is the image quality encountered in television, personal computer applications, and general scientific research. Images with fewer pixels, are regarded as having unusually poor resolution. These low resolution images look noticeably unnatural, and the individual pixels can often be seen. On the other end, images with more than 1000 by 1000 pixels are considered exceptionally good. This is the quality of the best computer graphics, high-definition television, and 35 mm motion pictures. The strongest motivation for using lower resolution images is that there are fewer pixels to handle which reduces both the transmission and processing complexity. It is common for 256 grey levels (quantization levels) to be used in image processing, corresponding to a single byte per pixel. This choice is mainly due to the fact that a brightness step size of 1/256 is smaller than the human eye can perceive. However some images are now stored with 10 bits, and current video processors start using

10 bit processing as well .This is mainly done to reduce the undesired effects of 8 bit processing(e.g. false contours, quantization noise) [2].

Color is added to digital images by using three numbers for each pixel, representing the intensity of the three primary colors: red, green and blue. Mixing these three colors generates all possible colors that the human eye can perceive. A single byte is frequently used to store each of the color intensities, allowing the image to capture a total of 256×256×256 = 16.8 million different colors [2].

2.2 Algorithms in Digital Video Processing

Raw video data at the inputs of a TV set pass through several processing steps in the TV processors' pipeline. The main stages in the video processing pipeline can be listed as

- Video format conversion
- Interlaced to progressive conversion
- Resolution up-conversion
- Coding artifact reduction
- Contrast enhancement
- Sharpness enhancement
- Color saturation enhancement
- Spatial and temporal denoising

Video format conversion refers to the issue of converting video signals which are recorded at different frame rates. Video cameras use a picture rate of 50 or 60Hz, while movie films are recorded at 24, 25 or 30Hz. The picture rate of TV and PC displays lie between 50 to 120Hz. High quality picture rate conversion methods make use of motion estimation and compensation techniques to predict the missing information [3].

With interlaced scanning, only half of the scanning lines of individual pictures are transmitted and reproduced at a TV receiver. The first field is made of the odd scanning lines and the second field is made of the even scanning lines. It has been shown that interlaced video display matches the demands of the human visual system very well however; the interlacing procedure is a complication for many digital processing tasks and also most modern displays cannot handle interlaced signals well. Therefore, de-interlacing, or interlaced-to-progressive conversion,

doubles the vertical-temporal sampling density to produce a suitable signal for the display [3].

With the introduction of HDTV-capable TV receivers, the transmission of SDTV material does not stop immediately, which requires up-conversion at the TV set. A similar situation occurs with PCs that have a screen resolution that is higher than required for television. In general the price of high resolution screens has come down to a level that it becomes affordable, even for TVs that have no HDTV reception. Consequently, resolution up-conversion is a hot topic to provide HD image quality from SD video source [3].

Coding artifacts and noise are two important issues apparent in video. Coding artifacts are apparent in digital video if the video is coded with low bit rate. Noise is introduced when either acquiring the video data, or in conversion steps (e.g. analog to digital conversion, quantization etc.) Both coding artifact reduction techniques and denoising techniques are also available in the current video processor chipsets. To improve the perceived image quality, current video processors also have color, contrast and sharpness enhancement blocks.

3. VIDEO RESOLUTION UP-CONVERSION ALGORITHMS

3.1 Sampling Structure Conversion

Video up-conversion problem can be defined as a subset of "sampling structure conversion" problem. A continuous time signal $x_c(t)$ can be represented as a discrete-time signal

$$x[n] = x_c[nT] \qquad (3.1)$$

where T is the sampling period. A 1D discrete time signal with sampling period of T can be converted into a discrete time signal with sampling period of T'

$$x'[n] = x_c[nT'] \qquad (3.2)$$

One approach to obtain $x'[n]$ from $x[n]$ is to reconstruct $x_c(t)$ by using a reconstruction filter, and a digital to analog converter, and than re-sampling the resultant analog signal with sampling period T'. However, this approach is usually not preferred due to the non ideal characteristic of the reconstruction filter and A/D, D/A filters [4].

3.1.1 Reducing Sampling Rate by an Integer Factor

Sampling rate of the discrete time signal, $x[n]$, in equation (3.1), with sampling period of T, can be reduced by a factor of M by down sampling $x[n]$ at a sampling period of $T' = M.T$. This process is referred as decimation. Decimation process can be modeled in two steps.

1. Multiplication by an impulse train to replace $M-1$ samples between every M^{th} sample with zero to obtain an intermediate signal $g[n]$

$$g[n] = x[n]\sum_{k=-\infty}^{\infty}\delta(n-kM) \qquad (3.3)$$

2. Discarding the zero samples to obtain the down sampled signal

$$x'[n] = g(Mn) \tag{3.4}$$

Fourier transforms of the signals $g[n]$ and $x'[n]$ can be computed as,

$$G(\omega) = 1/M \sum_{k=0}^{M-1} X(\omega - \frac{2\pi}{M}k) \qquad -\pi \le \omega < \pi \tag{3.5}$$

$$X'(\omega) = 1/M \sum_{k=0}^{M-1} X(\frac{\omega - 2\pi k}{M}), \qquad -\pi \le \omega < \pi \tag{3.6}$$

The frequency spectrum of the input, and decimated signals for $M = 2$, are illustrated in Figure 3-1 and Figure 3-2 respectively. It must be noted that horizontal axis ω in Figure 3-1 and Figure 3-2 is scaled with the sampling period T and T' respectively.

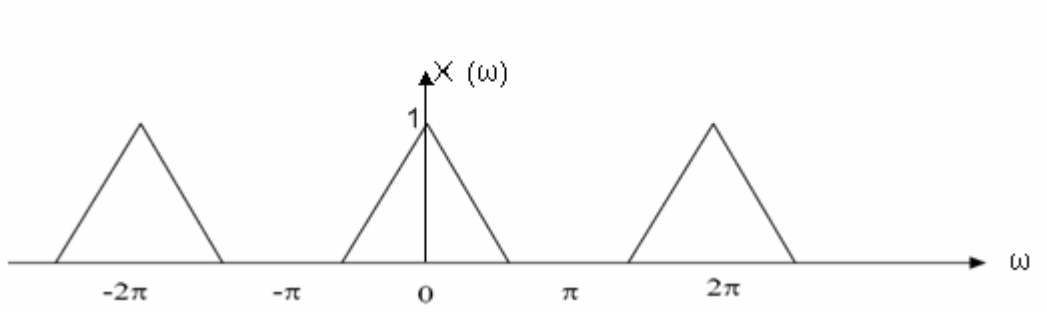

Figure 3-1 : $X(\omega)$, frequency spectrum of the $x[n]$ signal

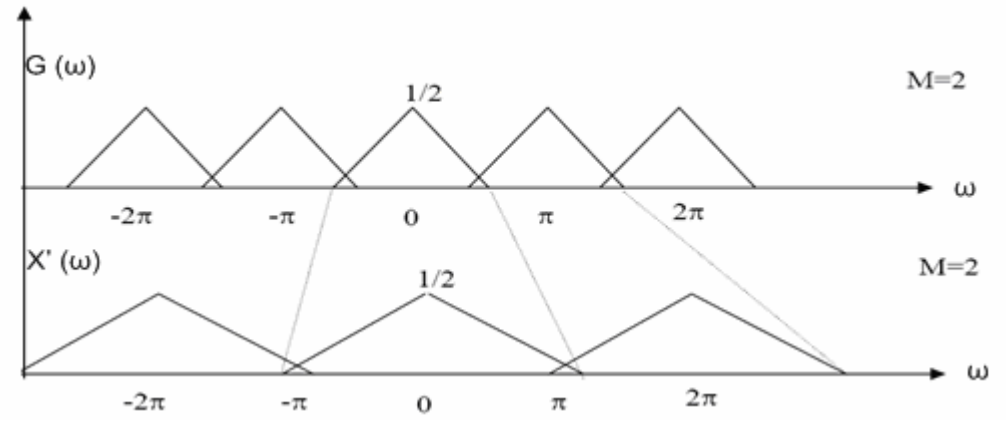

Figure 3-2 : $G(\omega)$, frequency spectrum of the intermediate signal $g[n]$ and $X'(\omega)$, frequency spectrum of the decimated signal $x'[n]$ for $M = 2$

As it can be seen from Figure 3-2 the spectra of the intermediate signal, consists of M replicas of the input signal spectrum in the interval $[-\pi, \pi]$, and the frequency spectrum of the decimated signal is expansion of the frequency axis of the intermediate signal spectrum. If the bandwidth of the input signal is greater than $1/2M$, then aliasing is observed at the decimated signal. To prevent aliasing at the decimated signal, either the input signal $x[n]$ which was sampled from $x_c(t)$ must have been over sampled by a factor of M (M times the Nyquist rate), or a digital low pass filter with cut-off frequency of $1/2M$ must be applied to $x[n]$ prior to decimation. Frequency spectrum of the decimated signal (with antialias filtering) is illustrated in Figure 3-3. It must be noted that horizontal axis ω in Figure 3-3-a and Figure 3-3-b is scaled with the sampling period T and T' respectively.

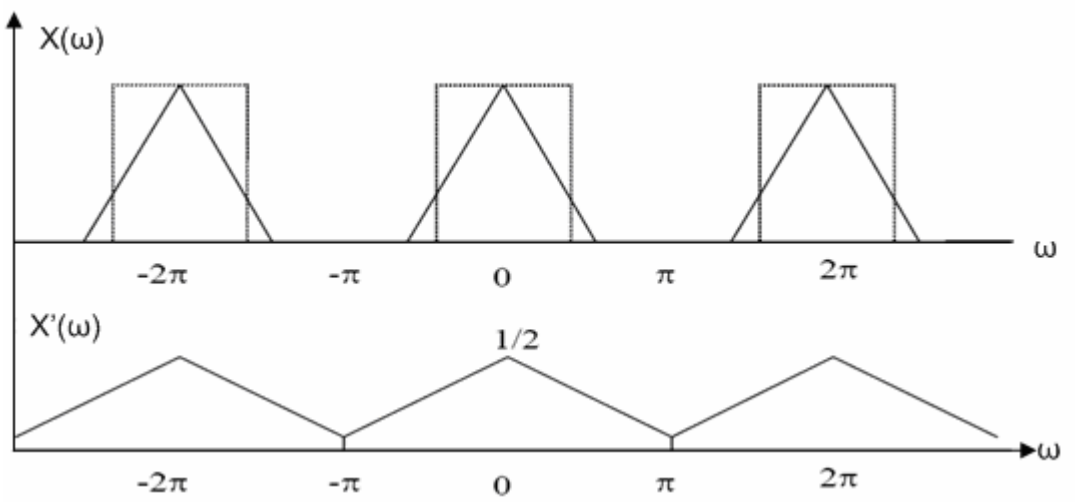

Figure 3-3 : a) $X(\omega)$,frequency spectrum of the input signal $x[n]$ (Dotted lines represent the antialising low pass filter) and $X'(\omega)$, frequency spectrum of the decimated signal $x'[n]$.

3.1.2 Increasing Sampling Rate by an Integer Factor

Sampling rate of the discrete time signal $x[n]$ in Eq. (3.1), with sampling period of T, can be increased by a factor of L to obtain $x'[n] = x_c[nT']$ where $T' = T/L$. This operation is referred as interpolation and using discrete-time operations, the process can be performed in two steps.

1. Up-sampling by zero filling where $L-1$ zeros are filled between every sample of the input signal. The operation can be modeled as,

$$g(n) = \begin{cases} x[n/L], \text{ if } n=k.L \text{ where } k \text{ is an integer} \\ 0, \text{ otherwise} \end{cases} \quad (3.7)$$

2. Low pass filtering of the filled in signal.

To analyze the frequency spectrum of the filled-in signal $g[n]$, Fourier transform of the $g[n]$ signal can be written as

$$G(\omega) = X(\omega L) \qquad (3.8)$$

The frequency spectrum of the input signal and the filled in signal for $L = 2$, is illustrated in Figure 3-4. It must be noted that horizontal axis ω in Figure 3-4-a and Figure 3-4-b is scaled with the sampling period T and T' respectively.

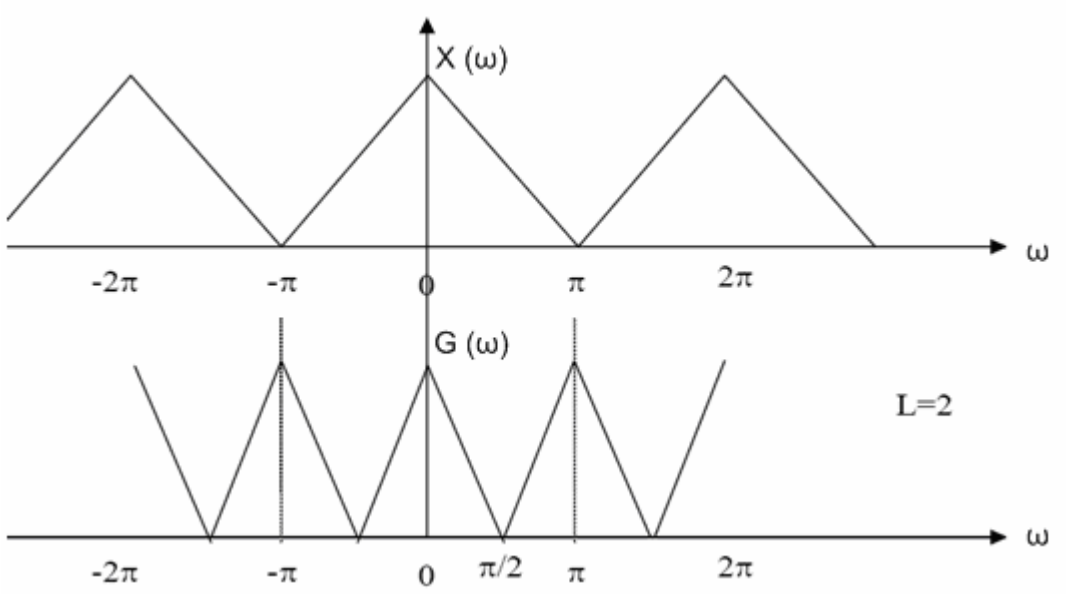

Figure 3-4 : $X[\omega]$, frequency spectrum of the input signal $x[n]$ and $G(\omega)$, frequency spectrum of the filled in signal $g[n]$ for $L = 2$

As it can be seen from Figure 3-4, the filled-in signal spectrum is a compressed version of the input signal spectrum by a factor of L. To remove the replications caused by zero filling, the signal must be filtered using the appropriate low pass filter with cut off frequency of $1/2L$.

3.1.3 SD to HD Sampling Rate Conversion

Spatial resolution up-conversion problem is basically the 2D version of the interpolation problem discussed in section 3.1.2 . In the case of SD to HD resolution up-conversion, the problem can be stated as “reverse process of the 2D decimation performed when down scaling the HD video to SD resolution”. In theory, it is possible to reconstruct the original HD signal, after decimation and interpolation process, however there are practical limitations.

1) To avoid antialising after decimation of the HD resolution signal to SD resolution signal, the HD signal must be over sampled by a factor of M times the Nyquist rate or it must be low pass filtered with $f_c = 1/2M$. In practice, it is not possible to over sample the original signal by a factor M times the Nyquist rate since video signal is not a band limited signal. Therefore to avoid anti-aliasing, it is common to pass the original signal through a low pass filter, which will remove high frequency components in the signal spectrum. It is not possible to recover these high frequency components in the interpolation process.

2) Both decimation and interpolation process requires low pass filtering, and the filtering operations mentioned in section 3.1.2 assume ideal low-pass filters available. However the impulse response of an ideal low pass filter is a *sinc* function and its implementation requires infinite number of input samples, which is not possible in practice. The impulse response of the *sinc* function is approximated by several methods, which are discussed in the following sections.

Figure 3-5 [3] illustrates decimation of a 1-D signal followed by interpolation. It must be noted that the horizontal axis ω is scaled with the sampling period. It is clearly seen that the high frequency part of the original signal spectrum is lost. HD - SD – HD conversion is similar to the steps involved in Figure 3-5, the difference being the video signals are 2D signals. The input signal at Figure 3-5a can be treated as the HD signal acquired from a camera, and Figure 3-5c can be treated as the down sampled SD signal which is broadcasted using an SD channel. The output signal at Figure 3-5e can be treated as the HD signal reconstructed from the broadcasted SD signal, using interpolation. Flat panel TV sets' spatial resolution up- conversion performance vary depending on the interpolation performed on the SD signal. Spatial resolution up-conversion problem had been solved in theory [5, 6] and approximations to the theory had been published in [7-10]. Several evaluations and implementations of these approximations are also available in [6-13]. More recently, more advanced methods on spatial resolution up-conversion problem had been published in [14-23] to improve performance on either subjective metrics, or mean squared error(MSE) performance. The conventional linear methods aim to approximate the ideal low pass filter behavior discussed previously, and try to find the optimal low-pass filter, while more advanced methods try recovering the missing spectral component[3].

3.2 Linear methods for up-scaling

Linear up-scaling is performed by convolving an interpolation kernel $h[n]$ with the sampled signal $x[n]$.

$$F(n) = \sum x(k)h(n-k) \qquad (3.9)$$

Here, $h[n]$ is basically the time domain impulse response of the low pass filter mentioned in section 3.1.2. For an ideal low pass filter, impulse response must be a *sinc* function, however it is already mentioned that such a filter can not be implemented at practice. Approximations to such a filter are obtained by constructing the convolution kernel based on piecewise polynomial functions.

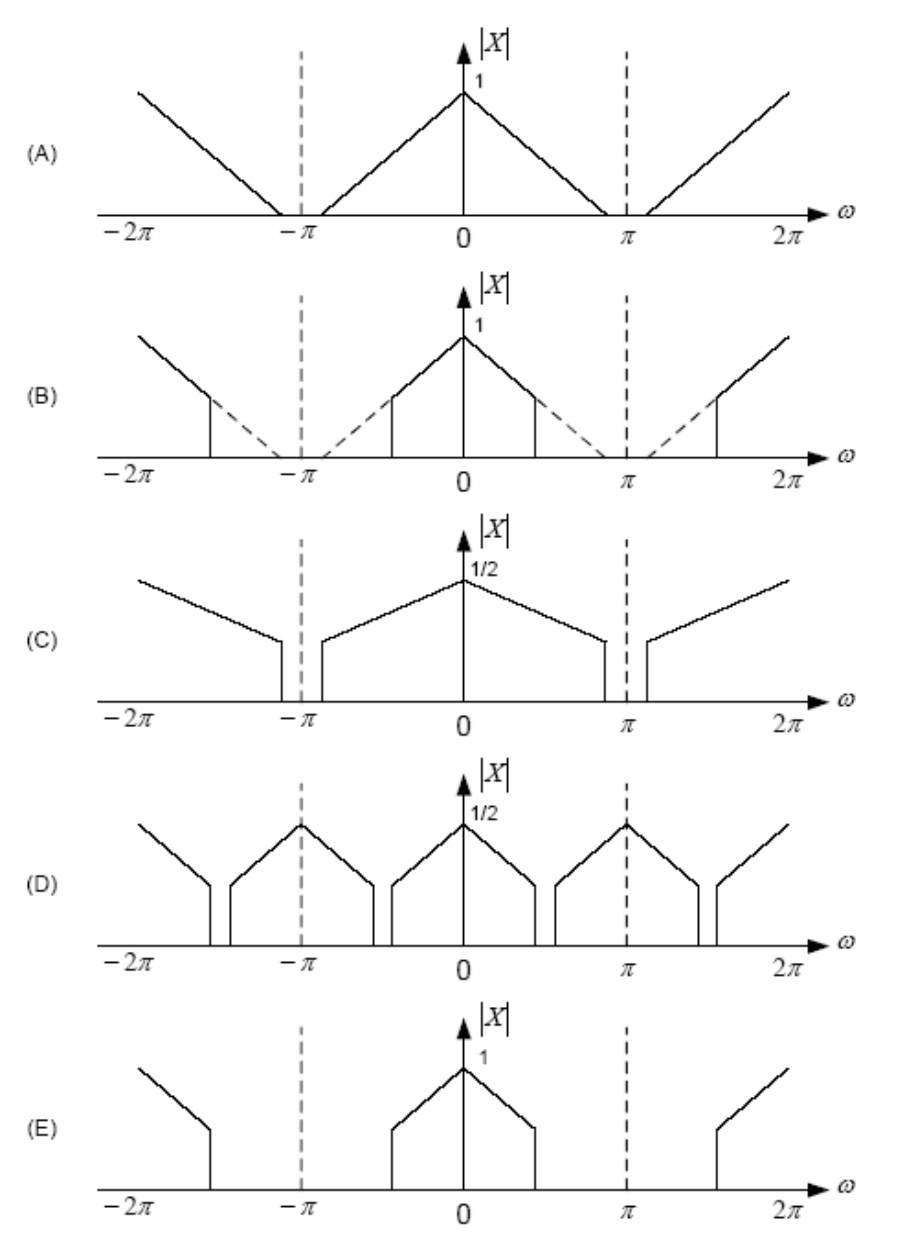

Figure 3-5: Frequency characteristics of decimation-interpolation process. Frequency spectrum of A) The original signal B) Low pass filtered signal C) Down sampled signal D) Up sampled signal E) Low pass filtered signal

3.2.1 Zero order hold interpolation

This is the simplest method of spatial up-scaling. The 1-D impulse response of the zero order hold interpolation kernel is given as

$$h[n] = \begin{cases} 1, -0.5 \le n \le L-1 \\ 0, otherwise \end{cases} \quad (3.10)$$

where L is the scaling ratio.

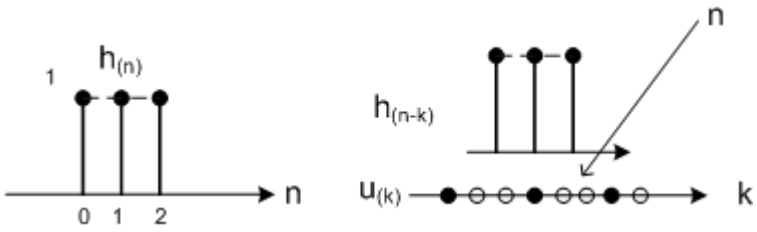

Figure 3-6: $h[n]$, the impulse response of the zero order hold interpolation function for $L = 3$, and illustration of convolution using this kernel

Impulse response of the zero order hold interpolation kernel is given in Figure 3-6**.** As it can be easily seen from Figure 3-6**,** the result of performing convolution using the zero older hold interpolation kernel with the SD signal, is the replication of the SD pixels at the output image. Therefore the method is also known as pixel replication. This method is rarely used in TV scalers due to its low image quality.

3.2.2 Linear Interpolation

Another low cost method is to use a first degree piecewise polynomial function, which is obtained by convolving the nearest neighborhood kernel with itself. 1-D impulse response of the linear interpolation kernel is given as;

$$h[n] = \begin{cases} \frac{L-|n|}{L}, if\ 0 \le |n| \le L-1 \\ 0, otherwise \end{cases} \quad (3.11)$$

Figure 3-7 illustrates the impulse response of the linear interpolation kernel for $L = 3$. As it can be seen from the convolution of the impulse response with the input signal, the interpolation process determines the missing signal value by taking the weighted linear average of the neighbor pixels, and the weights of neighbor pixels are proportional to the their distance to the interpolated pixel. For 2D applications,

linear interpolation is referred as bilinear interpolation, and despite its rather low image quality, it is widely used due to its low cost.

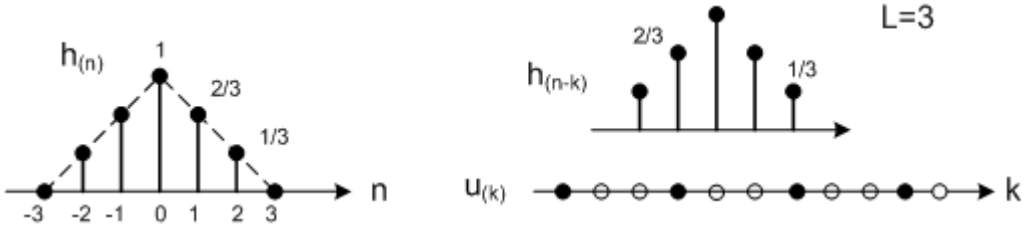

Figure 3-7: $h[n]$, impulse response of the linear interpolation kernel for $L = 3$ and illustration of convolution using this kernel

3.2.3 Cubic Interpolation

In cubic interpolation, the impulse response of the ideal low pass filter is approximated using three cubic polynomial pieces. The continuous interpolation kernel of cubic interpolation given in [7] is,

$$h(t) = \begin{cases} 3/2\,|t|^3 - 5/2\,|t|^2 + 1, & ,0 \le |t| \le 1 \\ -1/2\,|t|^3 + 5/2\,|t|^2 - 4\,|t| + 2 & ,1 \le |t| \le 2 \\ 0 & ,|t| \ge 2 \end{cases} \qquad (3.12)$$

Discrete time impulse response of the cubic convolution interpolation can be obtained sampling $h(t)$ with $4L+1$ samples in the interval -2 ≤ t ≤ 2. Figure 3-8 shows the discrete time impulse response sampled in this manner for $L = 2$.

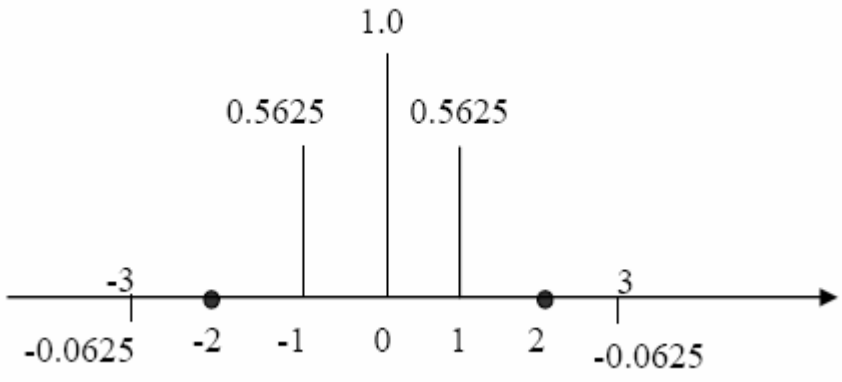

Figure 3-8: Impulse response of cubic interpolation function for $L = 2$

3.3 Advanced Methods for up scaling

In section 3.2, conventional linear up-conversion methods were described. The main idea behind these methods is to approximate the time domain impulse response of a *sinc* function, which serves as a low pass filter in frequency domain. As mentioned in section 3.1, even if the low pass filter used in the interpolation stage is ideal, there is still loss of high frequency components, since an antialising low pass filter is used prior to decimation. The main idea behind advanced methods is to recover the missing spectrum at the down sampled image.

3.3.1 Content Adaptive Training Based Methods

The idea behind training based content adaptive methods is to use prior information obtained from training with large sets of images in order to recover the missing spectrum at the down sampled image. Natural images are structured signals and they have considerably less variability than random signals [24]. Training based methods try to exploit the spatial characteristics of image signals prior to interpolation, using large sets of training images. This type of interpolation is performed in two steps, namely the training and the filtering steps.

Computationally intensive training step can be performed offline to reduce implementation complexity and cost. A large set of selected images are processed to extract statistical information about the spatial image characteristics. The information extracted consists of learned spatial structures, typically referred as context classes, and the way these structures are distorted during HD to SD conversion. Investigating the class specific distortions, one can design methods to restore degraded image components (typically high frequency components). Least Mean Square (LMS) method and Expectation Maximization (EM) algorithm are two commonly used techniques in the training stage.

Filtering step is performed on the fly, and typically consists of feature extraction, classification and interpolation stages. Appropriate type of filter kernel is selected at the end of the feature extraction and classification stages, and the selected kernel is used to interpolate the HD pixel from the SD pixel neighborhood.

Several training based interpolation methods are proposed in the literature. In [18] Atkins *et al* proposed a training based method, whose optimization method at the training step is based on expectation maximization algorithm. In [17], Kondo *et al* proposed a training based method where coefficients are obtained based on LMS criterion. In [25], Freeman *et al* proposed a method where HD image is synthesized

block by block, where the blocks are chosen from a database generated during the training step. These methods are briefly described in the following sections.

3.3.1.1 Digital Reality Creation (Kondo's method)

Kondo's method [17] is a training based, content adaptive interpolation method. The coefficients used during interpolation depend on the local content of the image. A classification is performed based on the pattern of the local neighborhood of the processed pixel, and interpolation is performed using the coefficients for the selected class. Filter coefficients are obtained by a training process performed offline. Figure 3-9 illustrates the training process performed. Training process uses the HD video and the SD video as the training material, and uses Least Mean Squares (LMS) criterion to obtain the optimal coefficients. Training process is computationally intensive; however it will not cause any trouble since it is performed only once and offline.

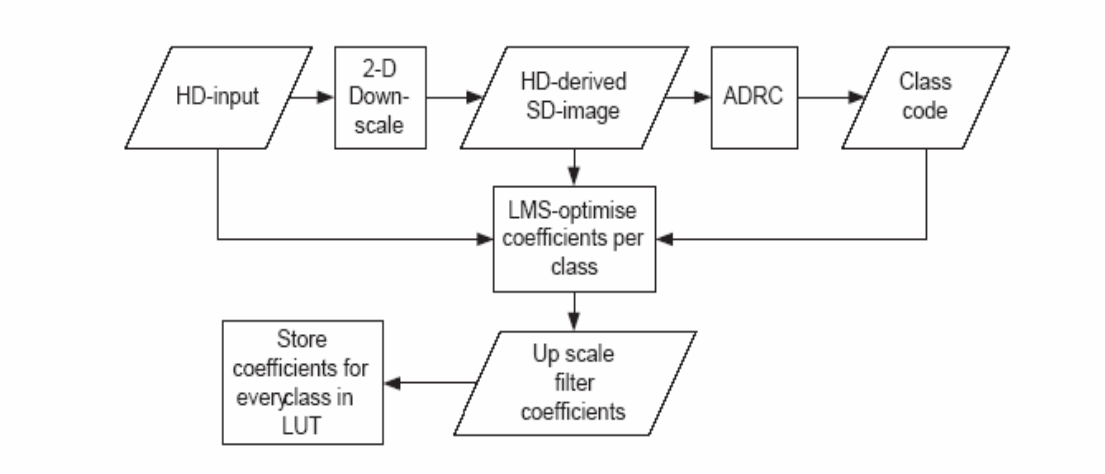

Figure 3-9: Training process performed in Kondo's method

Classification is performed using Adaptive Dynamic Range Coding (ADRC) [24] and equation (3.13) is used when encoding each pixel into 1 bit Q.

$$Q = \left\lfloor \frac{F_{SD} - F_{MIN}}{F_{MAX} - F_{MIN}} + 0.5 \right\rfloor, \qquad (3.13)$$

where F_{SD} is the SD pixel's luminance value, and F_{MAX} and F_{MIN} are the maximum and minimum luminance values around the 3X3 neighborhood of the centre pixel. In case no encoding was used, the number of classes for a 3X3 window would be $(2^8)^9$ and using equation (3.13), the number of classes is reduced to 2^9.

Figure 3-10 illustrates the aperture used in Kondo's method. Gray circles in the figure represent the input SD pixels, and the white circles represent the HD pixels produced. For a scaling factor of $L = 2$, SD pixel F_{11} will be replaced with 4 HD pixels denoted as A,B,C,D. Gray pixels F_{00} through F_{22} illustrate the 3X3 local neighborhood of F_{11}. The same local neighborhood of the centre pixel is used both in the classification and interpolation steps.

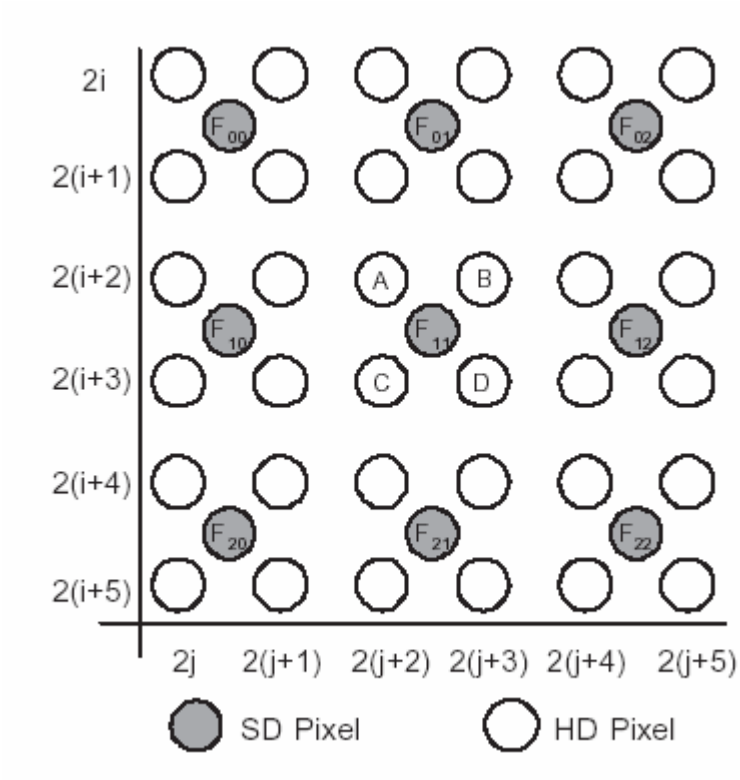

Figure 3-10: Aperture used in Kondo's method. The HD pixels A,B,C,D are interpolated using nine SD pixels, (F_{00} to F_{22})

The interpolation of the HD pixels, is performed using equation (3.14);

$$F_{HI}(2(i+2),2(j+2)) = \sum_{k=0}^{2}\sum_{l=0}^{2} w_{kl,c} F_{SD}(2(i+2k)+1),2(j+2l)+1) \qquad (3.14)$$

where $w_{kl,c}$ are the interpolation filter coefficients for class c.

3.3.1.2 Resolution Synthesis (Atkins' Method)

Atkins et al proposed a training based method, referred as Resolution Synthesis(RS), in [18]. The main idea behind RS is stated in [1] as "In a large training set, learn the high resolution image details that correspond to different spatial structures observed at low-resolution, such as edges of different orientations, uniform areas and texture regions, then use those learned relationships to identify and restore the details in other images." The approach is based on recognizing that pixels in natural images can be classified as belonging to a limited number of context classes. These classes are defined by pixel neighborhoods that are visually identifiable such as shown in Figure 3-11. If the size of the local window is 5x5, and each pixel luminance value is quantized at 8 bits, the number of possible patterns is 2^{8*25}, which is practically not possible to deal with.

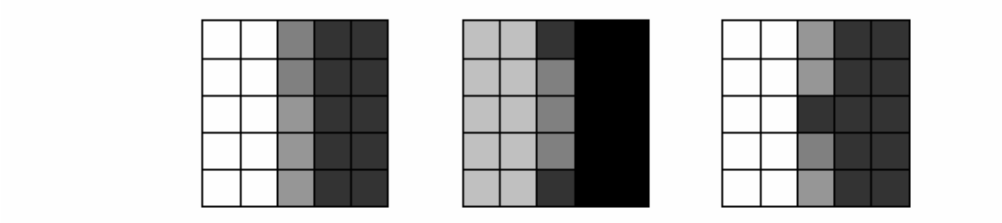

Figure 3-11: 5x5 pixel neighborhoods identified as a vertical edge

Fortunately, examination of the natural image signals show that number of meaningful spatial structures is limited, and a great rate of the possible patterns can be figured out as noise-like behavior. Even eliminating such noise-like patterns does not reduce the space to be explored; hence one can make use of the dominant spatial structure being the edge direction. If the optimum interpolation filters for the neighborhoods in Figure 3-11 are derived, they would possibly be very close[18]. So instead of assigning a context class to every pixel configuration, a context class can be assigned to large number of configurations with similar spatial structure. In resolution synthesis, number of context classes is fixed and is around 100. Structure of the RS algorithm predictor is depicted in[1] as in Figure 3-12.

As it is seen in the figure, the structures of Kondo's and Atkins' methods are similar, the difference being how classification blocks are implemented. While Kondo's method uses ADRC [24] in classification, Atkins' method uses Expectation Maximization algorithm [26].

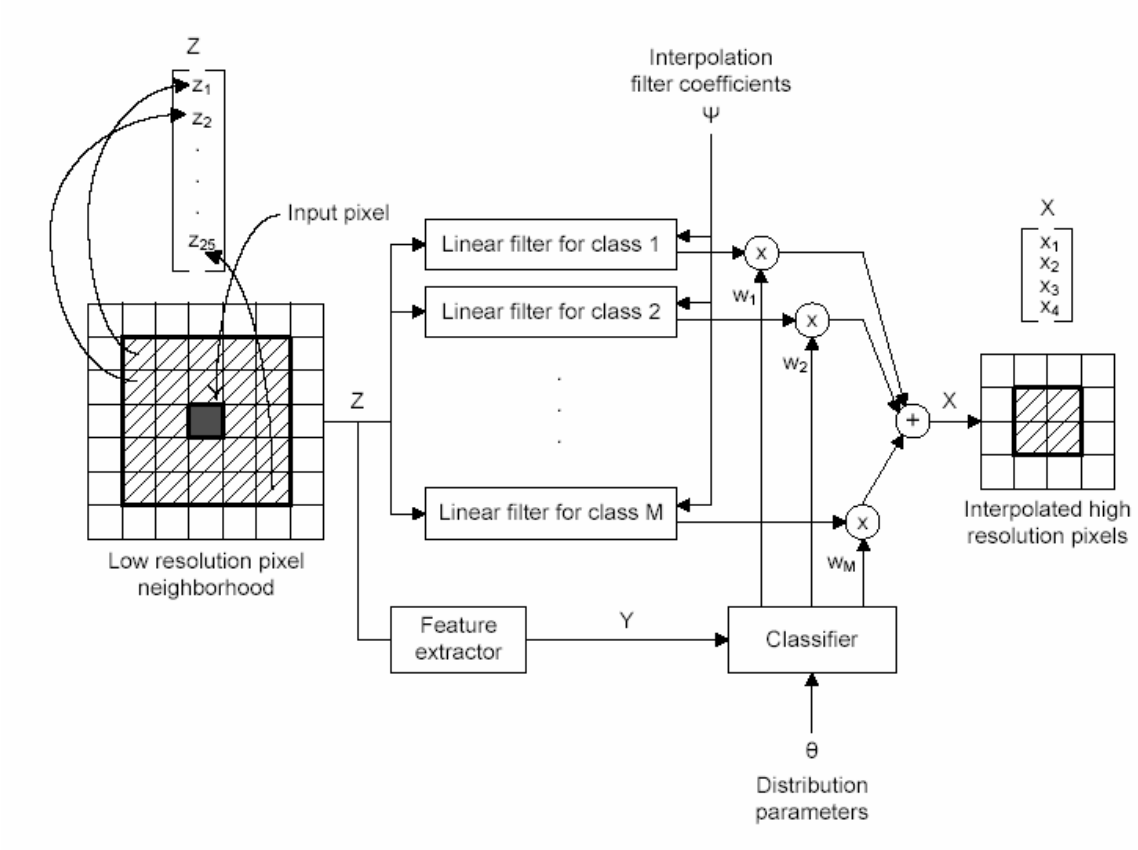

Figure 3-12: Structure of the RS predictor.

RS algorithm can be performed in 3 steps :

1) Feature Extraction : This is the first step of the algorithm to obtain the 8x1 classification vector $\vec{y}$ which is defined as :

$$\vec{y} = \begin{cases} \tilde{\vec{y}} \,\|\, \tilde{\vec{y}} \,\|^{-0.75} & , \tilde{\vec{y}} \neq 0 \\ 0 & , \text{otherwise} \end{cases} \qquad (3.15)$$

The $\tilde{\vec{y}}$ vector is an 8x1 vector constructed by the difference of the centre pixel with each neighbor pixel in the classification aperture in Figure 3-13 and $\vec{y}$ vector is the normalized version of the $\tilde{\vec{y}}$ vector, whose elements are given as :

$$\frac{\tilde{\vec{y}}_i}{\left[\sum_{j=1}^{8} \tilde{\vec{y}}^{\,2}\right]^{p}} \qquad (3.16)$$

where p is a parameter to control the amount of normalization. Feature vectors are normalized to unit length when p is selected as 1 and no normalization is performed when $p = 0$. In [18], p value is obtained as 0.75 experimentally.

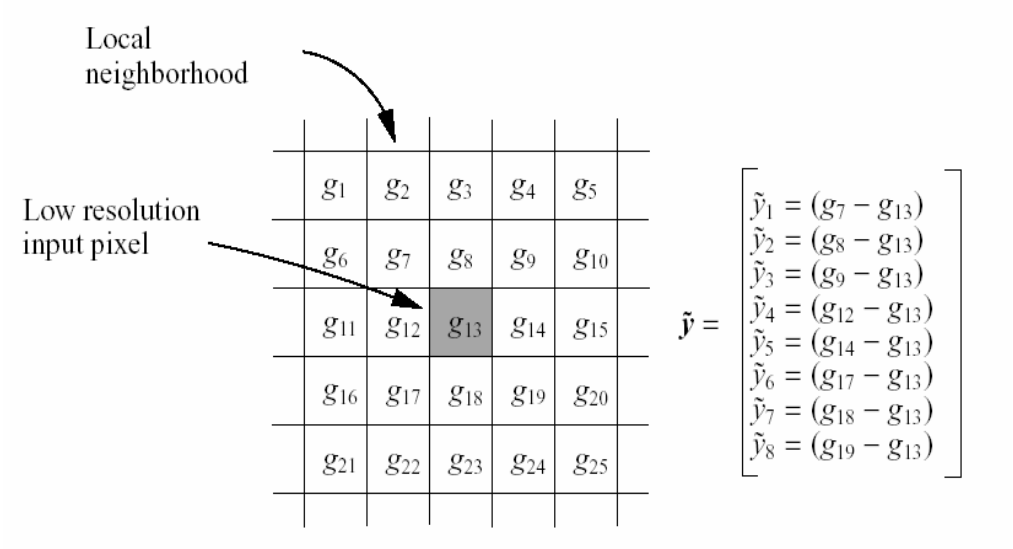

Figure 3-13: Extraction of the $\tilde{\vec{y}}$ vector

2) Context classification : In this step, classification vector $\vec{y}$ is compared with the representative vector $\overrightarrow{\mathrm{RV}}_c$ of the c^{th} class to calculate $P(c \mid \vec{y})$, the probability that the current neighborhood belongs to class c, which is given as :

$$P(c \mid \vec{y}) = \frac{CW_c \exp(\frac{-\| \vec{y} - RV_c \|^2}{2VAR})}{\sum_{d=0}^{M} CW_d \exp(\frac{-\| \vec{y} - RV_d \|^2}{2VAR})} \qquad (3.17)$$

Representative vector, $\overrightarrow{RV}$, represents the class, class weight, CW, indicates the global probability of the class, and the variance, VAR, indicates the average distance of the representative vector and the mean of the representative vector. These parameters are obtained for each class using EM algorithm at training stage.

3) Filtering : High resolution pixels are calculated by first filtering the low resolution block with filters from each class, and then taking the linear weighted average of the filter outputs, where weighting coefficient of each filter is its corresponding $P(c \mid \vec{y})$ value. The same aperture in Kondo's method is used for Atkins' method. From the definition, high resolution pixel A is calculated as

$$F_{HI}(2(i+2),2(j+2)) = \sum_{c=0}^{M-1} \sum_{k=0}^{2} \sum_{l=0}^{2} (a_{c,kl} F_{SD}(2(i+2k)+1,2(j+2l)+1) + b_{c,ij}) P(c \mid \vec{y}) \qquad (3.18)$$

Where a is a 4x9 matrix and b is a 4x1 matrix of class c , each row corresponding to coefficients to interpolate high resolution pixels A,B,C,D.

3.3.1.3 Example Based Super Resolution

Example based super resolution is different from Kondo's and Atkins' method both in the classification and reconstruction step. In example based super resolution, the input image is decomposed into low frequency and high frequency components. Low frequency image is obtained by first blurring and sub sampling the original high resolution image and then scaling back this image to its original size using a linear interpolation method. By this way, an image of desired size that lacks high resolution detail is obtained. In the training set, the differences between the high resolution image and linear interpolated image are stored. High resolution patches corresponding to every possible low-resolution image patch, typically of 5x5 and 7x7 pixels, are stored. The detail component of the image is obtained by combining the high resolution patches. The prior information obtained at the training step provides statistical information on which low resolution patch typically corresponds to which high resolution patch, and this information is used to reconstruct the high resolution details of the image. For its constraints on huge storage of patches, cost of the method is high for real time implementation.

3.3.2 Directional Interpolation Methods

New Edge Directed Interpolation (NEDI) proposed in [14] is the typical sample of a directional interpolation method. NEDI aims to interpolate along edges rather than across them to prevent blurring. NEDI differs from Kondo's and Atkins' method in the place coefficient optimization is performed. The optimization of the interpolation coefficients is performed by applying an LMS algorithm on-the-fly. The main advantage on performance is that the local neighborhood does not need any simplification. The disadvantage is that more calculations are required for optimization since the original image is not available. Figure 3-14 shows the aperture used in NEDI.

NEDI algorithm uses the following formulae for interpolation.

$$F_{HI}(2(i+1),2(j+1)) = \sum_{k=0}^{1}\sum_{l=0}^{1} w_{2k+l} F_{SD}(2(i+2k),2(j+2l)) \qquad (3.19)$$

where w coefficients in the formulae are computed at the runtime using LMS criterion. The sum of squared errors over a set S in the optimization is written as

the sum of squared differences between the original SD pixels and interpolated SD pixels. Sum of squared error(SSE) is given as :

$$SSE = \sum_{i,j} (F_{SD}(2i+2,2j+2) - F_{HI}(2i+2,2j+2))^2 \qquad (3.20)$$

Substituting F_{HI} in equation (3.20) and writing the equation in matrix form, SSE is written as :

$$SSE = \| \vec{y} - \vec{w}C \|^2 \qquad (3.21)$$

where $\vec{y}$ is the vector of SD pixels in S, and C is a $4 \times S^2$ matrix whose k^{th} row contains 4 diagonal SD neighbors of the k^{th} SD pixel in $\vec{y}$ vector. To find the minimum SSE, the derivative of SSE over $\vec{w}$ must be 0 hence $\vec{w}$ is obtained as :

$$\vec{w} = (C^T C)^{-1}(C^T \vec{y}) \qquad (3.22)$$

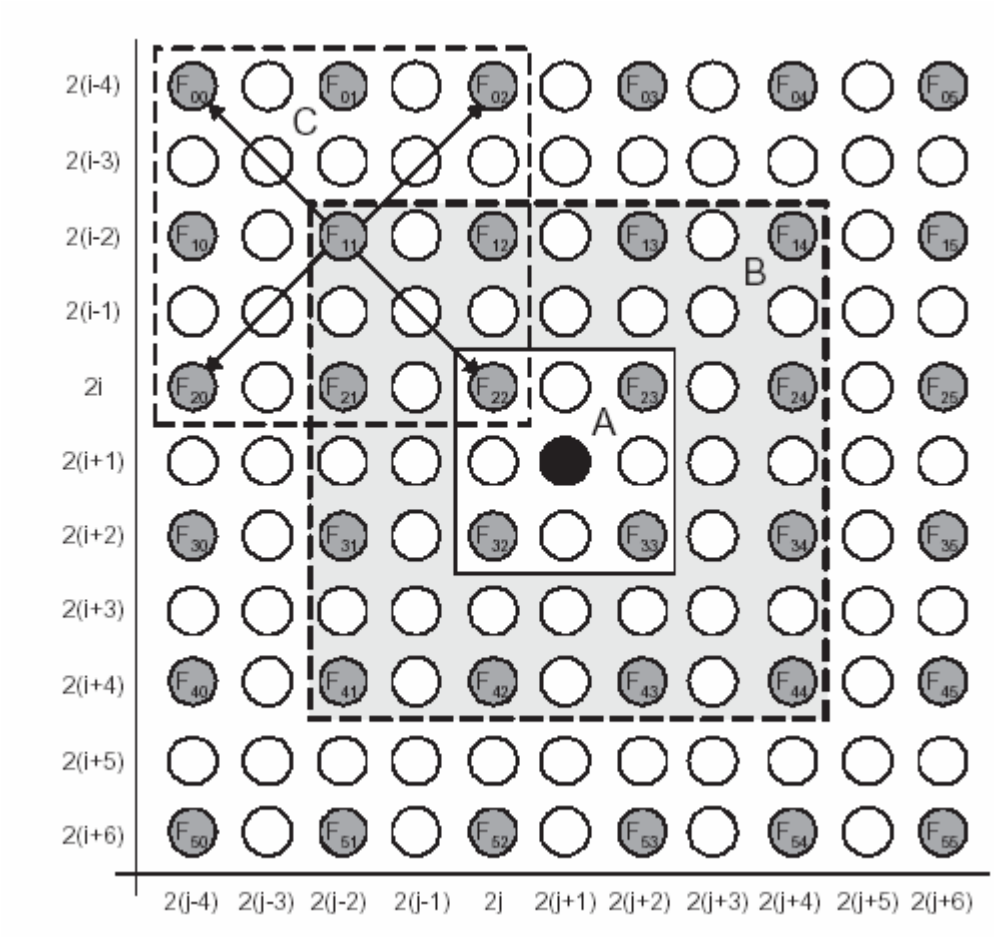

Figure 3-14: Aperture used in NEDI

The computation of the coefficient matrix used at the interpolation step is more costly than the interpolation step itself, which brings a major disadvantage for the real-time implementation of the algorithm.

3.3.3 Other Methods

Other than the training based methods, and the directional interpolation based methods, there exists other advanced algorithms proposed for video up-scaling problem. In [19-21] image interpolation is performed using neural networks. In [16], frequency domain operations are used. In [22], SD image is up scaled using a linear scaling algorithm and a non linear sharpening method, Luminance Transient Improvement (LTI) [27] is applied to the scaled image, to recover the high frequency components lost during the decimation process. There are several other methods which are either similar to or derived from the methods discussed in previous sections. These are out of scope for this study, hence are not mentioned in this section.

3.4 Video Performance Evaluation of the Methods

Evaluation of video enhancement algorithms is quite tricky since the aim of video enhancement algorithms is to improve the perceived image quality which is subjective. After all, the target is the human eye, and the success of the algorithm might differ with the user's taste. There exist subjective evaluation techniques which usually make use of a group of trained and non-trained eyes, to evaluate an algorithm's performance; however such processes are expensive, slow and hard to manage. Several image quality metrics have been proposed in the literature which aims to correlate well with perceived image quality. Though not being closely correlated with perceived image quality, some simple metrics like mean square error (MSE), or Peak Signal to Noise Ratio (PSNR) are also widely used to evaluate an algorithm's performance. In many of the scientific researches, MSE is taken as the major metric to compare the algorithms' performance. In [3], the algorithms discussed in previous sections (Example based super resolution in [25] being the exception) are also compared and evaluated using MSE criterion and subjective tests. In this evaluation, four still images and seven video sequences were used, with several spatial characteristics. In objective evaluation, MSE criterion is used with with the formulae:

$$MSE = \frac{1}{N}\sum_{i,j}(F(i,j) - G(i,j))^2 \qquad (3.23)$$

where $F(i, j)$ is the luminance value of the i^{th} row, j^{th} column in the original high resolution image, and $G(i, j)$ is the luminance value of the same spatial location in the up converted image. N is the number of pixels in the images. Table 3-1 shows the results of objective evaluation, performed in [3].

Table 3-1 : MSE performance evaluation performed in [3]

	Bilinear Int.	cubic Int. Keys kernel	cubic int. Mitchell Netravali kernel	Kondo's method	Atkins' met.	Plaziac's met.	Li's met.	Greenspan's method	Tegenbosch's met.
Average MSE Score	120.2	101.2	111.4	92.3	95.0	104.2	117.0	101.4	203.3

Figure 3-15 illustrates the results of subjective evaluation performed in [3].

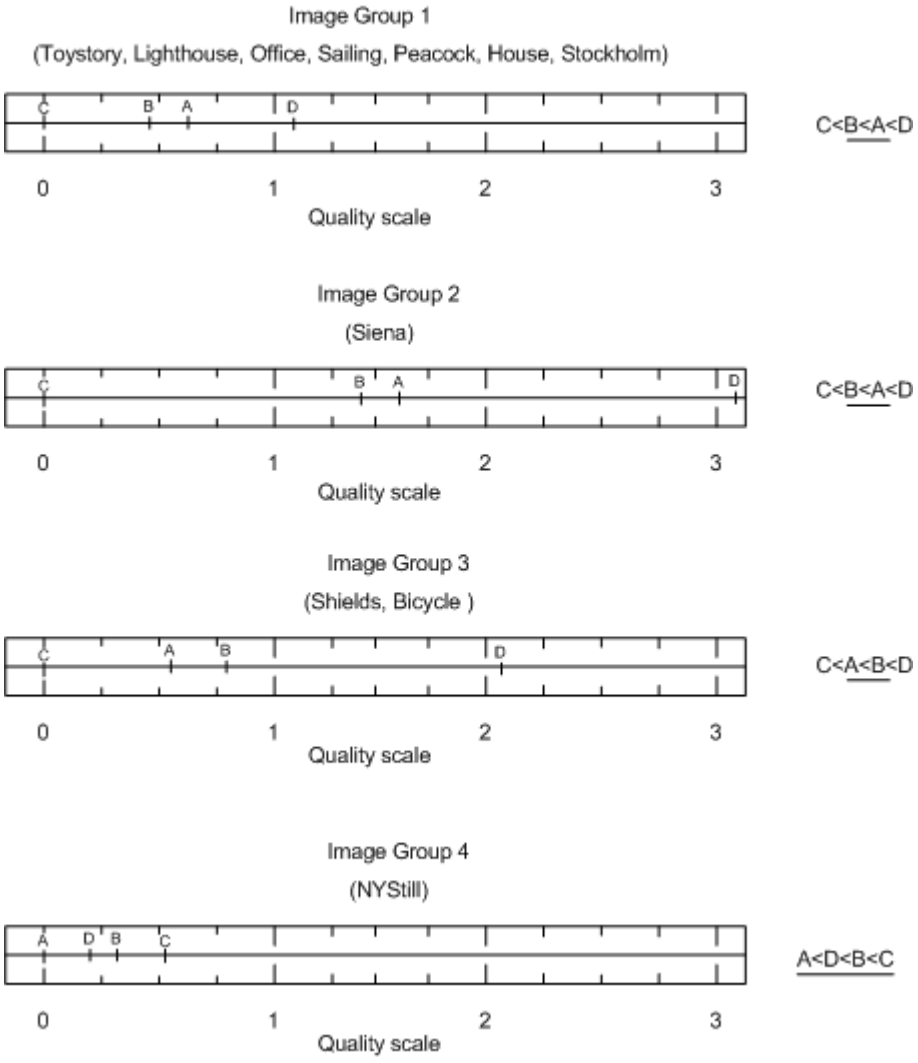

Figure 3-15 : Subjective evaluation performed in [3] (A : Cubic B-spline interpolation, B: Kondo's method, C: Li's method, D: Tegenbosch's method)

The subjective comparison is performed for four of the algorithms. In the figure, higher value horizontal scale represents higher perceived image quality. In the subjective evaluation, it is observed that optimal MSE score does not guarantee the best perceived image quality, since Tegenbosch's method [22], which gives poor MSE scores can be found to have a better perceived image quality by user. However it is also stated that MSE is the most appropriate start point for further enhancement in the perceived image quality. It must be also noted that Tegenbosch's method performs LTI over a bicubic based linear interpolation method, and further research on applying appropriate LTI over a training based interpolation may have the potential to get better perceived image quality as well as MSE score. More details on both subjective and objective evaluation of the algorithms, including the test pictures for each image group can be found in [28].

4. MODIFIED RESOLUTION SYNTHESIS

4.1 Overview

In chapter 3, it has been shown that scaling an image by linear filtering can not bring back the high frequency components degraded (reduced to noise level, completely filtered out or aliased) during sampling. This is where advanced resolution up-conversion techniques (which are also referred as resolution enhancement techniques) differ from linear scaling techniques. Resolution enhancement techniques can recover the missing or aliased high frequency components to a limited extent by estimating the missing high frequency components through spatially adaptive filtering and use of prior information. The main improvement offered by single frame resolution enhancement is observed around edges and textured areas. Compared to the results obtained by linear scaling filters such as bicubic interpolation combined with unsharpen filtering, techniques such as the resolution synthesis [18] algorithm can offer much smoother, continuous edges with sharp transitions, remove the blurry look from textured areas and rectify slight aliasing artifacts (where aliased signal components can not disturb the dominant spatial structure). Several content adaptive methods were considered to be a start point for the work done in this thesis, and resolution synthesis [18] was taken as the start point since it had performed well in terms of MSE and perceptive image quality. The details of the algorithm had been given in section 3.3.1.2.

In its current form, resolution synthesis is computationally too demanding for systems with limited computational resources and memory. The high computational load is mainly due to the large number of classes required for satisfactory performance (typically anywhere between 30-100) and the requirement for weighted combination (soft filtering) in Eq. (3.18). Linear combination is especially demanding since it requires repeating application of a 5×5 filter, implying 25 additional multiplications and an additional accumulation for every class included in soft filtering. In addition, the combination weights ($P(c \mid \vec{y})$'s in equation (3.17) must be computed to obtain the final result. In [1] several trials were performed to reduce the complexity of RS. It is observed that directly reducing the number of classes (below ~30) severely degrades performance. Also using only one class (the class with maximum membership) to compute the high resolution pixels results in degraded performance. It is found out that the discrimination power of the feature vectors

defined by equation (3.15) was severely degraded as the number of context classes was reduced below ~ 25. These shortcomings render resolution synthesis useless for customer grade flat panel displays, where the computational complexity must be kept below some threshold. The goal in [1] is to introduce some modifications so that RS can operate satisfactorily with as low as 5 context classes using hard decision (using a single class in filtering).

4.2 Algorithm Description

Modified RS scheme consists of two phases, namely the offline training phase, and the online filtering phase.

4.2.1 Offline Training Phase

Proposed training method in [1] is shown in Figure 4-1.It is based on the observation that interpolation filter design stage has direct access to the high resolution pixels. If one can couple interpolation filters to the feature extraction and classification stages, the resulting clustering should improve. Given the low and high resolution training images, proposed method iteratively extracts the best interpolation filters and the context class prototypes that are used to determine input pixel's context.

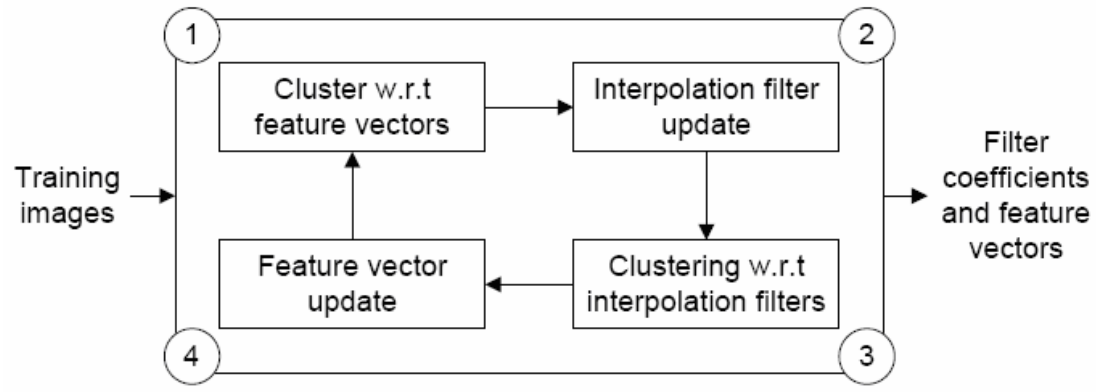

Figure 4-1: Proposed training scheme in [1]

The iterative training works as follows.

0.Initialization :

After extracting the feature vectors of all the low resolution pixels in the training set, class prototypes are initialized randomly. The prototype for class number 1 is manually set to a vector of all zeros. This guarantees that a class is reserved for uniform areas. All covariance matrices are set to identity matrices.

1. Clustering with respect to features:

After initialization, the low resolution pixels are classified with respect to their feature vectors. This is done by going through all low resolution pixels, computing the weighted Euclidian distance (the weighting matrix is the inverse of feature covariance matrix) between the pixel's feature vector, which is a representative of the local image characteristic of the low resolution pixel and the cluster prototypes, which are representatives of different context classes. Then the input low resolution pixel is labeled with the index of the cluster whose feature vector is the closest to the low resolution pixel's feature vector.

2. Filter update:

Once the low resolution pixels are clustered with respect to their feature vectors (context) the interpolation filters for all clusters are updated with the filter that minimizes the mean squared error between the interpolated and the true high resolution pixels computed for all low resolution pixels in a specific cluster. While preparing the training samples, a small amount of blurring prior to down sampling is necessary to model the camera response and also to avoid aliasing. But completely filtering out the high frequency components effectively creates an inverse problem where the filters are asked to bring back completely removed signal components (this is only possible in multi-frame case), resulting in bad filters.

3. Clustering with respect to filters :

After filter update, all the input pixels are clustered with respect to the minimum mean-squared-error interpolation filter. This is accomplished by going through all low-resolution training pixels, computing the interpolated high resolution pixels one by one, and comparing the interpolated pixels to the available high resolution pixels. The low resolution pixel is then labeled with the index of the interpolation filter that gives the minimum mean-squared-error between the interpolated and real high resolution pixels.

4. Class prototype update :

Once all the input pixels are classified, the feature vectors of the obtained clusters are updated one by one. This update can be done in various different ways such as taking the average of the median of the feature vectors. Class covariance matrices are updated next. To reduce computational complexity, diagonal covariance matrices are assumed. Then the algorithm steps back to step 1, clustering with respect to features, and iterate in this fashion for predetermined times. In [1] the experiments are done with 2 iterations.

4.2.2 Online Content Adaptive Filtering Phase

At the end of offline training phase, for each class i ; an 8x1 representative vector $\vec{C}_i$, each of which represents a class, an 8x1 variance set, σ_i, which indicates the average distance of each representative vector and the mean of the representative vectors, and four 5x5 interpolation filter kernels $K_{i00}, K_{i01}, K_{i10}, K_{i11}$ are calculated.

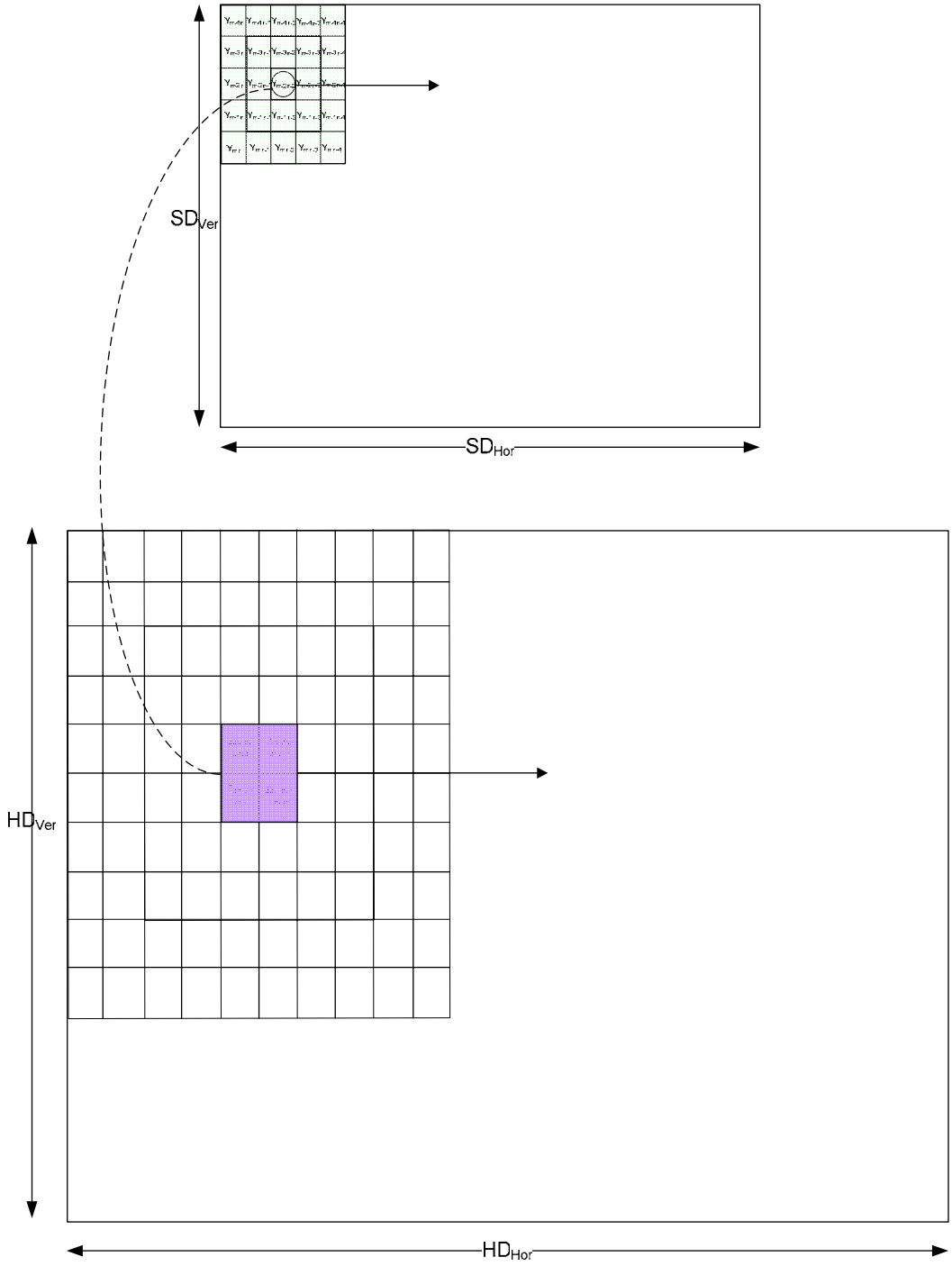

Figure 4-2 : Up conversion of SD resolution image to HD resolution image

Once the offline training phase is completed and $\vec{C}_i$, σ_i and $K_{i00}, K_{i01}, K_{i10}, K_{i11}$ values for each class is computed, online computations can be started to up-convert an SD resolution image to an HD resolution image as shown in Figure 4-2.

The inputs to the algorithm are

- the image matrix Y of size $SD_{Hor} xSD_{Ver}$, with each element Y_{mn} of the matrix representing the luminance value of m^{th} row, n^{th} column of the low resolution image.
- representative vectors $\vec{C}_i$ of size 8x1 for each class i.
- set of normalization constants σ_i of size 8x1 for each class i.
- Four 5x5 filter kernels $K_{i00}, K_{i01}, K_{i10}, K_{i11}$ for each class i.

The output of the algorithm is

- The $HD_{Ver} xHD_{Hor}$ image matrix Z with each element Z_{mn} of the matrix representing the luminance value of m^{th} row, n^{th} column of the high resolution image.

The algorithm consists of three stages, namely, feature extraction, context classification, and filtering. Feature extraction and context classification is performed for every pixel of the input SD image, and filtering is performed for every pixel of the output HD image. The aperture used in these stages, is shown in Figure 4-3

$Y_{m-4,n}$	$Y_{m-4,n-1}$	$Y_{m-4,n-2}$	$Y_{m-4,n-3}$	$Y_{m-4,n-4}$
$Y_{m-3,n}$	$Y_{m-3,n-1}$	$Y_{m-3,n-2}$	$Y_{m-3,n-3}$	$Y_{m-3,n-4}$
$Y_{m-2,n}$	$Y_{m-2,n-1}$	$Y_{m-2,n-2}$	$Y_{m-2,n-3}$	$Y_{m-2,n-4}$
$Y_{m-1,n}$	$Y_{m-1,n-1}$	$Y_{m-1,n-2}$	$Y_{m-1,n-3}$	$Y_{m-1,n-4}$
$Y_{m,n}$	$Y_{m,n-1}$	$Y_{m,n-2}$	$Y_{m,n-3}$	$Y_{m,n-4}$

Figure 4-3: Aperture used in Modified Resolution Synthesis

4.2.2.1 Feature Extraction

In the feature extraction stage, for every SD input pixel, the feature vector of the pixel is extracted from a 3×3 local neighborhood. The feature vectors will be denoted as ϕ. To obtain the feature vector, first, an 8x1 vector is obtained by subtracting the centre pixel value $Y_{m-2,n-2}$ from its 8 neighbors and taking the 4th power of the difference. The elements of the 8x1 feature vector are given as,

$$
\begin{aligned}
FV_1 &= (Y_{m-2,n-2} - Y_{m-3,n-1})^4 \\
FV_2 &= (Y_{m-2,n-2} - Y_{m-3,n-2})^4 \\
FV_3 &= (Y_{m-2,n-2} - Y_{m-3,n-3})^4 \\
FV_4 &= (Y_{m-2,n-2} - Y_{m-2,n-1})^4 \\
FV_5 &= (Y_{m-2,n-2} - Y_{m-2,n-3})^4 \\
FV_6 &= (Y_{m-2,n-2} - Y_{m-1,n-1})^4 \\
FV_7 &= (Y_{m-2,n-2} - Y_{m-1,n-2})^4 \\
FV_8 &= (Y_{m-2,n-2} - Y_{m-1,n-3})^4
\end{aligned}
\qquad (4.1)
$$

Then the resultant vector FV is normalized obtain the normalized feature vector ϕ.The following equation shows the mapping used to obtain each element of the normalized feature vector ϕ

$$
\phi_i = \frac{FV_i}{(\sum_{j=1}^{8} FV_j{}^2)^{0.75}} \qquad (4.2)
$$

4.2.2.2 Context classification

In the context classification stage, the feature vector ϕ obtained in the previous step is compared to five predetermined $\vec{C}_i$ vectors namely the representative vectors of each class. The distance between the feature vector ϕ and the representative vector of class i, $\vec{C}_i$ is computed as

$$
d_i = \| \phi - \vec{C}_i \| = \sum_{j=1}^{8} \frac{(\phi_j - C_{i,j})^2}{\sigma_{i,j}} \qquad i = 1,2,3,4,5 \qquad (4.3)
$$

That is, the distance between ϕ and $\vec{C}_i$ is the sum of squared differences between the corresponding entries of each vector divided by normalizing constants. The context class that is closest to the ϕ vector (with minimum d_i) is declared as the context of the current low resolution pixel $Y_{m-2,n-2}$, and its index cl is passed to the

interpolation stage. It must be noted that the $\vec{C}_i$ vectors and the σ_i normalizing constants are different for every class and they are pre-computed in the training phase.

4.2.2.3 Filtering

In the filtering stage, index passed by the context classification step is used to pick the interpolation filter corresponding to the context class of the current low resolution pixel. Interpolation is performed for every HD pixel, therefore the iteration is not performed in the SD input image, but on the HD output image to be created. For every output pixel at the HD output image, first the corresponding vertical and horizontal SD image coordinates y_v and y_h , are computed using the following formula :

$$y_v = \left\lfloor \frac{z_v}{L} \right\rfloor, y_h = \left\lfloor \frac{z_h}{L} \right\rfloor \quad (4.4)$$

where, z_v and z_h are the horizontal and vertical coordinates of the HD pixel and L is the scaling ratio. Then the horizontal and vertical phases hp and vp of the output pixel is computed using the following formula:

$$hp = \left\lfloor Q_h x(\frac{z_h}{L} - y_h) + \varepsilon \right\rfloor \quad (4.5)$$

$$vp = \left\lfloor Q_v x(\frac{z_v}{L} - y_v) + \varepsilon \right\rfloor \quad (4.6)$$

where Q_v and Q_h are the vertical and horizontal quantization values set as 2 in [1].

Using the context class index cl , horizontal phase hp and vertical phase vp , 5x5 interpolation kernel $K_{cl,hp,vp}$ is selected for the output pixel, and the interpolation of the output pixel is performed by convolving the 5x5 neighborhood of the corresponding SD coordinates, with the selected kernel. For the output pixel coordinate of $Z_{2(m-2),2(n-2)}$ the corresponding SD input pixel will be $Y_{m-2,n-2}$,from Eq.(4.4) and hp and vp will be both zero from Eq.(4.5) and Eq.(4.6). The selected filter will be $K_{cl,00}$ (denoted as K in Eq.(4.7)) and the interpolation output will be computed by

$$Z_{2(m-2),2(n-2)} = \sum_{i=-2}^{2}\sum_{j=-2}^{2} Y_{m-2+i,n-2+j} K_{ij} \qquad (4.7)$$

Pseudo code of the online filtering phase of the modified resolution synthesis algorithm is given in Figure 4-4.

Inputs: $Y = (SD_{hor} x SD_{ver})$ input image matrix, $\vec{C}_c$ (8x1 representative vector for each class), σ_c (8x1 normalization constant set for each class), $K_{c00}, K_{c01}, K_{c10}, K_{c11}$ (four kernel of size 5x5 for each class), c = 1,2,3,4,5

Output: $Z = HD_{Ver} x HD_{Hor}$ output image matrix

for m from 2 to $SD_{ver} - 2$ **do**

for n from 2 to $SD_{hor} - 2$ **do**

$SFV := 0$

for i = -1 to 1 **do**

for j = -1 to 1 **do**

$$FV_{3(i+1)+(j+1)} = (Y_{m,n} - Y_{m+i,n+j})^4$$

$$SFV = SFV + \left(FV_{3(i+1)+(j+1)}\right)^2$$

$SFV = SFV^{0.75}$

for i = -1 to 1 **do**

for j = -1 to 1 **do**

$$\phi_{3(i+1)+(j+1)} = FV_{3(i+1)+(j+1)} / SFV$$

for c = 1 to 5 **do**

for k=1 to 8 **do**

$$d_c = d_c + \frac{(\phi_k - C_{c,k})^2}{\sigma_{c,k}}$$

$cl_{m,n}$ = index_of(Min (d_c)) c = 1,2,3,4,5

for s from 2 to $HD_{ver} - 4$ **do**

for t from 2 to $HD_{hor} - 4$ **do**

compute SD coordinates m,n corresponding to s,t

Select the class of the filter $cl_{m,n}$

Select the vertical and horizontal phase of the filter

Select the appropriate filter K using vp,hp, and $cl_{m,n}$

for x = -2 to 2 **do**

for y= -2 to 2 **do**

$$Z_{s,t} = Z_{s,t} + Y_{m-x,n-y} * K_{x,y}$$

Figure 4-4 : Pseudo code of the Modified RS algorithm

4.3 Visual Quality Results

Prior to the implementation of the algorithm, several subjective benchmarks were run on the algorithm, in Vestel Video Quality Lab. In industrial video and image processing applications, it is mainly the perceived image quality that is targeted hence MSE criterion is discarded. The images scaled with MRS algorithm were compared with images scaled using bicubic scaling algorithm. MRS algorithm performance was also compared with state of the art video processors. Figure 4-5 shows the input image with 200x200 resolution. The 400x400 output images scaled by MRS and bicubic scaling algorithms are shown in Figure 4-6 and Figure 4-7

Figure 4-5 : Input image with 200x200 resolution

Figure 4-6: MRS output image with 400x400 resolution

Figure 4-7 : Bicubic scaler output image with 400x400 resolution

4.4 Complexity Analysis

In this section, a hardware complexity analysis of the RS and MRS algorithms are given. In computer science, time and space complexity of an algorithm is usually given in terms of asymptotic complexity, and the common approach is to use the order of n notation (e.g., “Big O”, “Big Theta” etc notations). Asymptotic complexity analysis seeks to find the complexity of a problem, when the size of the problem goes to infinity. Such an analysis will not make any benefit in this work since the point of interest is the algorithms’ hardware complexity. Several metrics are proposed to estimate the complexity of an algorithm in terms of its hardware resources. In this study, the basic approach which is also used in [6] is used. The algorithms are compared in terms of number of multiplications, additions and memory elements required per output pixel.

4.4.1 Complexity of RS and Modified RS algorithms

4.4.1.1 Feature Extraction :

Feature extraction of the RS algorithm is given in Eq.(3.15). Discarding the control logic and the registers inserted for pipelining or delaying the operands, the

complexity of the feature extraction block in terms of adder, multiplier and memory cells can be estimated in four steps :

i) Computation of the $\tilde{\tilde{y}}$ vector

ii) Computation of || $\tilde{\tilde{y}}$ || from $\tilde{\tilde{y}}$

iii) Computation of || $\tilde{\tilde{y}}$ ||$^{-0.75}$

iv) Computation of $\tilde{\tilde{y}}$ || $\tilde{\tilde{y}}$ ||$^{-0.75}$

Computation of the $\tilde{\tilde{y}}$ vector in step *i* requires 8 additions. Computation of the norm of the $\tilde{\tilde{y}}$ vector in step *ii* requires 8 multiplications followed by 8 additions. Assuming a piecewise linear implementation of the constant exponentiation operation, step *iii* can be performed using a look up table of 2^k words where k is the size of the input operand. Finally step *iv* requires 8 multiplications to compute the normalized feature vector. At total, the complexity of the feature extraction block in RS can be given as

$$Complexity_{RS\_FE} = 16\ multiplications + 16\ additions + 2^k\ words\ memory \qquad (4.8)$$

Feature extraction module of the MRS algorithm given in Eq.(4.1) and Eq.(4.2) is similar to the one in RS algorithm and can be estimated in four steps :

i) Computation of the $\overrightarrow{FV}$ vector

ii) Computation of || $\overrightarrow{FV}$ || from $\overrightarrow{FV}$

iii) Computation of || $\overrightarrow{FV}$ ||$^{-0.75}$

iv) Computation of $\overrightarrow{FV}$ || $\overrightarrow{FV}$ ||$^{-0.75}$

Computation of $\overrightarrow{FV}$ differs from the RS algorithm, due to the additional squaring operations, which will require 16 multiplications. The rest of the steps use the same operations and hence have the same cost as in RS. At total, the complexity of feature extraction stage in modified RS can be given as

$$Complexity_{MRS\_FE} = 32\ multiplications + 16\ additions + 2^k\ words\ memory \qquad (4.9)$$

4.4.1.2 Context Classification

Context classification stage of the RS algorithm is given in Eq.(3.17). The expression in the nominator requires 8 additions for the 8x1 vector subtraction, 8 multiplications, 8 additions and 8 divisions for the vector norm computation, 2 multiplications for the division by a constant, and squaring operations, and at least $\log N$ multiplications for the exponentiation operation where N is exponent. In the denominator, the arithmetic operations mentioned for the nominator are done for M times and finally M additions and a division is required to obtain the $P(c \mid \vec{y})$ value for one class. To obtain the $P(c \mid \vec{y})$ values for M classes, one should only compute the nominator since the denominator will be the same for all classes. Therefore approximately, 2 M (16 additions + 10 Multiplications + log N multiplications + 8 divisions) + M additions + M divisions are required . Using basic assumptions (taking the complexity of a division operation and exponentiation operation 10 times the complexity of a multiplication), the complexity of the classification step can be estimated as

$$Complexity_{RS\_CL} \approx 33M \; additions + 66M \; multiplications \qquad (4.10)$$

Context classification stage of the MRS algorithm is given in Eq.(4.3). The expression in the nominator requires 8 additions and 8 multiplications. Divisions by a constant can be implemented by multiplication by the inverse of the constant hence the division operation requires 8 multiplications and the summation operation requires 7 additions at the final stage. Since the operations will be performed for every class, these operations will be performed M times. To find the minimum distance value, 5 class distances are compared with each other, which will require 4 comparators. Assuming the complexity of a comparator is the same with an adder, total complexity of the classification step of the MRS algorithm can be estimated as

$$Complexity_{MRS\_CL} \approx (16M - 1) \; additions + 16M \; multiplications \qquad (4.11)$$

4.4.1.3 Interpolation

Interpolation stage of the RS algorithm is given in Eq.(3.18) Soft filtering requires a 5x5 convolution for each class and a linear combination of these convolutions, therefore the number of multiplications required will be equal to 5*5* M for the convolutions and M for the linear combinations. Number of additions will be (5*5 –

1) * M for the convolutions and $M-1$ for the linear combinations. Total complexity of the interpolation stage of the RS algorithm can be estimated as

$$Complexity_{RS\_IN} \approx (25M-1)\ additions + 26M\ multiplications \quad (4.12)$$

Interpolation stage of the MRS algorithm is given in Eq.(4.7). Since hard interpolation is performed in MRS instead of the soft interpolation operation in RS, 5x5 convolution is performed only for the selected class. Total complexity of the interpolation stage of the MRS algorithm can be estimated as,

$$Complexity_{MRS\_IN} \approx 24\ additions + 25\ multiplications \quad (4.13)$$

4.4.2 Complexity Comparison of RS and MRS algorithm

Number of multiplications, additions and memory words required in RS and MRS algorithms are listed in Table 4-1. The aim of the MRS algorithm had been discussed in Section 4.1. MRS algorithm tries to reduce the complexity of the RS algorithm by

- Replacing soft interpolation with hard interpolation
- Reducing the number of classes, M .

without sacrificing the performance, due to several modifications at the training stage. The effect of replacing the soft interpolation with hard interpolation to the hardware cost is obvious in Table 4-1. The complexity of the interpolation stage is roughly reduced by a factor of M , and the complexity of the classification stage is roughly reduced by a factor of 2, by the modifications performed to replace soft interpolation with hard interpolation. In [1], another modification is done on the number of context classes. It is reported that RS algorithm requires at least ~30 context classes, and MRS algorithm requires 5 context classes to perform satisfactorily. Substituting $M = 30$ in the RS complexity equation, and $M = 5$ in the MRS complexity equation, RS will require ~2400 multiplications + 1800 additions per output pixel, whereas MRS will require 137 multiplications + 120 additions per output pixel.

Table 4-1 : Number of arithmetic operations in RS and MRS algorithms

	Multiplication	Addition / subtraction	Memory words
RS Feature Extraction	16	16	2^k
RS Classification	66M	33M	
RS interpolation	26M	25M-1	
RS Total	**92M+16**	**55M+16**	2^k
MRS Feature Extraction	32	16	2^k
MRS Classification	16M	16M-1	
MRS interpolation	25	24	
MRS Total	**16M+57**	**16M+40**	2^k

5. PROPOSED HARDWARE ARCHITECTURE

The aim of this study is to find an efficient architecture to implement a content adaptive video resolution up-conversion scheme in a low cost hardware that is targeted for a high-end flat panel display product. The target application is video standard conversion where the input is standard definition video and the output is high definition video. Modified RS scheme, presented in chapter 4, is used for interpolation. The computational complexity of the Modified RS scheme in terms of multiplications and adders is given in chapter 4. In this chapter, an efficient hardware architecture to implement Modified RS scheme is given. Since the hardware is targeted to work on a real time platform, a detailed analysis of the system requirements is necessary. Section 5.1 lists the performance requirements to be considered in the design of the hardware architecture. The details of the hardware architecture designed considering the requirements are given in Section 5.2.

5.1 Performance Requirements

5.1.1 Throughput Constraints

The constraint on the throughput arises from the input and output pixel rates of the video standard to be converted. In this study, the input video standard is NTSC 480p @60Hz, which has 480 lines of vertical resolution and 720 columns of horizontal resolution, and the output video standard is NTSC 720p @60Hz which has 720 lines of vertical resolution and 1280 columns of horizontal resolution. The details of the input and output video specifications are given in Electronic Industries Alliance(EIA) DTV resolution standards specification [28]

Figure 5-1 and Figure 5-2 illustrate the timing diagram for the input and output video standards. The throughput and the pixel clock frequencies can be derived from the video standards. Pixel clock period, T_{clk} can be written as,

$$T_{clk} = \frac{T_{frame}(Total\ time\ \ for\ one\ video\ \ frame)}{N_{cpf}(number\ of\ \ pixel\ clock\ cycles\ \ for\ one\ video\ \ frame)} \quad (5.1)$$

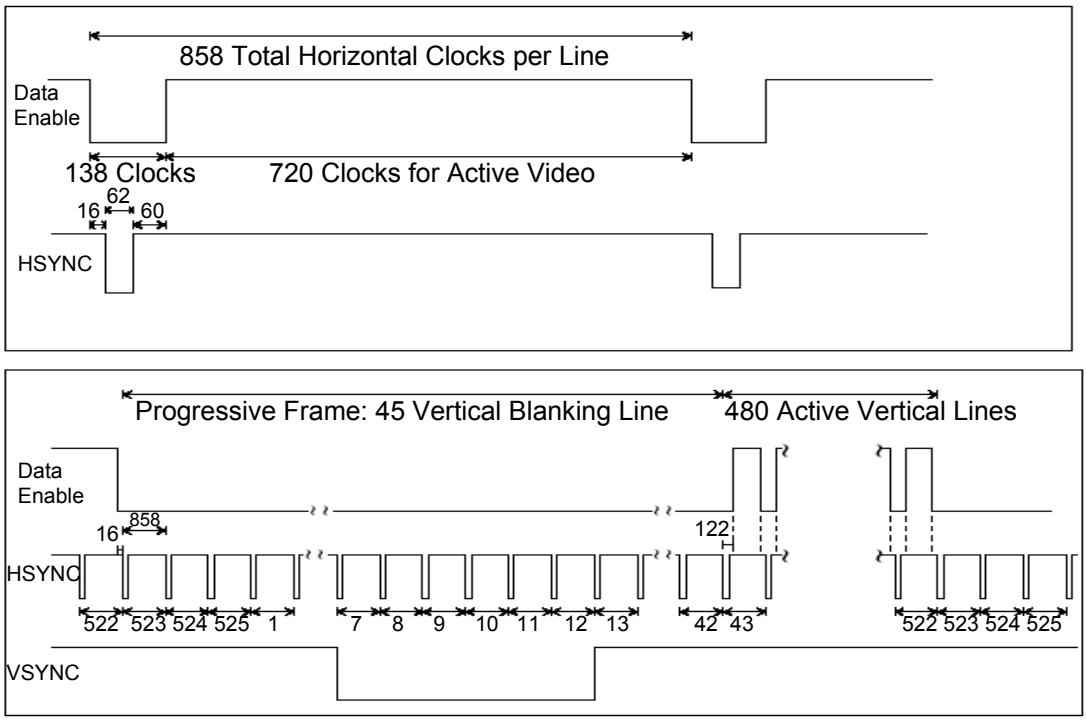

Figure 5-1: Timing diagram for NTSC 480p@60 Hz video standard

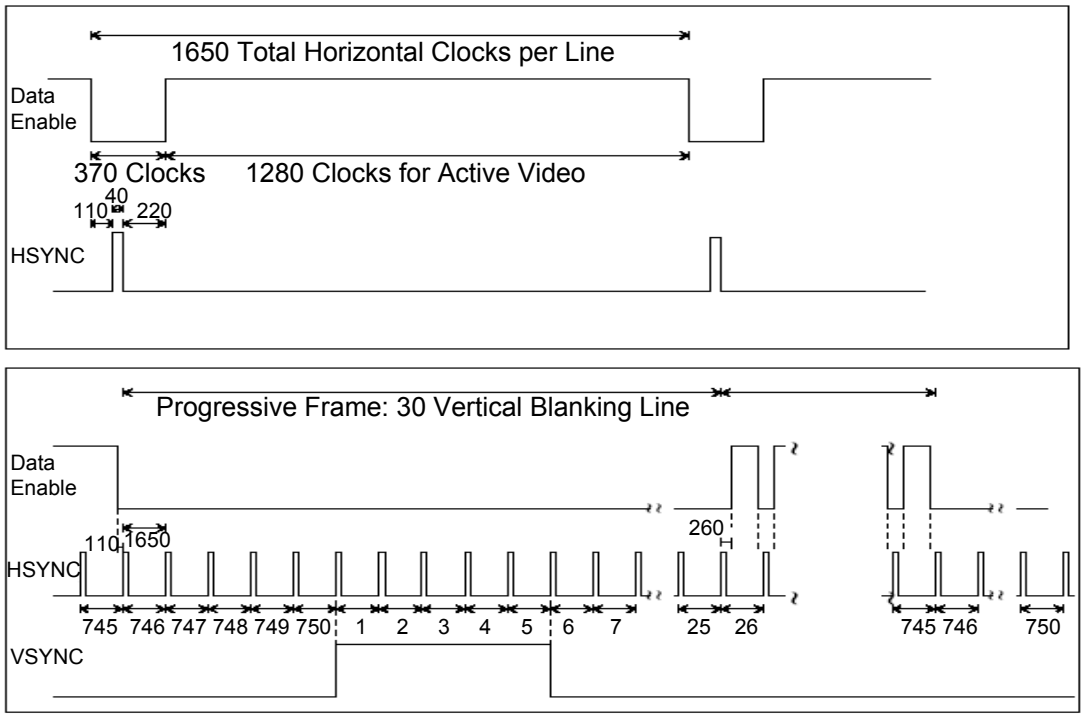

Figure 5-2 : Timing diagram for 720p@60 Hz video standard

Since the input and output video has a frame rate of 60 Hz defined in the standards specification [28] , the nominator will be equal to 1/60 seconds for both the input and output. The denominator can be derived as,

$$N_{cpf} = (active\ video\ clocks\ per\ line + blanking\ time\ clocks\ per\ line) \\ x\ (active\ video\ lines\ per\ \ frame + blanking\ time\ lines\ per\ \ frame) \quad (5.2)$$

Substituting the values from Figure 5-1 into Eq. (5.1), input pixel clock period can be found as,

$$T_{clk\_in} = \frac{1/60}{(720+138)x(480+45)} = 37 \text{seconds} \quad (5.3)$$

and input pixel clock frequency can be found as

$$F_{clk\_in} = 1/T_{clk\_in} = 27.027Mhz \quad (5.4)$$

Substituting the values from Figure 5-2 into Eq. (5.1), output pixel clock period can be found as,

$$T_{clk\_out} = \frac{1/60}{(1280+370)x(720+30)} = 13.468 \text{ seconds} \quad (5.5)$$

and output pixel clock frequency can be found as,

$$F_{clk\_out} = 1/T_{clk\_out} = 74.25Mhz \quad (5.6)$$

At each 74.25Mhz output pixel clock, one interpolated output pixel must be available, therefore the throughput of the system will be

$$Througput_{output} = 74.25M\ \ pixels/\sec ond \quad (5.7)$$

In 480p to 720p standards conversion, the scaling ratio is less than 2, which is chosen as an upper bound in the proposed architecture to reduce the complexity. Setting an upper bound at the scaling ratio, the hardware implementation of the proposed scaling algorithm can be performed in two different ways:

- ***Output based flow:*** For each HD pixel coordinate, find the corresponding 5x5 low resolution window, and perform feature extraction, classification, and interpolation on this neighborhood.
- ***Input based flow:*** For each SD pixel coordinate, find the corresponding four HD pixel coordinates. Since these HD pixels are to be interpolated from the same SD pixel neighborhood, perform feature extraction, and classification steps only for once. Then perform interpolation for all four HD pixels corresponding to the SD pixel, and arrange the interpolated pixels in raster-scan order using line buffers. Discard redundant pixels at the interpolation output if L is a non-integer value.

Since the complexity of the interpolation scheme is mainly on the feature extraction and classification blocks, input based method is preferred in this work, thus eliminating unnecessary computations for the feature extraction and classification stages. When output based flow is followed, maximum of four pixels must be available at every input pixel clock cycle, and therefore the throughput of the system will be

$$Throughput = 4x27 = 108\ Mpixels/\sec ond \qquad (5.8)$$

5.1.2 Logic Area Constraints

The volume of the production of a flat panel display solution is in the order of million units for the consumer electronics market, and this brings a constraint on the cost of the solution, hence the logic area. In order the integrate the designed solution into an industrial product, only the low cost FPGA families are feasible, which eliminates the use of high performance FPGA families such as Xilinx Virtex-4 and Virtex -5 family, or Altera Stratix –II or Stratix-III family. The target platform was chosen from the Xilinx low cost FPGA family, and the architecture was designed for a Spartan3 XC3S2000 FPGA, with equivalent gate count of 2Milllion system gates.

5.1.3 Resource Sharing Options

Modified RS algorithm introduces reduced implementation cost by replacing soft filtering with hard filtering and by reducing the number of classes used in the classification stage. The hardware cost can be further reduced by exploring the design space. Two main factors that can affect the implementation efficiency are the input/output data rate, and the target implementation platform. The optimum degree of pipelining and resource sharing achievable may vary depending on these two factors. An efficient implementation for a low cost FPGA family may result in an over

pipelined, or over parallelized architecture, with low resource utilization for a high performance FPGA family. Although the design is targeted for 480p SD to 720p HD video conversion, a more flexible architecture is proposed to easily adapt to different scaling ratios. To provide an efficient implementation for different data rates, and different implementation platforms, the degree of resource sharing should be variable. The degree of resource sharing in feature extraction and classification units is defined as

$$D_r = \frac{\mathrm{N_e}(\#\ \text{of elements in the feature vector})}{\mathrm{N_p}\ (\#\ \text{of processing paths in feature extraction unit})} \quad (5.9)$$

Choosing a non integer D_r value will result in $(\lceil D_r \rceil * N_p) - N_e$ slots of the processing path to be idle at every $\lceil D_r \rceil$ clock cycle. Choosing an integer D_r value will fully utilize the data path since $\lceil D_r \rceil = \mathrm{D_r}$ which implies no idle slots. To achieve the desired data throughput with different resource sharing levels, core clock frequency, F_{clk_core}, must be related to the input pixel clock frequency, F_{clk_in} with the following formula

$$F_{clk\_core} = D_r * F_{clk\_in} \quad (5.10)$$

One can choose the degree of resource sharing, D_r by investigating the input pixel rate and the maximum core clock frequency achievable by the target FPGA

5.2 Hardware Blocks

5.2.1 Top Level

Top level block diagram of the hardware architecture designed for the modified RS algorithm is shown in Figure 5-3. There are 6 main blocks; input memory (IM), output memory (OM), feature extraction (FE), classification (CL), interpolation (IN), and color space conversion units (rgb2ycbcr,ycbcr2rgb). Modified RS algorithm is performed on the luminance (Y) path. Chrominance signal is interpolated performing pixel replication. RGB input data is converted to YCbCr using ITU equations. 4 rows of luminance data are stored in line buffers to provide the 5x5 pixel neighborhood required for interpolation step.

At each input pixel arrival, the control unit reads a 4x1 pixel column from the line buffers, writes the incoming pixel to the line buffers and provides the 3x3 and 5x5 pixel windows to the feature extraction and classification units. Feature extraction

unit operates on a 3x3 window and extracts an 8x1 feature vector which is used by the classification unit. The classification unit calculates the distance between the feature vector and the predetermined class vectors and outputs the index of the class with minimum distance to the feature vector. Interpolation unit uses the class index to address the filter coefficient LUT and select the appropriate filter for the input pixel neighborhood. Convolution is performed using constant coefficient multipliers. Interpolated outputs are stored at output line buffers. Since the input and output sample rates are different, output memories are required to provide appropriate rate and order of data at the outputs.

The proposed architecture supports scaling of video signals where the scaling ratio $L \leq 2$. When scaling ratio $L = 2$, 4 HD pixels are interpolated for each SD pixel. For non integer scaling ratios $1 < L < 2$, output pixel selector block finds the pixels to be discarded using Eq.(4.4).

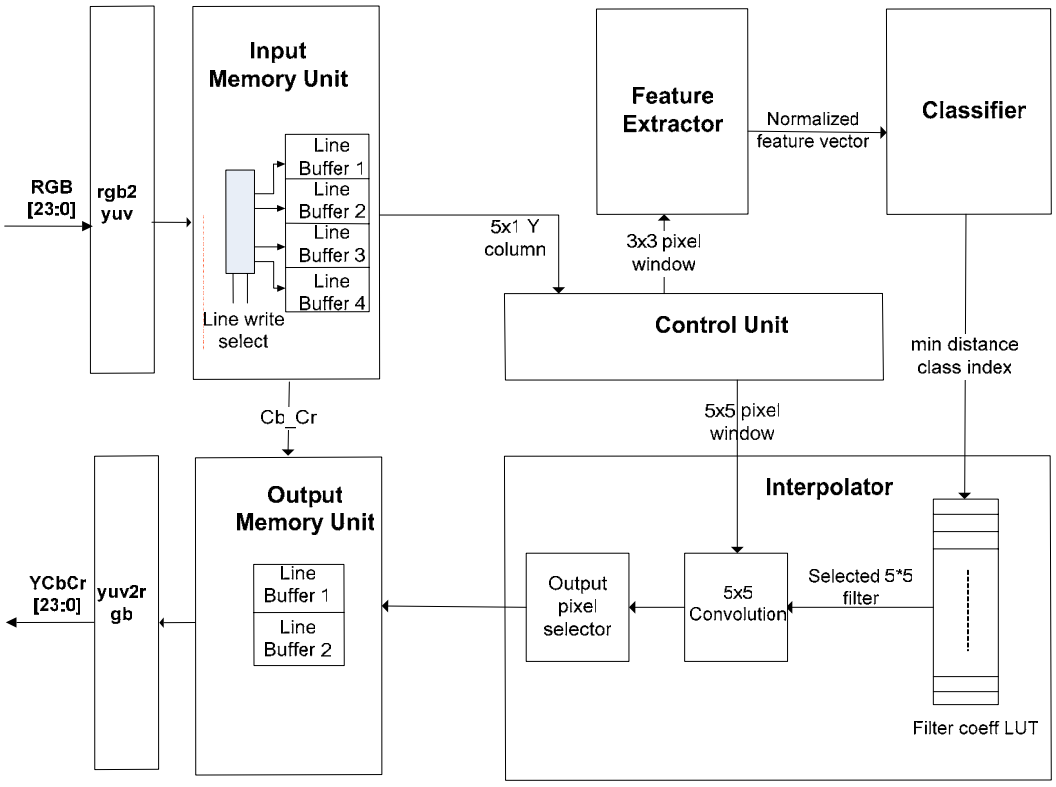

Figure 5-3: Top level block diagram of the proposed hardware

5.2.2 Control Unit (CU)

Control unit provides input data to the data path blocks at appropriate timing and format. The functionality of the CU is illustrated in Figure 5-4.

The control unit

- Generates the memory address and control signals for IM and OM blocks, and pipeline control signals for other blocks.
- generates the synchronization signals defined at the video standards (hsync,vsync,data enable).The porch time and sync time are defined as generics in the RTL code, and the values corresponding to the input and output video resolutions must be available before logic synthesis.
- Generates the hp and vp values used at interpolation stage.
- For scaling ratios less than 2, control unit selects the pixels to be omitted using the low resolution pixel coordinate and the scaling ratio L .

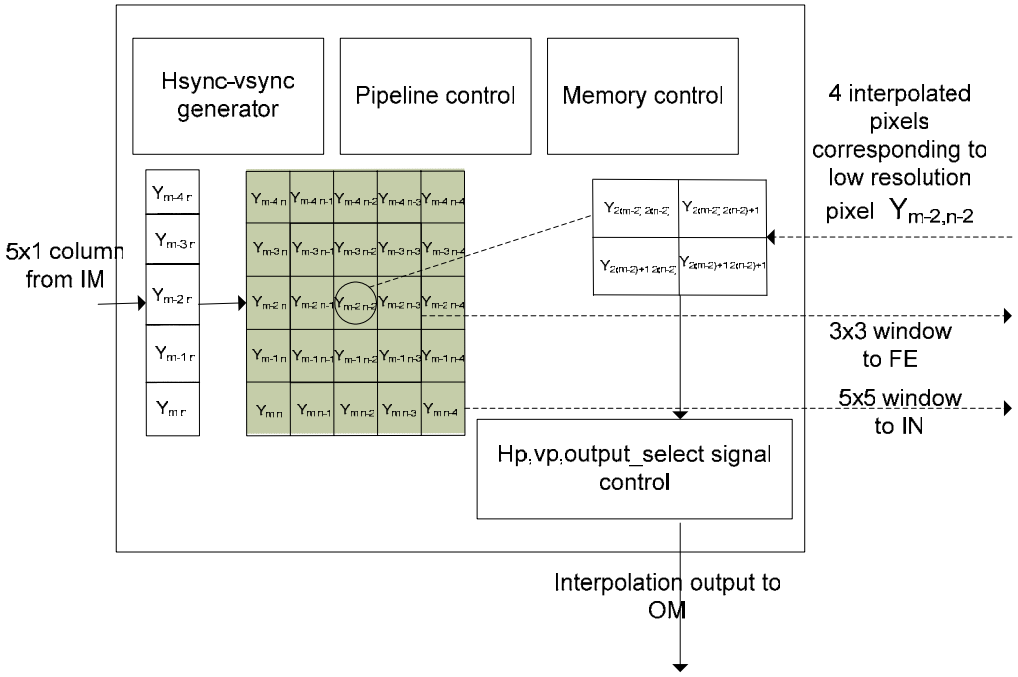

Figure 5-4 : Control unit block diagram

5.2.3 Color Space Conversion Unit (CSC)

Color space conversion unit converts the RGB data to YCbCr data and vice versa using the equations defined in ITU-R BT.601 standard specification. The equations to convert gamma corrected RGB data, referred as R'G'B' into YCbCr are

$$Y = (77/256)R' + (150/256)G' + (29/256)B' \quad (5.11)$$
$$Cb = -(44/256)R' - (87/256)G' + (131/256)B' + 128$$
$$Cr = (131/256)R' - (110/256)G' - (21/256)B' + 128$$

The equations to convert YCbCr data into R'G'B' are

$$R' = Y + 1.371(Cr - 128) \quad (5.12)$$
$$G' = Y - 0.698(Cr - 128) - 0.336(Cb - 128)$$
$$B' = Y + 1.732(Cb - 128)$$

Both the RGB data and the coefficients have 8 bit precision. The multiplications in the color space conversion block are implemented as shift-add operations since the coefficients are always constant.

5.2.4 Input Memory Unit (IM)

IM unit operates at input pixel clock frequency f_{clk_in}. The block consists of 4 line buffers, to provide a 5x1 pixel column to the CU. The length of the line buffers is equal to the input video's horizontal resolution. Block diagram of the IM unit is shown in Figure 5-5.

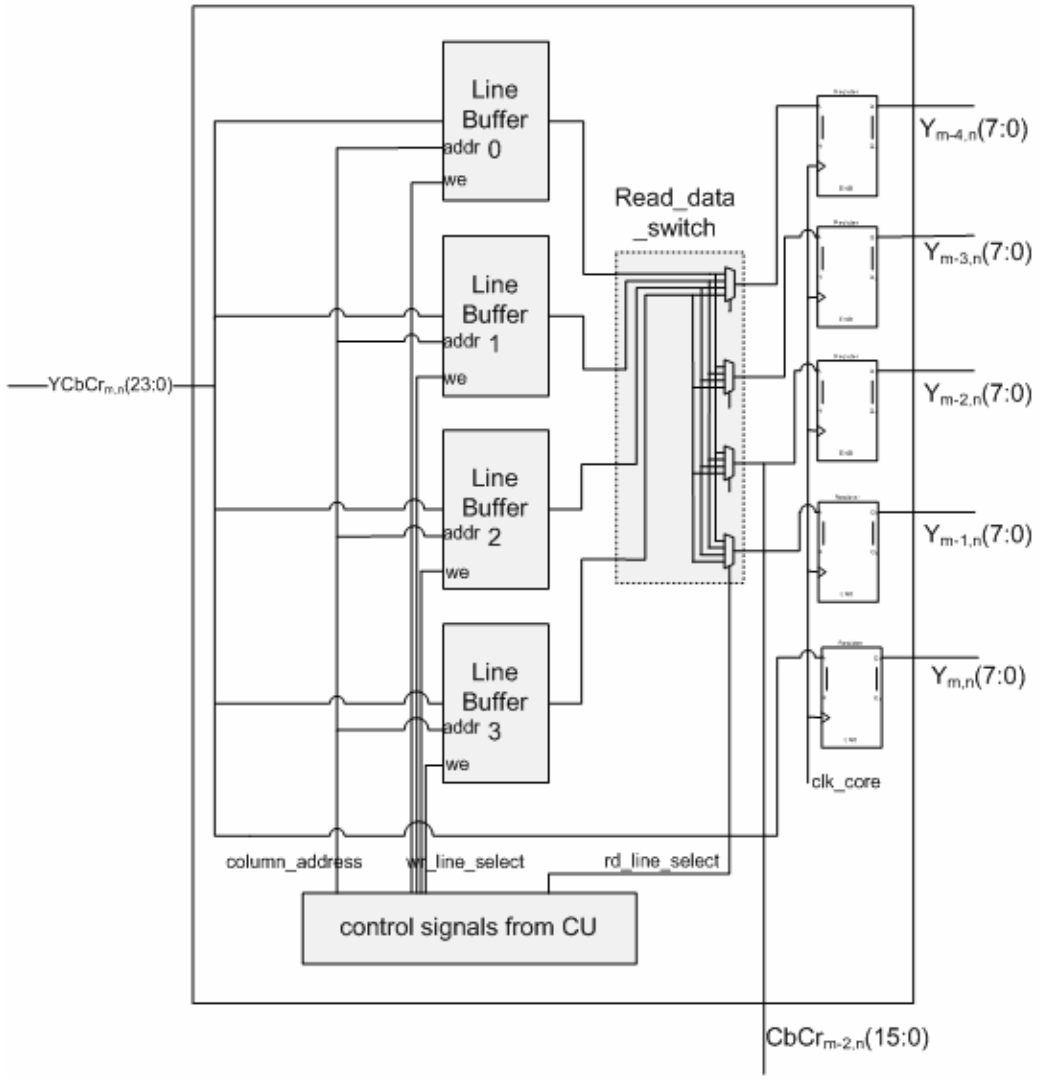

Figure 5-5 : Input memory unit block diagram

When the YCbCr data for m^{th} row, n^{th} column, $YCbCr_{m,n}(23:0)$ arrives at the IM unit, it is bypassed to the last element of the 5x1 pixel column register at the outputs. The remaining 4 pixels of pixel column registers are read from column n of the line buffers. Finally the input pixel is written to one of the line buffers, selected by the control unit. Line buffers are designed in the form of a circular buffer. m^{th} row of the image is written into line buffer $m(\mathrm{mod}\,4)$, therefore the 2 least significant bits of the line counter can be decoded to serve as the write enable signal to the line buffers. Since the order of rows at the line buffer are changing at every image row, the line buffer outputs are switched to the appropriate column registers, with the multiplexer logic controlled by the CU. In addition to 4 rows of luminance(Y) values, the chrominance(CbCr) values must also be stored in line buffers to line align the processed Y value and replicated CbCr values at the output. However 2 line buffers are sufficient since the latency of the processed Y values will be equal to 2 lines + 5 columns + pipeline latency.

5.2.5 Feature Extraction Unit (FE)

FE unit operates at core clock frequency f_{clk_core} defined at Eq.(5.10). Figure 5-6 shows the architecture of the feature extraction unit for resource sharing value $D_r = 1$. A fully parallel implementation where $D_r = 1$, will use 15 adder/subtractors and 32 multipliers. Exponentiation with constant value, 0.75 is performed using a look up table. For $D_r = 1$, the core clock frequency of the FE unit can be calculated from equation (5.10) to be equal to the input pixel clock frequency which is 27.027 Mhz, for 480p to 720p conversion. Running the FE unit four times the input pixel clock frequency, D_r can be set to 4 to reduce the number of adder/subtractors to 4 and the multipliers to 8, with an ignorable increase at the number of pipeline registers. Figure 5-6 shows the architecture of the feature extraction unit for $D_r = 4$. Instead of computing each feature vector element in parallel by inserting 8 pixels to the pipeline at every 27.027 Mhz clock, the vector elements are computed in a serial parallel manner where two feature vector elements are inserted into the pipeline at every 108 Mhz clock. The block will generate 2 feature vector values at each 108 Mhz clock cycle, producing a throughput of 8x1 feature vector per input pixel clock cycle. As long as the design can be synthesized to work at faster clock frequencies, increasing the resource sharing level reduces the total area, since the overhead introduced by such an operation is only the increased number of pipeline stages, which is not a major issue for an FPGA based design where usually the lookup tables are the critical resource. The result of the FPGA synthesis for different resource sharing values is illustrated in chapter 6.

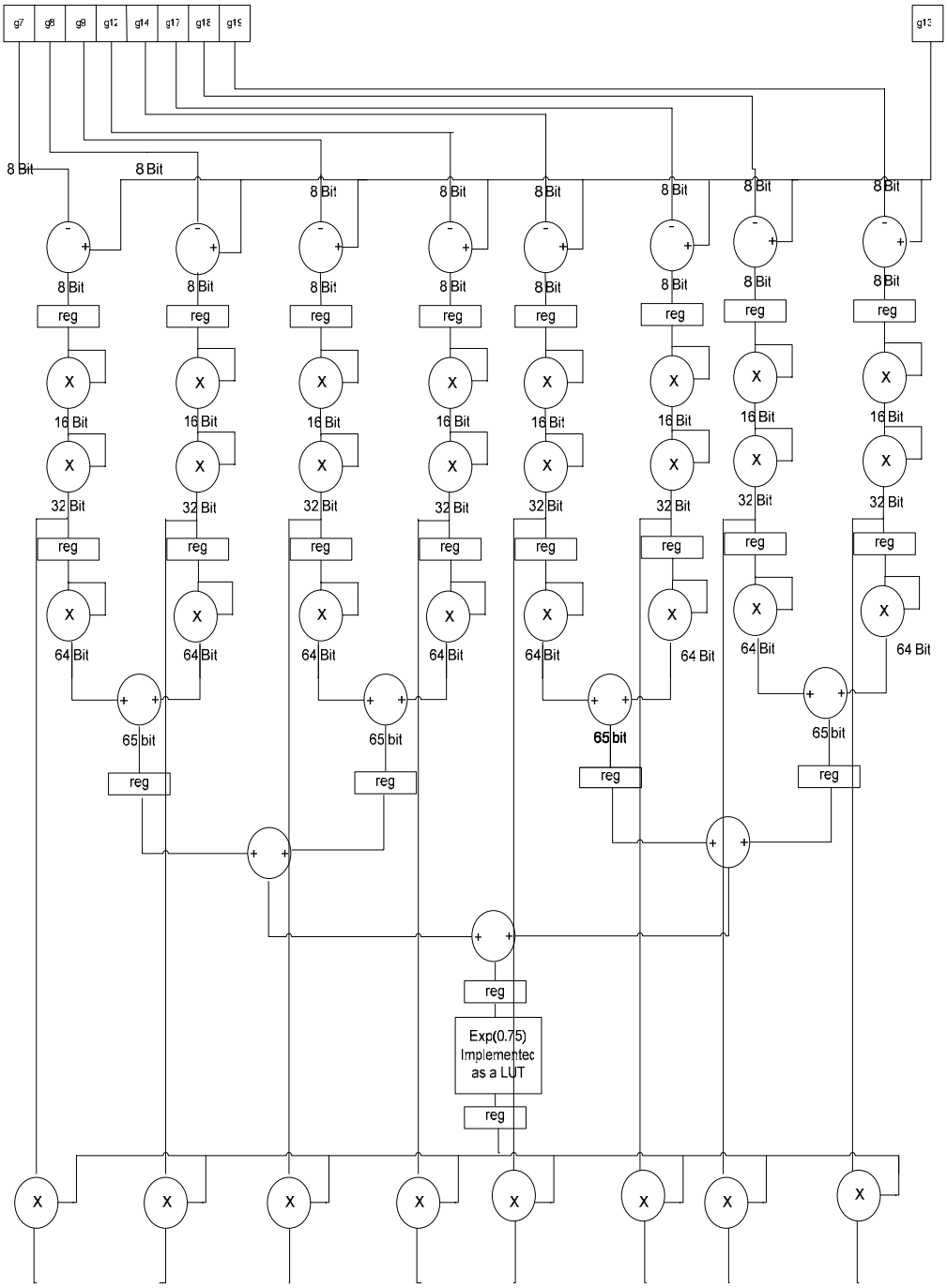

Figure 5-6 : Architecture of the feature extraction unit for $D_r = 1$

The control signals are not illustrated in Figure 5-6 and Figure 5-7 for clarity of the figures. For the fully parallel implementation in Figure 5-6, the control logic is straightforward. Each register will require an enable signal, which can be generated delaying the enable signal at the input of the design. For the serial-parallel implementation in Figure 5-7 ,in addition to the enable signals, appropriate control signals must be applied to the select inputs of the multiplexers, and synchronous reset inputs of the accumulation registers.

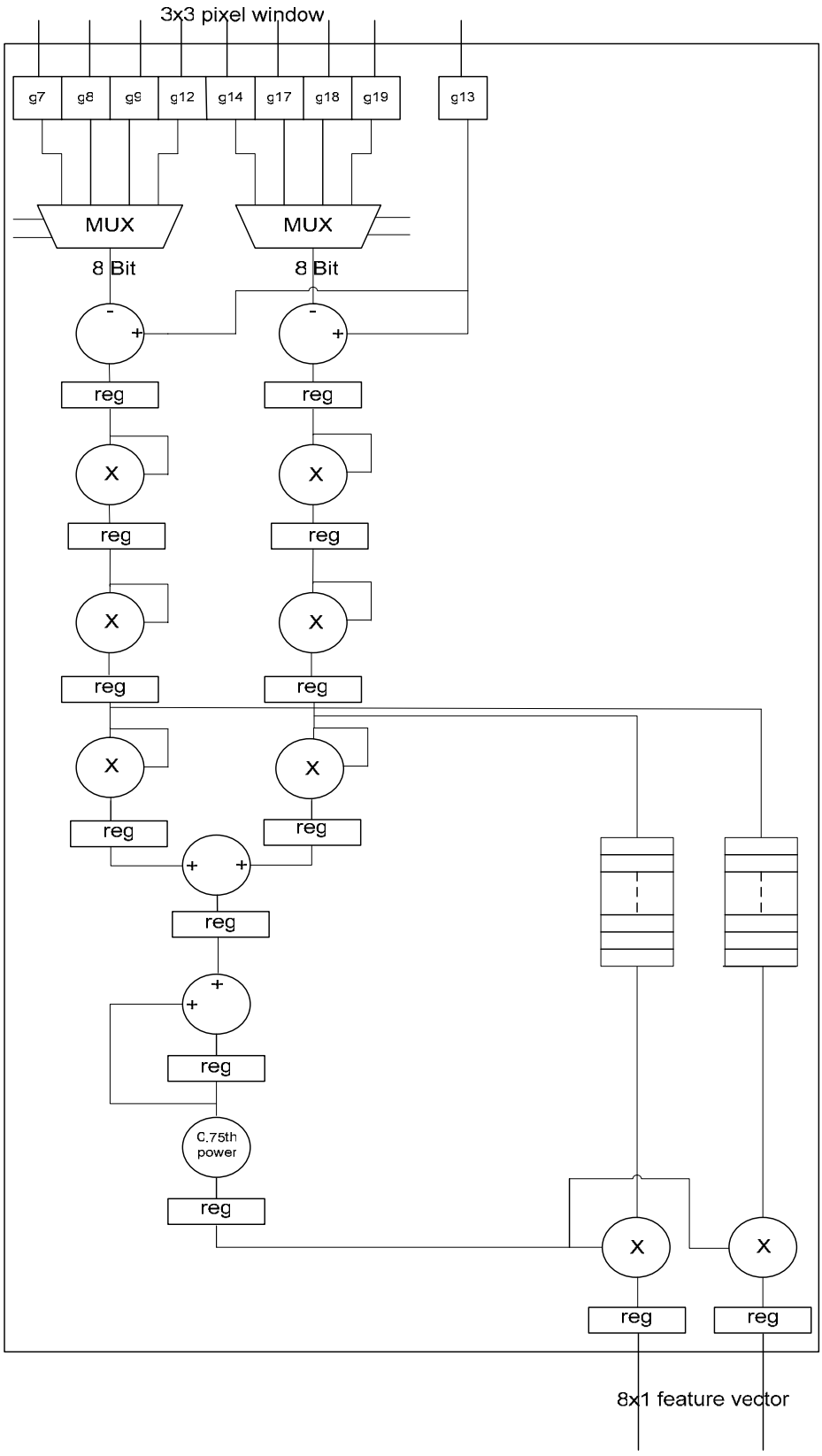

Figure 5-7 : Architecture of the feature extraction unit for $D_r = 4$

5.2.6 Classification Unit (CL)

CL unit operates at core clock frequency f_{clk_core}. A fully parallel implementation where $D_r = 1$ will use 75 adder/subtractors, and 80 multipliers. Setting $D_r = 4$ will reduce the number of adders and multipliers to 20. Figure 5-8 shows the architecture of the classification unit when $D_r = 4$. An alternate implementation of the classification unit can make use of the LUTs to implement the distance computation part of the logic. The size of the required LUT can be given as $2^k * v * c * o$ where k is the bit width of the feature vector element, v is the dimension of the feature vector, c is the number of classes, and o is the output bit width of the LUT output. For $k = 9, v = 8, c = 5$ and $o = 13$, the size of the LUT will be approximately 265 Kbits.

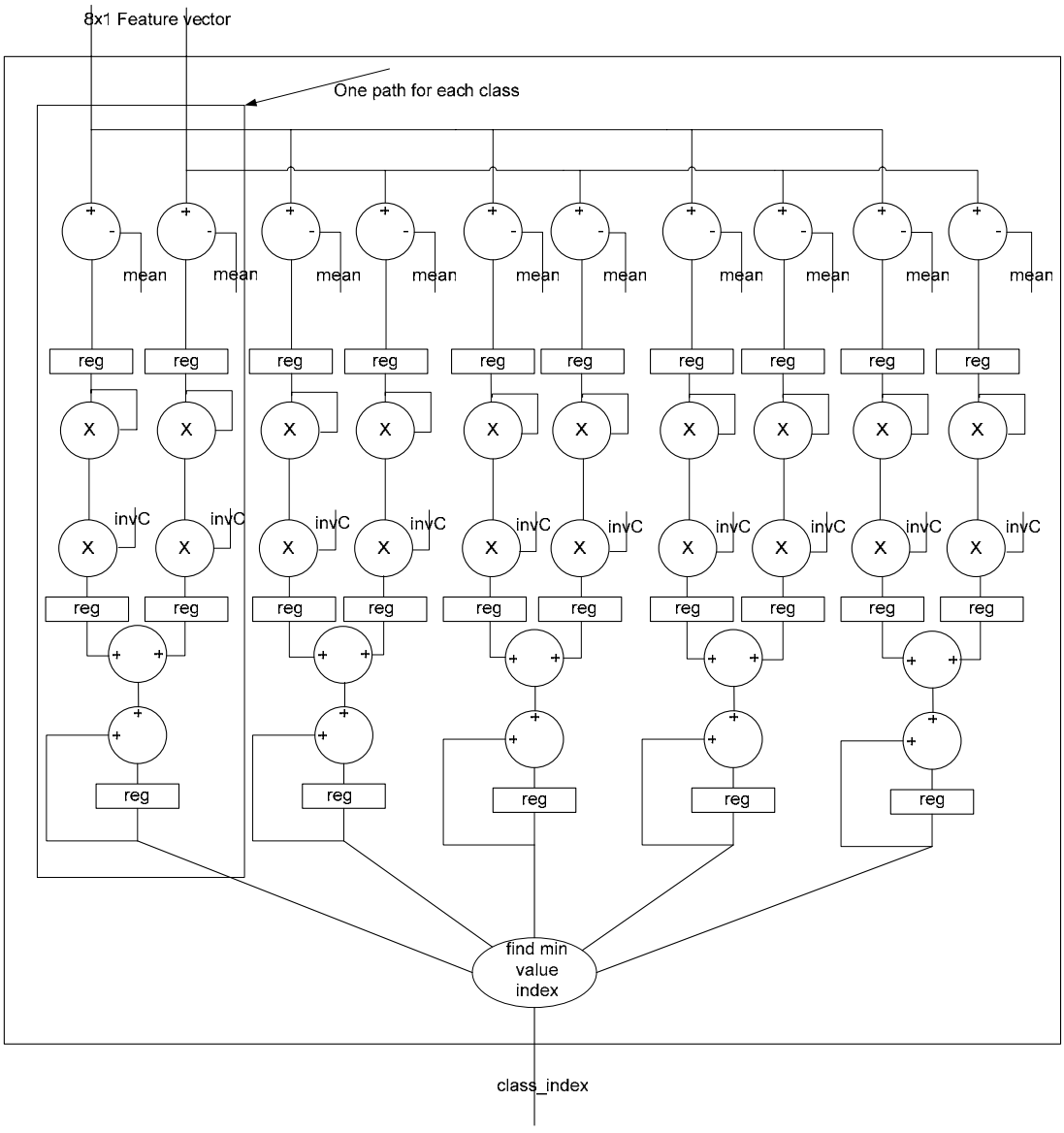

Figure 5-8 : Architecture of the context classification unit for $D_r = 4$

5.2.7 Interpolation Unit (IN)

IN unit operates at core clock frequency f_{clk_core}. Figure 5-9 shows the architecture of the interpolation unit. For every pixel luminance value in the image the IN block accepts the 5x5 window generated by the IM and CU, the class index from the CL, vertical and horizontal phase of the high resolution pixels from the CU, and generates 4 high resolution pixels using the selected interpolation filters.

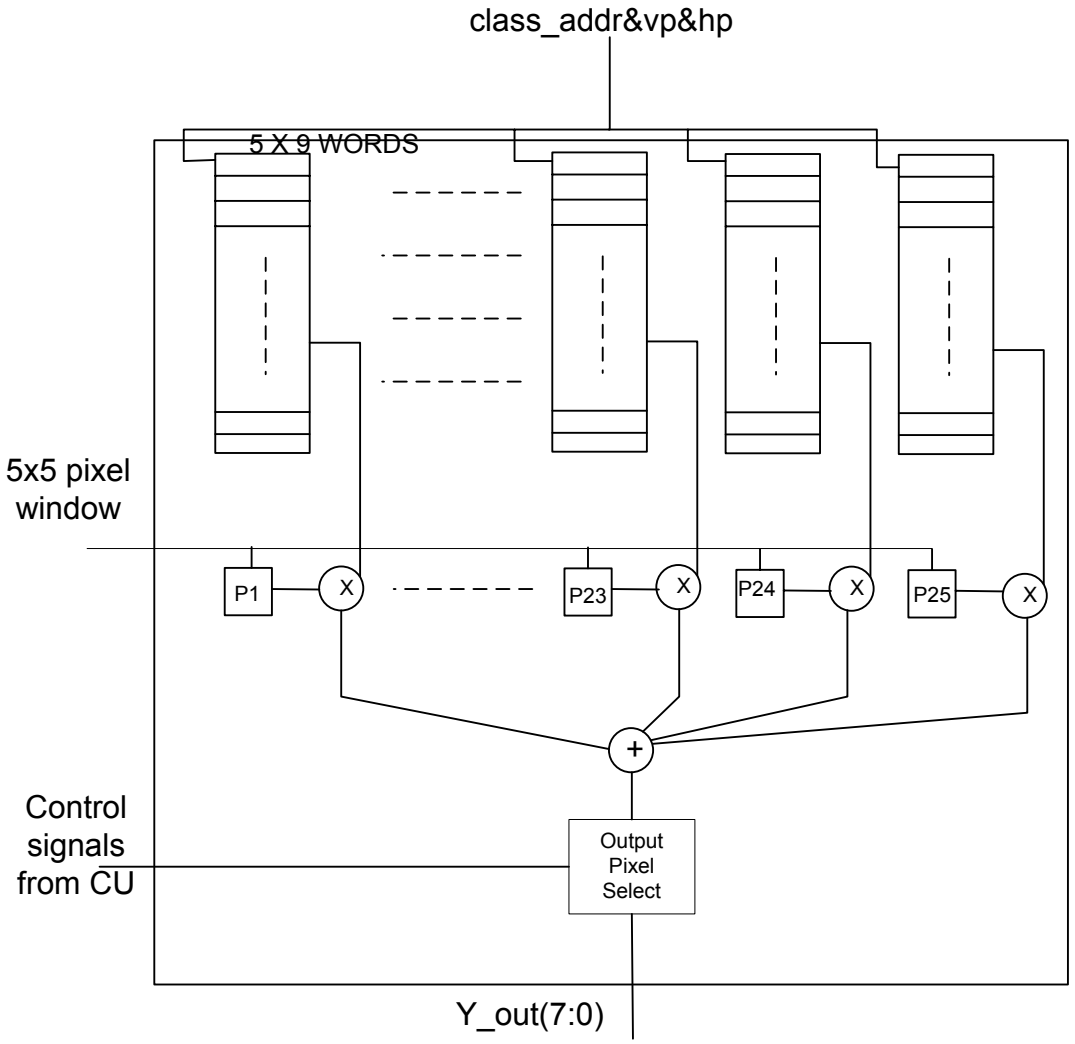

Figure 5-9 : Architecture of the interpolation unit

5.2.8 Output Memory Unit (OM)

OM unit also operates at core clock frequency f_{clk_core}. It basically chooses the appropriate pixels between the high resolution pixels generated by IN unit and arranges them in raster-scan order according to the scaling ratio. There is a unique mapping between low and high resolution image pixels for each scaling ratio. For non-integer scaling ratios only a certain amount of high resolution pixels are used in

output image. Eq. (4.4) shows the corresponding low resolution pixel for given high resolution pixel.

For scaling ratios smaller than 2, output memory consists of 2 line buffers having the size of high resolution image's horizontal resolution. The line buffers are essential as the two of four high resolution pixels generated from m^{th} row ; n^{th} column pixel by IN unit belongs to the $(2m)^{th}$ line while the other two pixels belong to $(2m+1)^{th}$ line of high resolution image. These two latter pixels should be stored till all the high resolution pixels prior to them in raster-scan mode are produced and sent to the output.

In the case of scaling ratio $L=1.5$, output memory unit chooses between high resolution pixels produced from the low resolution pixel at coordinates (m,n) in four different ways depending on the values of m and n. Output memory unit operation for scaling ratio of 1.5 is illustrated in Figure 5-10

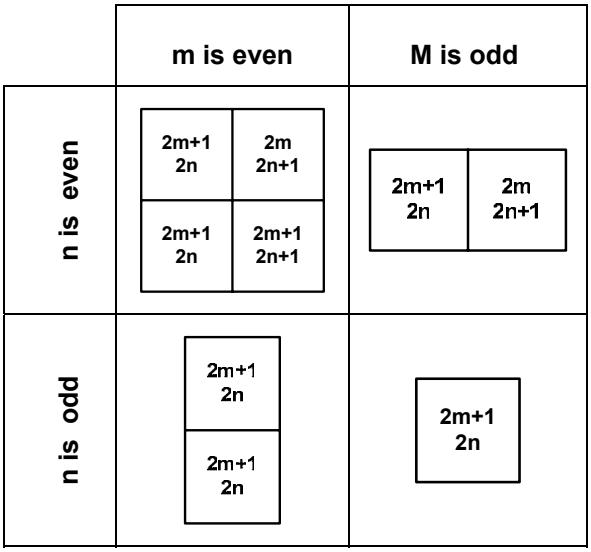

Figure 5-10 : Output memory unit operation for $L=1.5$

5.2.9 Asynchronous Buffers and DDR Frame Buffer

Output memory unit aims to order the interpolated pixel outputs in raster scan format, which is the required format in an LCD panel input. For demonstration purposes, the designed hardware does not directly drive the LCD panel, but it drives the video processor inside the TV. In the case of driving an LCD panel directly, the vertical and horizontal sync time of the output video can be modified to resolve the asynchrony in line rates. However in the case of driving a video processor, the design must output the video data in exactly the same timing illustrated at Figure 5-2. Since the input and output pixel rates are different, there is an asynchrony in the design. The output memory works at core clock frequency, f_{clk_core} , which is

different from the output resolution clock frequency, f_{clk_out}, hence there should be an asynchronous buffer after the output memory unit. The buffer size needed can be calculated examining the input and output relations at Figure 5-1 and Figure 5-2. Since the input and output video have same frame refresh rates, it is easy to see that the input and output are synchronized at each frame, hence a frame buffer may guarantee to resolve the asynchrony. However, depending on the line rates of the input and output video, a solution without a frame buffer would also be possible.

Vertical scaling ratio of the design is referred as $L_{vertical}$ .If the design can output $L_{vertical}$ lines, at the time period where one input line arrives, the input and output would be fully synchronized and there would be no need for buffering. In case this condition is not met, the number of lines generated and written into the asynchronous buffer at the time one output line is read from the buffer must be computed to find the required buffer size. From the output video specifications, the time, one output line is read from the output buffer will be

$$T_{line\_out} = (clocks\ per\ output\ line)\ x\ T_{clk\_out} \quad (5.13)$$

From the input video specifications, the time one input line arrives to the scaler will be

$$T_{line\_in} = (clocks\ per\ input\ line)\ x\ T_{clk\_in} \quad (5.14)$$

At T_{line_out} time interval, the number of interpolated output lines, which will be written to the asynchronous buffer can be written as

$$N_{async\_line\_wr} = \frac{T_{line\_out}}{T_{line\_in}}\ xL_{vertical} \quad (5.15)$$

The number of output lines read from the asynchronous buffer at T_{line_out} time interval, is referred as $N_{async\_line\_rd}$ and is equal to 1. For 480p to 720p video resolution up-conversion, the number of interpolated output lines written to the asynchronous buffer at T_{line_out} time interval, which is referred as $N_{async\_line\_wr}$ can be computed using Eq. (5.13), Eq. (5.14), Eq. (5.15), Figure 5-1 and Figure 5-2.

$$N_{async\_line\_wr} = \frac{(1280+370)x\dfrac{1/60}{(1280+370)x(720+30)}}{(720+138)x\dfrac{1/60}{(720+138)x(480+45)}}x1.5 = 1.05 \quad (5.16)$$

Since at every output line, 1 line is read from the buffer, but 1.05 line is written to the buffer, 0.05 lines must be stored. For 720p video output, the required buffer size will be 720x0.05= 36 lines. In order to store 36 lines of output, the required memory size will be 36x1280x24 = 1105920 bits. Since the internal line storage limit of the target FPGA is 720Kbit, an external storage is required, and a DDR frame buffer was used. Asynchronous buffers and DDR frame buffer controller design was not in the scope of this study, and those design blocks were obtained from the development board used for the project. Several modifications were made in the design to match the read and write frequencies of the asynchronous buffers. Figure 5-11 illustrates the frame buffer and asynchronous buffer structure used in the design.

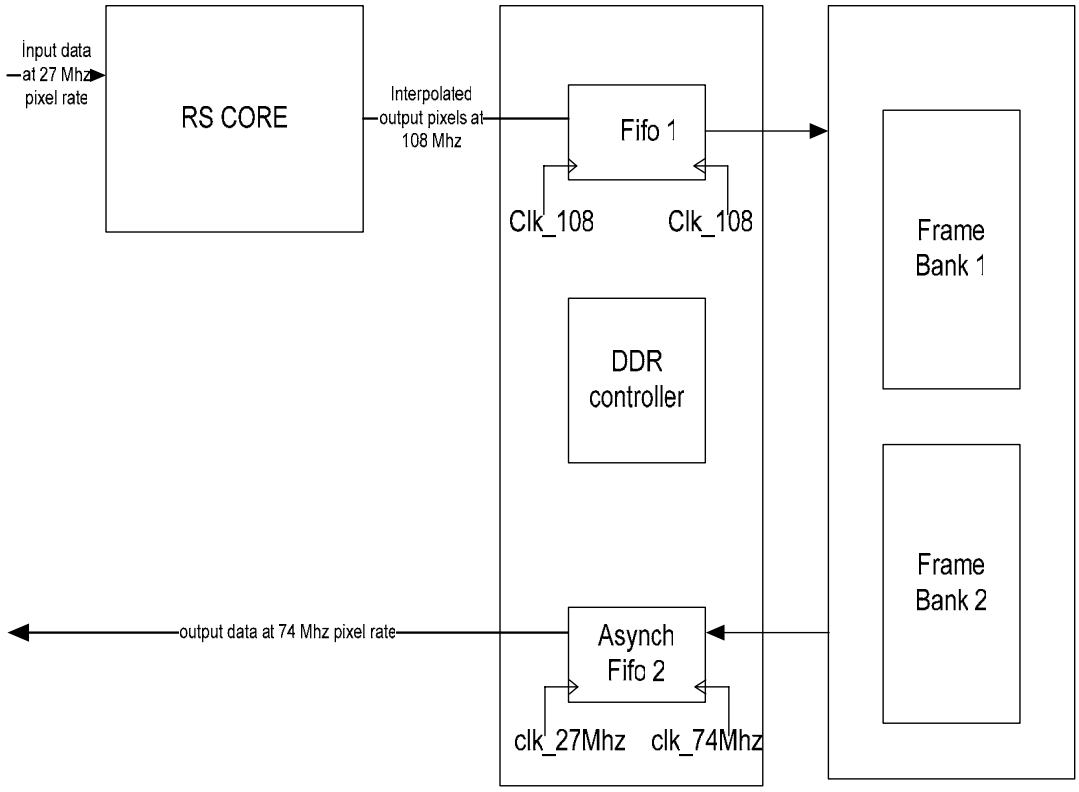

Figure 5-11 : DDR frame buffer and asynchronous buffers

The MRS core generates interpolated output at 108 Mhz clock frequency. The interpolated outputs are written into Fifo1 and are written to the DDR memory in

bursts. DDR controller generates the DDR signaling for DDR write and read operations. Ping pong architecture is used to read/write from the frame buffers. When the n^{th} frame is written to bank 1, $(n-1)^{th}$ frame is read from bank2. At the next frame write operation, the ddr controller switches the banks, hence when n^{th} frame is read from bank 1, $(n+1)^{th}$ frame is written to bank2. FIFO 2 is an asynchronous FIFO that inputs pixels from the DDR buffer at pixel clock frequency of 108 MHz and outputs pixels at pixel clock frequency of 74.25 MHz.

6. FUNCTIONAL VERIFICATION, FPGA MAPPING & REAL TIME TESTS

6.1 Functional Verification

6.1.1 Simulation Platform

The fixed point C model of the algorithm and the VHDL testbench is used to verify the system prior to FPGA mapping. Figure 6-1 shows the structure of the functional verification platform. Reference input and output stimuli files are generated by the fixed point C model. Reference input stimuli file includes the RGB data for a 480x720 input image, and reference output stimuli file includes the RGB data for a 720x1280 output image which is up scaled using the MRS algorithm fixed point C model. The testbench is written in VHDL. The aim of the testbench is not only to test the datapath blocks but also to test whether the control unit correctly generates the control signals, namely the horizontal and vertical sync signals and the data enable signal in the desired format at the standards.

To test the functionality of the timing signals generated by the MRS_TOP module

- the hsync-vsync generator block generates the hsync-vsync-data_en signals for 480p standard, the input stimuli is read from the text files, and applied to the MRS_TOP module using these control signals.
- Output reference file is read using the hsync-vsync-data_en signals for 720p standard generated by MRS_TOP module. The testbench checks whether the control signals generated by the MRS_TOP module accurately matches the timing defined for 720p standard

To test the data path,

- The testbench compares each MRS_TOP output pixel with the corresponding pixel of the reference output file.
- The testbench writes each valid RGB output into an output file for visual inspection. The output text file is converted to a bitmap image using a basic matlab script, and the up scaled image is compared with the corresponding image generated by the fixed point C model.

The simulations are run using the Modelsim simulator, on pre-synthesis register transfer level VHDL code. Post place&route simulations are not performed, since it

takes huge amount of computation time both to generate and simulate the post place & route model, due to the size of the design. Static timing analysis is performed instead to assure correct behavior after FPGA mapping. Since the mapped design is tested on real time, the lack of computational power for post place & route simulations does not introduce a major problem.

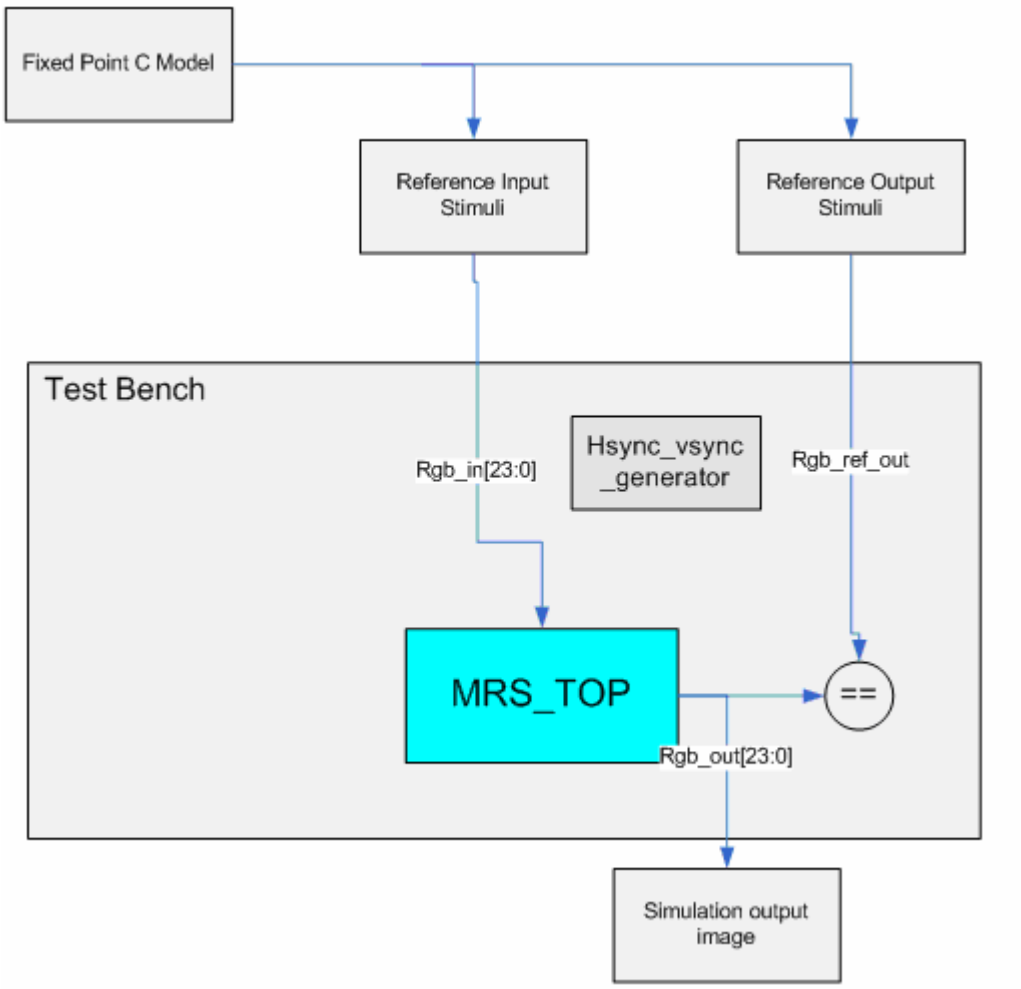

Figure 6-1 : Functional verification platform

6.1.2 Simulation Results

Using the platform described in section 6.1.1, the simulations are run for 5 video frames. 5 input video frames (5 x 480 x 720 = 1.728.000 samples) are first processed by the fixed point C model to generate the 5 reference output stimuli files (5x720x1280 = 4608000 samples).

Figure 6-2 and Figure 6-3 shows a 200x200 portion of the input bitmap image of 480x720 resolution and a 300x300 portion of the output image of 720x1280 resolution generated by functional simulations.

Figure 6-2 : 200x200 portion of the input image

Figure 6-3 : 300x300 portion of the scaled output image

In addition to verifying the outputs of the MRS top level module, simulation waveforms are examined in depth for a limited number of samples. To examine the simulation waveforms, random samples are chosen from the output image, and all the intermediate signal values are dumped into a text file from the fixed point c model, to serve as a reference. In the simulation waveform, the fixed point model references are compared with the signal values at the waveform.

6.2 FPGA Mapping

6.2.1 FPGA Mapping Methodology

In section 5.1, the input and output pixel rates are declared to be 27.027 MHz and 74.25 MHz respectively. Using Eq. (5.10), core clock frequency can be set to 27 MHz, 54 MHz, 108 MHz or 216 MHz, for D_r values 1,2,4,8 respectively. For the

target FPGA family (spartan3), running the modules at 216 MHz is above the practical limits. After extensive synthesis efforts, it is found that internal clock frequency of 108 MHz is an achievable target. Therefore D_r is set to 4 to obtain the maximum resource sharing possible for the target device. Running the internal blocks at 108 MHz requires several efforts on the mapping flow. The required timing is at the limits of the FPGA's performance, therefore several modifications were performed on the RTL code to meet the over constrained goals.

Logic synthesis is performed using Synplify Pro 8.5, and Place & Route operation is performed using Xilinx ISE 8.1 platform.

6.2.2 FPGA Mapping Results

Design had been mapped to Xilinx XC3S2000 FPGA, for D_r values of 2 and 4. To optimize the area, several trial synthesis efforts were performed using different FPGA resources in several blocks. One major architectural decision is to synthesize the classification stage using multipliers or look up tables. Classification step includes 80 multiplications, and as mentioned in section 5.2.6, the multiplication operations can be performed using a lookup table of size 265 Kbits. The choice of whether to use LUTs or multipliers depends on the D_r value and the availability of the logic resources as well. For higher D_r values, the number of multipliers to realize the operation will be reduced by a factor of D_r however the size of the LUT will not change if the implementation is realized using LUTs. Therefore it would be meaningful to use multipliers instead of LUTs for higher D_r values. Another important issue that can affect the implementation result is to choose between block/logic multipliers on any multiplication operation and to choose between block/distributed RAMs in storage elements. As a rule of thumb, the logic synthesizer uses logic multipliers for multiplications that have relatively small operand size, and uses block multipliers for multiplications that have relatively large operand size. The decision criterion is similar for RAM implementation. The synhtesis tool infers block rams for large storage, and distributed RAM for smaller storage. In this study, for particular cases where the timing could not be met, the synthesis tool was forced to infer the desired FPGA resource by the user using several synthesis constraints. The result of the FPGA mapping is listed in Table 6-1. The FE,CL,CU and IN units are referred as core modules in the table.

Table 6-1 : FPGA mapping results for D_r values 2, and 4

		CL stage with BRAMs		CL stage with multipliers	
		D_r=2	D_r=4	D_r=2	D_r=4
core	Slices	6402	2360	9502	3198
	BMULTs	40	34	40	40
	BRAMs	16	16	0	0
total	Slices	8412	3533	10551	4623
	BMULTs	40	34	40	40
	BRAMs	30	30	14	14

The results in Table 6-1 show that when the degree of resource sharing is increased by a factor of 2, the slice count is approximately reduced by a factor of 2. Optimum value of D_r for the particular video scaling problem (480p to 720p) and for the particular target FPGA platform(Xilinx XC3S2000) is found to be 4 after synthesis efforts.

The performance results and timing constraints of the implementation is illustrated in Table 6-2. The latency figures of the design is also illustrated in the table, however the latency of the design is not of any concern since the human eye can not notice such short time intervals. 2x1280 is the latency introduced from storing 2 lines, which is required to start processing. Additional clock cycles are the pipeline latency.

Table 6-2 : Performance results of the implementation

	D_r=2	D_r=4
Clock Frequency(MHz)	54	108
Latency(clock cycles)	2*1280+21	2*1280+37
Throughput(MPixels/second)	74.25	74.25
Frame Processing Time(ms)	1.6	1.6

The results obtained for $D_r = 4$ is compared with several work on literature. To the author's best knowledge, a hardware implementation for a classification based resolution enhancement method has not been presented previously. Proposed method is therefore compared with several FPGA implementations presented on simpler linear scaling methods. In [29] a run time configurable, spline interpolation system is implemented in a Xilinx XC4000 FPGA and for a 522x128 output image an execution time of 35.5 ms is reported. In [30] bicubic interpolation is implemented in Xilinx VirtexII-Pro FPGA, and for a 640x480 output image an execution time of 3.5 ms is reported. In [31] adaptive Newton interpolation is implemented in FPGA and it is verified on an LCD panel with output resolution of 1024x768. In [32] a video scaler with output video of 1024x768 @30fps was presented. Results obtained from the previous work and the results obtained from this work are given in Table 6-3. It must be noted that the results in the table are given for a general idea on the proposed method's complexity, and not to compare scaling methods with the proposed resolution enhancement method, since the approaches followed(linear scaling vs. content adaptive scaling), target FPGA platforms and target application(input and output resolutions) are entirely different.

Table 6-3 : Comparison of the implementation with previous work

	Image Size	Exec. time	Freq	FPGA	Slices	Block RAMs
[29]	522x128	35ms	30 Mhz	Xc 4000	na	Na
[30]	640x480	3.5ms	100 Mhz	Virtex2 pro	~1700	56KB
[31]	1024x780	na	na	na	na	Na
[32]	1024x780	na	28 Mhz	XCV 2600	~7500	30KB
This work	1280x720	1.6ms	108 Mhz	XC3S 2000	3533	60KB

6.3 Real Time Tests

The design had been mapped to a Xilinx XC3s2000 FPGA and is tested on real time, for a 480p to 720p standards conversion application. Real time test platform and test results are explained in detail in the following sections.

6.3.1 Real Time Test Platform

In order to verify the implementation and evaluate the performance of the algorithm, the design was integrated into a flat panel display product in Vestel R&D Labs. Figure 6-4 illustrates the platform that was set to test the implementation.

Real time test platform is composed of a DVD player, a 40 inch LCD TV with native resolution of 1366x768, and the Tora TB3S2000 development board. DVD player is set up to output 480p NTSC video standard. DVDs are encoded with minimum compression rate possible to prevent any ringing, or blocking artifacts that can be caused by mpeg compression. DVD player outputs are bypassed to the FPGA via the DVI interface. HDMI receiver and transmitter modules inside the development board make the necessary voltage conversion for HDMI standard. The inputs to the FPGA are the RGB data and synchronization signals (hsync, vsync, and data enable). The FPGA receives the input video signal at 480p(480x720 @60 Hz), scales the signal by a factor of 1.5 to obtain video at 720x1080 resolution, inserts black pixels to the leftmost and rightmost 100 pixel columns to obtain 720x1280 video, generates the hsync, vsync and data enable signals for 720p(720x1280@60 Hz) and outputs the video through the DVI output interface of the TB3S board. In order to evaluate the performance of the designed scaler, TV's scaling function must be bypassed. TV software is modified to bypass any data when the input source is detected to be 720p. Therefore the TV does not try to scale up the 720x1280 input video (that was generated by the FPGA) into its native resolution (1366x768) but instead it inserts black regions for the missing pixels.

6.3.2 Test Results

Once the platform was set up, two LCD TVs were put side by side to compare the scaling performance of the scaler in this study, and the TV's scaler inside the video processor. Since there is no objective evaluation method for the algorithm's performance on TV platform, the algorithms were compared subjectively with "Vestel Video Quality Group's" comments. The implemented design was found to perform much better on preserving details of the image , while the bicubic scaling based method on the TV's video processor had a more blurry output. Training of the filter

coefficients were performed for several different training sets to obtain the filter coefficients with optimum performance.

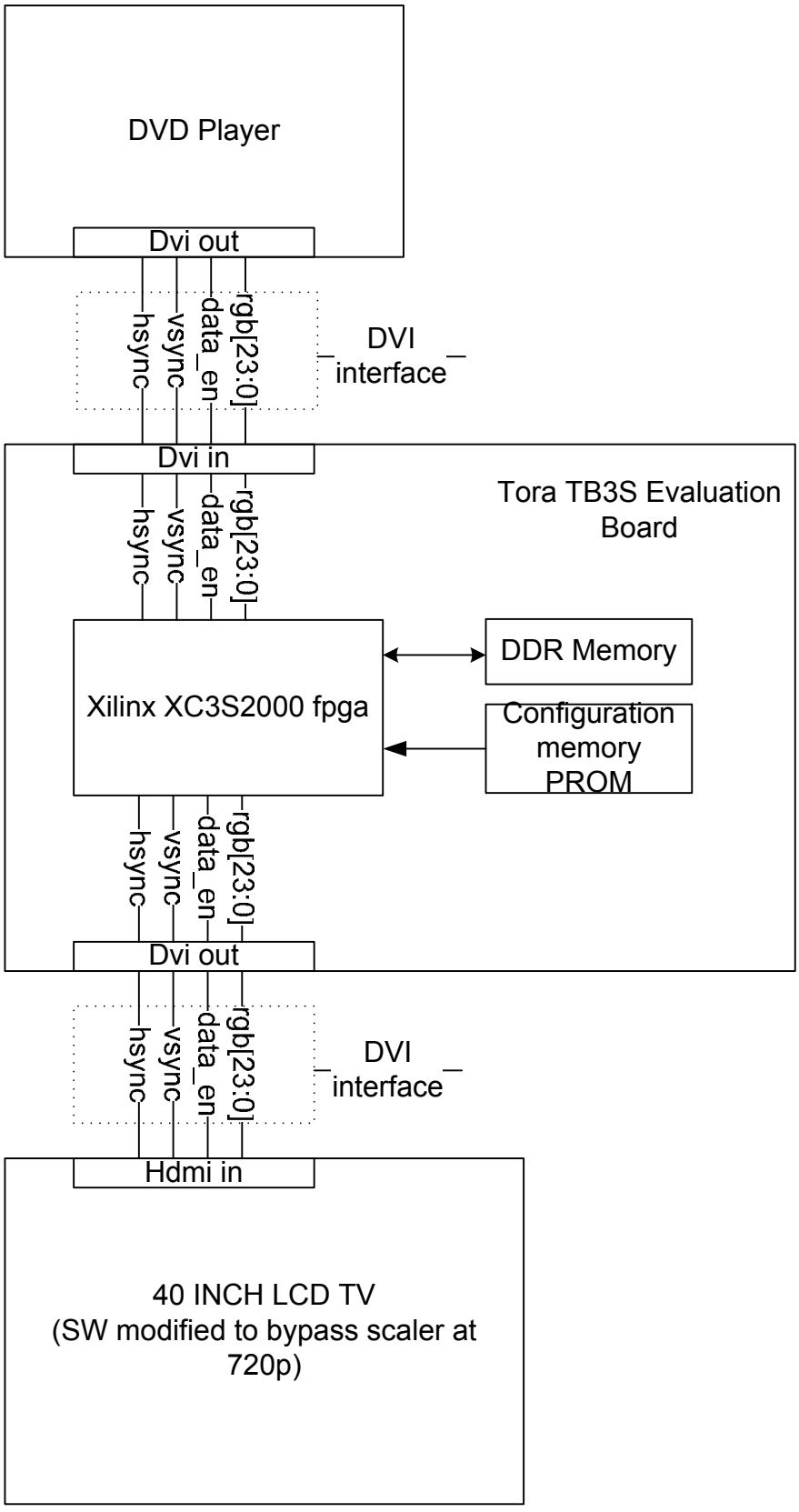

Figure 6-4 : Real time test platform

7. CONCLUSION AND FUTURE WORK

7.1 Concluding Remarks

In this thesis, an area efficient, resource shared, pipelined architecture was proposed for a training based content adaptive video resolution up conversion algorithm. Floating point C model of the modified RS algorithm in [1] was converted to its fixed point model, considering the target FPGAs area constraints and the algorithm's performance. A resource shared architecture was proposed by running the classification, feature extraction and interpolation stages, four times the input pixel clock. The multipliers and adders in the FE, CL and IN units were shared to reduce the area. The proposed architecture was written in RTL level VHDL and verified with functional simulations prior to logic synthesis. Verified design was mapped to FPGA after several synthesis efforts, and several modifications on the pipeline stages to meet the tight timing constraints imposed by the real time nature of the video processing application. The design was mapped to a low cost FPGA and the mapped design was tested on a TV platform bypassing the scaler of the TV for the particular test resolutions. The performance of the algorithm was satisfactorily better than an industry standard, bicubic based scaler. To the author's best knowledge, there exists no hardware architecture for any of the advanced content adaptive resolution up-conversion algorithms listed in [3]. This study demonstrates that better visual quality can be obtained using an advanced scaling method, at a low cost FPGA. The design in this study was funded by Vestel, Vestek R&D, and the results of the study were demonstrated in IFA 2006 Consumer Electronics fair. Results of this work is accepted for publication in "*Proceedings of the IEEE 2007 Workshop on Signal Processing Systems(SIPS 2007)".*

7.2 Future Work

Scaler can be referred as the heart of the video processor inside a display unit. The difficulty of the scaler design is due to the necessity of performing a complex interpolation algorithm for better video quality and the necessity to support multiple input and output resolutions. In this study, the interpolation algorithm's complexity was the key issue in the implementation, and the main concern was the algorithm's core rather than its interface. The resolution conversion designed in this study supports several scaling ratios ranging from 1 to 2, but does not support resolution up conversion for scaling ratios greater than 2. Furthermore, even if the scaling ratio is ranging from 1 to 2, several modifications need to be performed in order to support different standards at runtime. One other point that was not discussed on

this study was video down-conversion. Although not so common, sometimes the display unit may have to downscale the image instead of up scaling. The future work of this study includes solving the issues mentioned in order to produce an industrial video scaler chipset, from the proposed architecture.

REFERENCES

[1] **Akgun, T., Altunbaşak, Y., and Arici, T.**, 2006. Method and apparatus for enhancing the resolution of a digital image, *European Patent, No:* 06251104.3, dated March 2006.

[2] **Smith, S.W.**, 1999. The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, San Diego.

[3] **Zhao, M.**, 2006. Video enhancement using content adaptive least mean square filters, *Phd Thesis*, Eindhoven University of Technology, Eindhoven.

[4] **Oppenheim, A.V., Schafer, R.V., and Buck, J.R.**, 1999. Discrete Time Signal Processing, Prentice Hall, New Jersey.

[5] **Jahne, B.**, 1997. Digital Image Processing, Concepts, Algorithms, and Scientific Applications, Springer-Verlag, New Jersey.

[6] **Lehmann, T.M., Gönner, C., and Spitzer, K.**, 1999. Survey : Interpolation methods in medical image processing, *IEEE Transactions on Medical Imaging*, **18**, 1049-1075.

[7] **Keys, R.G.**, 1981. Cubic convolution interpolation for digital image processing, *IEEE Transactions on Acoustics Speech, Signal Processing*, **29**, 1153-1160.

[8] **Meijering, E.**, 2003. A note on cubic convolution interpolation, *IEEE Transactions on Image Processing*, **12**, 477-479.

[9] **Mitchell, D.P. and Netravali, A.N.** 1988. Reconstruction filters in computer graphics, *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, **22**, pp. 221-228.

[10] **Unser, M.**, 1999. Splines: a perfect fit for signal and image processing, *IEEE Signal Processing Magazine*, **16**, 22-38.

[11] **Maelend, E.**, 1988. On the comparison of interpolation methods, *IEEE Transactions on Medical Imaging*, **7**, 213-217.

[12] **Th`evenaz, P., Blu, T., and Unser, M.**, 2000. Image interpolation and resampling, in *Handbook of Medical Imaging, Processing and Analysis*, pp. 393-420, Academic Press, San Diego CA.

[13] **Th`evenaz, P., Blu, T., and Unser, M.**, 2000. Interpolation revisited, *IEEE Transactions on Medical Imaging*, **19**, 739-758.

[14] **Li, X. and Orchard, M.T.**, 2001. New edge-directed interpolation, *IEEE Transactions on Image Processing*, **10**, 1521-1527.

[15] **Ratakonda, K. and Ahuja, N.** 1998. POCS based adaptive image magnification, *Proceedings of the IEEE International Conference on Image Processing* **3**, pp. 203-207.

[16] **Greenspan, H., Anderson, C.H., and Akber, S.**, 2000. Image enhancement by nonlinear extrapolation in frequency space, *IEEE Transactions on Image Processing*, **9**, 1035-1048.

[17] **Kondo, T., Fujiwara, T., Okumura, Y., and Node, Y.**, 2001. Picture conversion apparatus, picture conversion method, learning apparatus and learning method, *US-patent*, dated November 27, 2001.

[18] **Atkins, C.B., Bouman, C.A., and Allebach, J.P.** 2001. Optimal image scaling using pixel classification, *Proceedings of the IEEE International Conference on Image Processing*, **3**, pp. 864-867.

[19] **Plaziac, N.**, 1999. Image interpolation using neural networks, *IEEE Transactions on Image Processing*, **8**, 1647-1651.

[20] **Go, J., Sohn, K., and Lee, C.**, 2000. Interpolation using neural networks for digital still cameras, *IEEE Transactions on Consumer Electronics*, **46**, 610-616.

[21] **Carotenuto, R., Pappalardo, M., and Sabbi, G.**, 2002. Spatial resolution enhancement of ultrasound images using neural networks, *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, **49**, 1039-1049.

[22] **Tegenbosch, P. and Hofman, M.B.** 2004. Improving nonlinear up-scaling by adapting to the local edge orientation, *Proceedings of the SPIE Visual Communications and Image Processing*.

[23] **Morse, B.S. and Schwartzwals, D.** 2001. Image magnification using level-set reconstruction, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, **1**, pp. 333-340.

[24] **Kondo, T. and Kawaguchi, K.**, 1995. Adaptive dynamic range encoding method and apparatus, *US-patent*, dated August 22,1995.

[25] **Freeman, W., Jones, T., and Pasztor, E.**, 2002. Example-based super-resolution, *IEEE Computer Graphics and Applications*, **22**, 56-65.

[26] **Moon, T.K.**, 1996. The expectation-maximization algorithm, *IEEE Signal Processing Magazine*, **13**, 47-60.

[27] **Bellers, E.B. and Caussyn, J.** 2003. A high definition experience from standard definition video, *Proceedings of the SPIE*, **5022**, pp. 594-603.

[28] **EIA/CEA-861B**, 2002. A DTV Profile for Uncompressed High Speed Digital Interfaces, *Electronic Industries Alliance Technology Strategy & Standards Department*.

[29] **Hudson, R.D., Lehn, D.I., and Athanas, P.M.** 1998. A run time reconfigurable engine for image interpolation, *Proceedings of the IEEE Symposium on FPGAs for Custom Configurable Computing Machines*, Napa, California.

[30] **Aurelio, M., Maganda, N., and Arias-Estrada, M.O.** 2005. Real time fpga-based architecture for bicubic interpolation : An application for digital image processing *IEEE International Conference on Reconfigurable Computing and FPGAs(ReConfig'05)*.

[31] **Xiao, J., Zou, X., Liu, Z., and Guo, X.** 2006. Adaptive interpolation algorithm for real time image resizing, *Proceedings of the International Conference on Innovative Computing Information and Control*, **2**, pp. 221-224.

[32] **Ramachandran, S. and Srinivasan, S.** 2003. Design and FPGA implementation of a video scalar with on-chip reduced memory utilization, *Euromicro Symposium on Digital Systems Design*, pp. p206.

BIOGRAPHY

Muzaffer Barış Uyar was born in Alanya, TURKEY in 1979. He graduated from Alanya Anatolian High School in 1997. In 2002, he received B.Sc. degree in Electronics and Communication Engineering from Istanbul Technical University.He started an M.Sc programme in Computer Science in Istanbul Technical University, Informatics Institute in 2003. He had worked in ST Microelectronics Istanbul Design Center during 2003-2005 and has been working as a design engineer at Vestek Electronic R&D Center since 2006. His research interests are in the field of digital VLSI design, energy efficient processor design and IC design for video processing.