

APPLICATIONS *for* DIGITAL SIGNALS

***(in Digital Speech Processing, Digital Image Processing and
Consumer Electronics)***

Manish Pradhan

APPLICATIONS *for* **DIGITAL SIGNALS**

CONTENTS

PREFACE **GLOSSARY**

- 1. Digitized Signals, *page 1***
- 2. Digital Signal Processing, *page 5***
- 3. Digital Filters, *page 15***
- 4. Adaptive Filters, *page 23***
- 5. Frequency Transforms, *page 31***
- 6. Applications in Speech Processing, *page 43***
- 7. Applications in Image Processing, *page 57***
- 8. Modulation Techniques, *page 75***
- 9. Audio Compression Techniques, *page 93***
- 10. Video Compression Techniques, *page 115***
- 11. Packetization of Audio, Video, & Data for Delivery over Networks, *page 141***
- 12. Conditional Access (Security) Systems, *page 179***
- 13. Symmetric & Asymmetric Encryption/Decryption Algorithms, *page 191***
- 14. Typical DVB SETTOP Box Application, *page 205***
- 15. Typical DVD Playback Application, *page 221***
- 16. Typical DOCSIS® Cable Modem Application, *page 231***
- 17. Cable TV SETTOP with Integrated DOCSIS® Cable Modem, *page 243***
- 18. Desktop & Embedded Applications, *page 245***
- 19. Microsoft® Windows Based Educational Applets, *page 259***

REFERENCES & WEBSITES, *page 261* **INDEX, *page 265***

PREFACE

The goal of this book, and its accompanying educational only software applets, is to provide, a under-graduate OR a graduate student of computer science and electrical engineering, an idea as to what applications are possible in the areas of Digital Signal Processing, Digital Image Processing and Consumer Electronics. This book should be a good starting point for an ENGINEER adventuring out into these areas.



Manish Pradhan

GLOSSARY

ADPCM	Adaptive Digital Pulse Code Modulation
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
CELP	Code Excited Linear Predictive Coding of Speech
CDMA	Code Division Multiple Access
CM	Cable Modem
CMTS	Cable Modem Termination System (Cable Headend)
COFDM	Coded Orthogonal Frequency Division Multiplexing
CPU	Central Processing Unit
CSS	Content Scrambling System for DVD
DAC	Digital to Analog Converter
DCT	Discrete Cosine Transform
DDRAM	Double Data Rate DRAM (synchronously fetches data on both edges of clock)
DES	Data Encryption Standard
DHCP	Dynamic Host Control Protocol
DVD	Digital Video (or Versatile) Disk
EAV	End of Active Video
ES	Elementary Stream used in MPEG
FIR	Finite Impulse Response filter
HDTV	High Definition Television
IIR	Infinite Impulse Response filter
ISI	Inter Symbol Interference
JPEG	Joint Photographic Experts Group
LMS	Least Mean Squares algorithm
LPC	Linear Predictive Coding
MAC	Media Access Control Layer
MDCT	Modified Discrete Cosine Transform
MPEG	Motion Picture Experts Group
PCM	Pulse Code Modulation
PES	Packetized Elementary Stream used in MPEG
PHY	Physical Access Layer (Access Layer to Physical Media)
PS	Program Stream used in MPEG
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RLS	Recursive Least Squares algorithm
ROI	Region of Interest
SAV	Start of Active Video
SAW	Surface Acoustic Wave Filter
SDRAM	Synchronous DRAM (synchronously fetches data on single edge of clock)
SDTV	Standard Definition Television
SNMP	Small Networks Management Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TDMA	Time Division Multiple Access
TS	Transport Stream used in MPEG
UDP	Universal Datagram Protocol
VSF	Vestigial Side Band Modulation
YUV	a.k.a. YCbCr Component Video, where YCbCr are Luminance, Blue and Red components

DIGITIZED SIGNALS

Today signals can be digitized at any stage (Radio Freq, Intermediate Freq and at Base-band Freq) because of the availability of ADC's which perform digitization at very high frequencies (in the Giga-Hertz Range).

A typical (modulation) application goes through the following 5 stages,

- Baseband [Hz or low MHz range] Signal (Data, Audio, Video) which is processed by computers or digital hardware →
- **Convert this DIGITAL Baseband Signal into the ANALOG domain using a Digital to Analog Converter (DAC)**
- Upconvert to Intermediate [high MHz range] Frequency (1st modulation stage – modulate a mid freq carrier with baseband signal) →
- Upconvert to Radio [GHz range] Frequency (2nd modulation stage – modulate a high freq carrier with the output from 1st modulation stage) →
- Transmit the signal (from the 2nd modulation stage) over any media like terrestrial, cable and satellite.

A typical (de-modulation) application goes through the following 5 stages,

- Receive the Radio Frequency signal transmitted over the media (terrestrial, cable and satellite) →
- Down Convert or demodulate (1st demodulation stage) to Intermediate Frequency →
- Down Convert or demodulate (2nd demodulation stage) to Baseband Frequency →
- **Convert this ANALOG Baseband Signal into the DIGITAL domain using a Analog to Digital Converter (ADC)**
- This digital baseband signal (Data, Audio, Video) can now be processed by computers or digital hardware.

Now in most applications only the baseband signal (Hz or low MHz range) is kept in a digitized state. But with the availability of high performance ADC's the Intermediate frequency or even the radio frequency signal can be digitized and processed by digital hardware.

Digitization of a signal works as shown next in figure 1. An analog signal having a single frequency (F_o) component is digitized with a sampling rate greater than $2 \cdot F_o$ (Nquist freq) to prevent aliasing in the frequency domain. Digitization is a process whereby a analog signal is converted (quantized) into finite discrete steps.

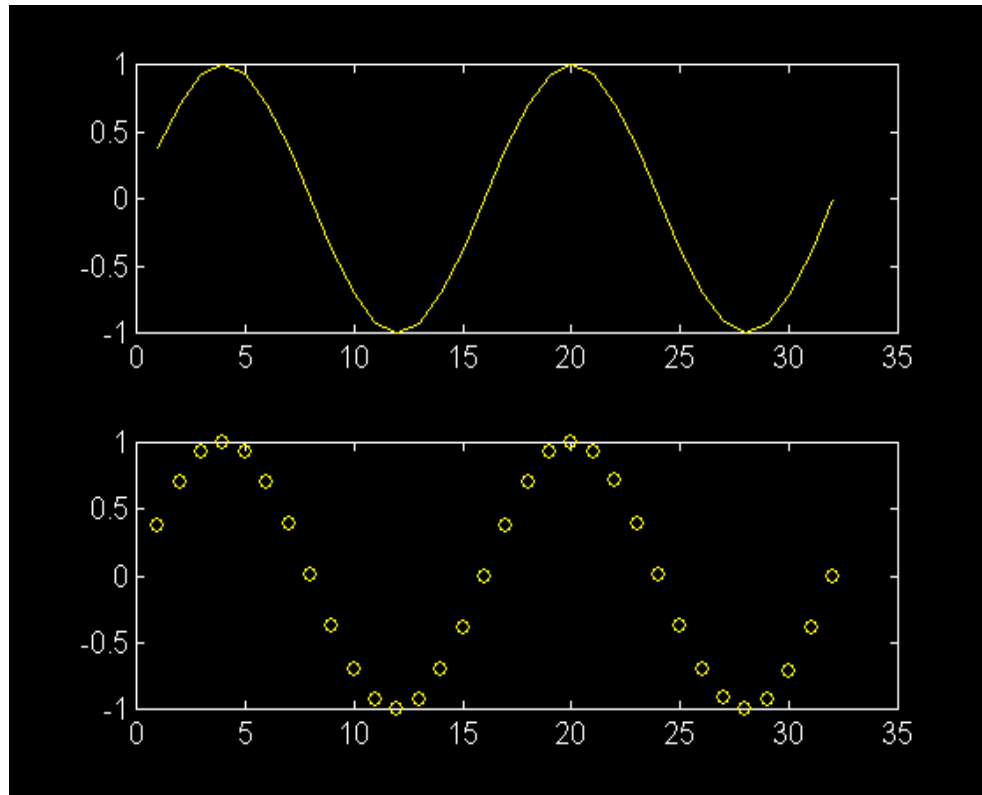


Figure 1. Signal with Single Freq in its Analog and Digital Domain

Aliasing is a phenomena, by which if the sampling freq (F_s) of a signal is not greater than twice its highest freq component (say F_o) then multiples (aka. harmonics) of that freq. component ($2*F_o$, $3*F_o$, $4*F_o$, etc.) will alias or overlap each other making it difficult for any filter to sort out the overlapped regions.

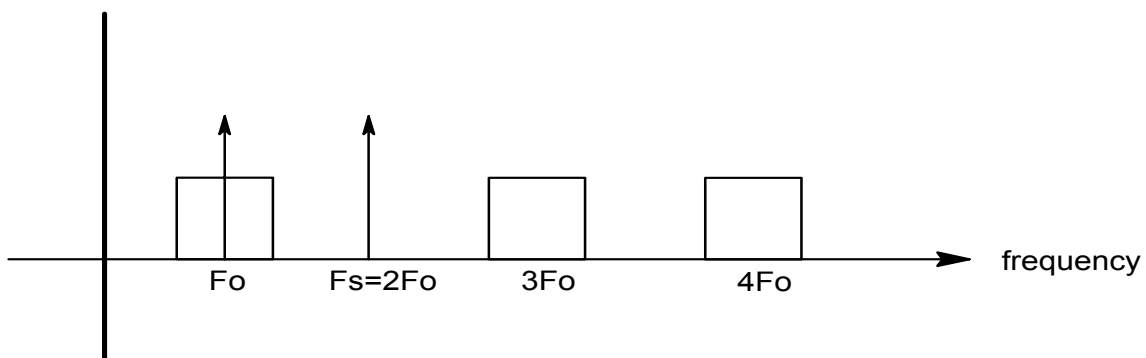


Figure 2. Signal with Single Freq (F_o) in Freq Domain

The following applications employ digitization of analog signals,

- **Sound cards** digitize analog audio from a microphone or some other analog speech source, send it to a DSP or a PC for further processing.
- **Frame grabbers** digitize analog video from a camera source into frames which are then stored in memory for further processing by any digital hardware.
- **Digital scopes** digitize analog signals which can then be analyzed & processed in the digital domain.
- **SETTOP boxes** which use QPSK, QAM, VSB, COFDM modulation techniques digitize the demodulated analog signal at (RF, IF, Baseband) frequencies for further processing by the digital hardware to provide the user with watch-able content like news, movies, songs, etc.
- **Digital Cell phones** which use GSM, CDMA, etc. modulation techniques digitize the demodulated signal at (RF, IF, Baseband) frequencies for further processing by the inbuilt digital signal processor (DSP) to provide the user with speech or data signals.
- **Cable Modems** which use QAM as their modulation technique, digitize the received downstream signal (or channel) for further processing to provide the user with a ability to access the internet (for downloading webpages, files, etc.)

The whole idea behind the **analog to digital transformation** of any signal is to put the signal in a form, such that it can be easily read and manipulated by machines (computers, digital hardware, Digital Signal Processors, Digital Image Processors, etc) which understand and use the binary language of 1's and 0's for their computing.

There are 2 types of ADC's,

- **Flash ADC** – here all bits (N bits of resolution) for each signal sample are generated instantaneously and the ADC is clocked by $F_s \geq 2 \times \text{highest frequency component in the signal}$.
- **Sigma Delta single bit ADC** – the output of this type of ADC is a single bit, over sampled by a factor $F_s \times x$ (where x is the over sampling ratio). The idea behind this technique is to produce 1 bit at $F_s \times x$ and then LPF and integrate these bits over subsequent stages at $F_s \times x/2$, $F_s \times x/4$, ..., etc. till you get the required resolution of N bits. This technique keeps the ADC hardware minimum, but is only feasible for signals having low sampling rates like speech.

DIGITAL SIGNAL PROCESSING

Digital Signal Processing is the science by which a digital signal (data, speech or image) is sent to a digital hardware block which performs a transformation on it in the time $[h(m)]$ or frequency $[h(f)]$ domain to provide a new transformed (or processed) signal.

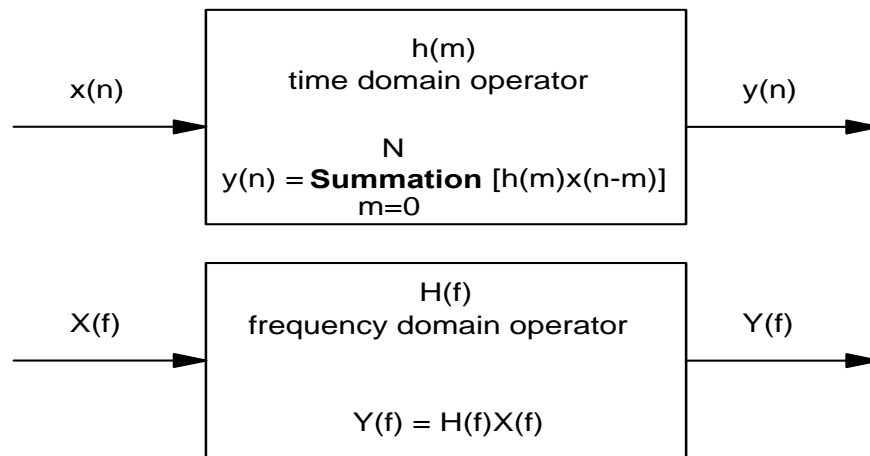


Figure 3. Time and Frequency Domain Digital Operator

As shown above a time domain signal $[x(n)]$ series is convolved with a time domain operator $[h(m)]$ having N points (where each point is spaced apart by 1 delay of sampling time interval) to obtain $y(n)$. The Associated freq domain operator is $H(f)$ and the freq domain output $Y(f)$ is obtained when the operator is multiplied with the freq domain input signal $X(f)$. Thus convolution in the time domain is equivalent to multiplication in the time domain.

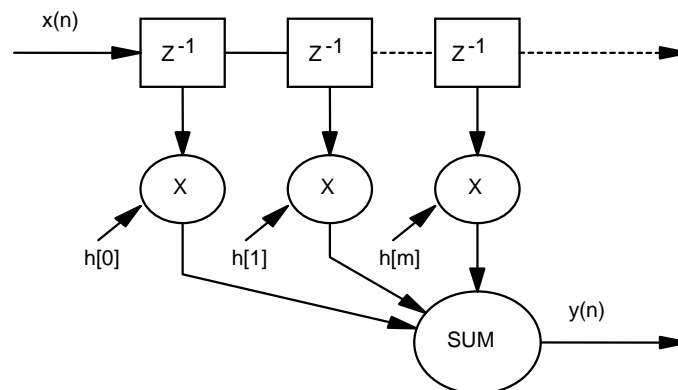


Figure 4. Typical Digital Operator $h(m)$

A **Z transform** of a time domain signal $x(n)$ is the frequency domain representation of the signal on a unit circle (with real and imaginary axes). The Z transform has 2 variants (FFT & CZT).

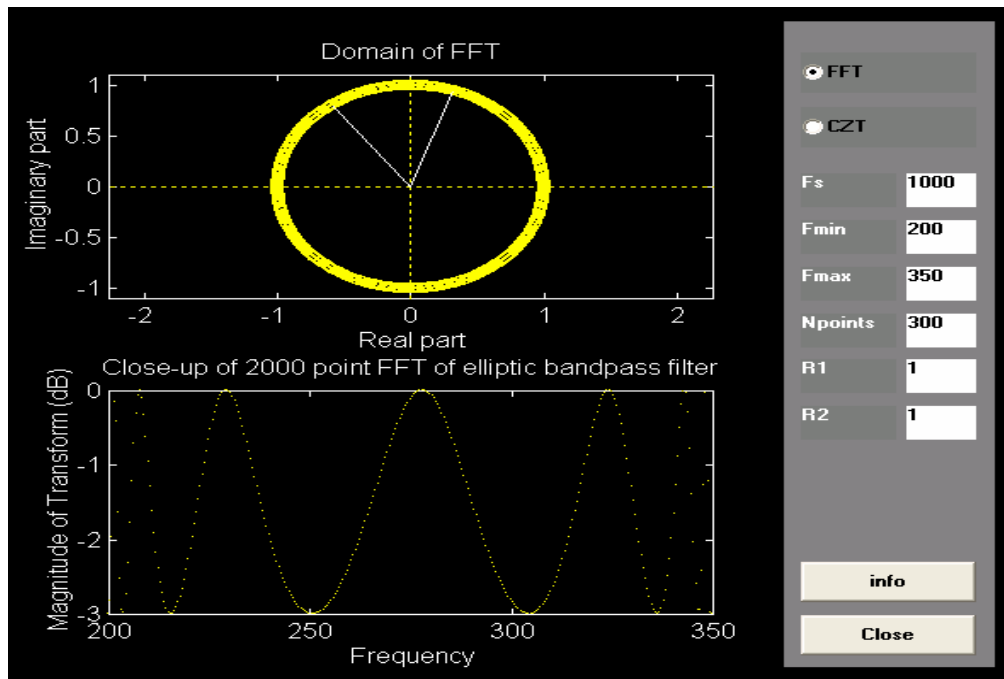


Figure 5. FFT - the Fast Fourier Transform - This computes the Z-transform on equally spaced points around the unit circle as shown.

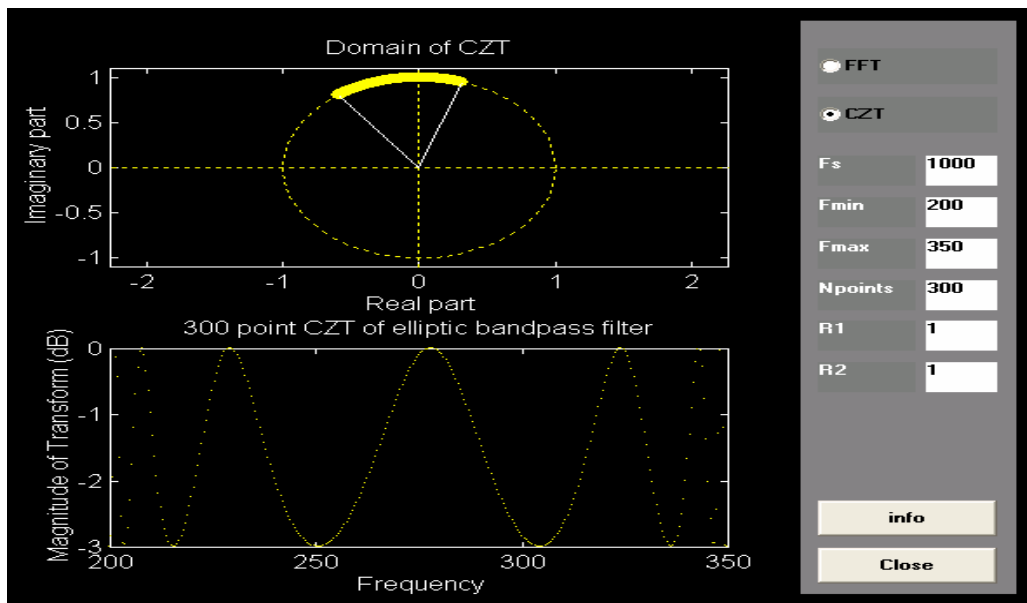


Figure 6. CZT - the Chirp-Z Transform - This computes the Z-transform on a spiral or "chirp" contour. The contour is defined by initial frequency F_{min} and radius $R1$, and final frequency F_{max} and radius $R2$.

In both these cases (**FFT & CZT**), the upper sub plot shows the transform domain on the unit circle, and the lower sub plot shows the transform of a band-pass digital filter on the points within the wedge shown on the upper plot.

F_s is the sampling frequency, F_{min} and F_{max} define a "wedge" on the unit circle, N_{points} is the number of Z-transform points computed on the unit circle in the wedge defined by F_{min} and F_{max} .

With the **FFT**, the length of the transform is $(N_{points} * F_s) / (F_{max} - F_{min})$, which computes N_{points} points in the range F_{min} to F_{max} . If a small frequency range is needed the **CZT** is much more efficient because it can "zoom-in" on the frequency range you need.

Some aspects of Digital Speech Processing

Digital Speech Processing is a branch of digital signal processing where the digitized signal sent to the digital hardware (e.g. Digital Signal Processor) is human speech. In most applications it becomes necessary to process digitized speech before delivery to the human aural system. Some of them are,

- **Removal of certain speech frequency components using standard digital filters like low pass, high pass, band pass or notch filters**
- **Line Echo cancellation using adaptive filters to remove echoes in speech transmitted over a phone line or over a IP network**
- **Acoustic Echo cancellation using adaptive filters to remove echoes in speech transmitted over the air (e.g. teleconferencing)**
- **Building Equalizers using adaptive filters to remove inter-symbol interference when voice is transmitted across a channel**
- **Using standard digital filter structures to emphasize or de-emphasize certain frequencies in the speech spectrum**

Human Speech contains all frequencies (tones) in the range from 10Hz to 20KHz and reproducing all the frequencies in this range accurately (without distortion) in the digital domain is a challenging task. Depending on the capture device (A/D) each speech sample can have different resolutions (8/12/16/20/24 bits with 24 bits being common on some hi-fidelity speech or audio processing systems). Also Speech or Digital Audio programs today can have upto 6 separate single speech channels (Left, Right, Center, Left Surround, Right Surround and a low frequency or subwoofer channel) typically found in digital Dolby® AC3 on DVDs.

Usually speech is represented in the digital domain by spectrograms (2D spectrum plots) which can depict each spoken word. If $x(n)$ is digitized speech (captured via a microphone, etc.) then a spectrogram splits $x(n)$ into several overlapping segments, processes each segment with a hanning or hamming window function and forms the columns of another 2D series $y(n)$, zero-pads the

elements in each column to N points over which a FFT is applied. Thus each column of the series $y(n)$ is now transformed into an estimate of the short-term, time-localized frequency content of the signal $x(n)$. The spectrogram is a plot of the frequency transformed columns of $y(n)$ vs. its rows (time) as shown in figure 7 and figure 8.

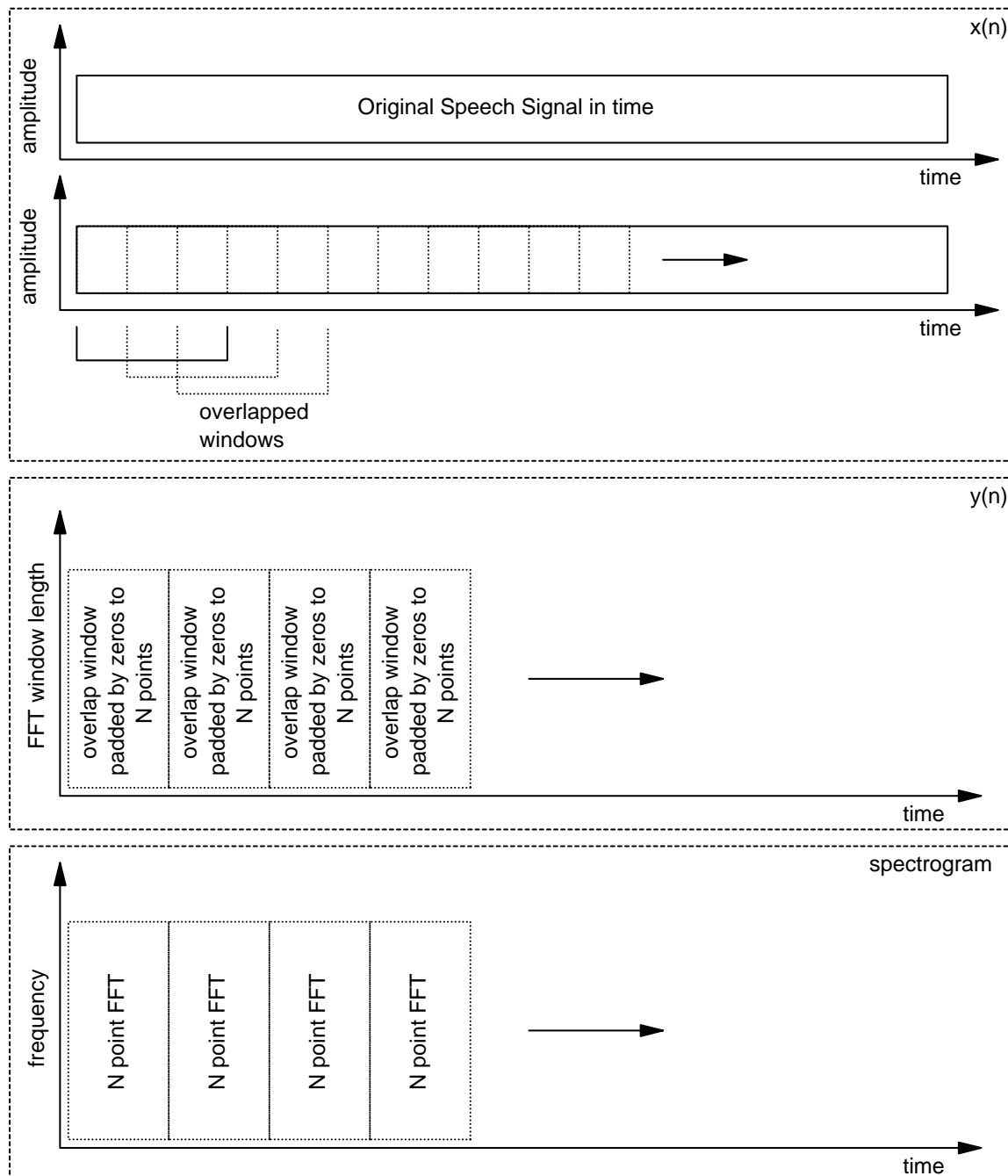


Figure 7. Construction of a Spectrogram to represent digitized speech

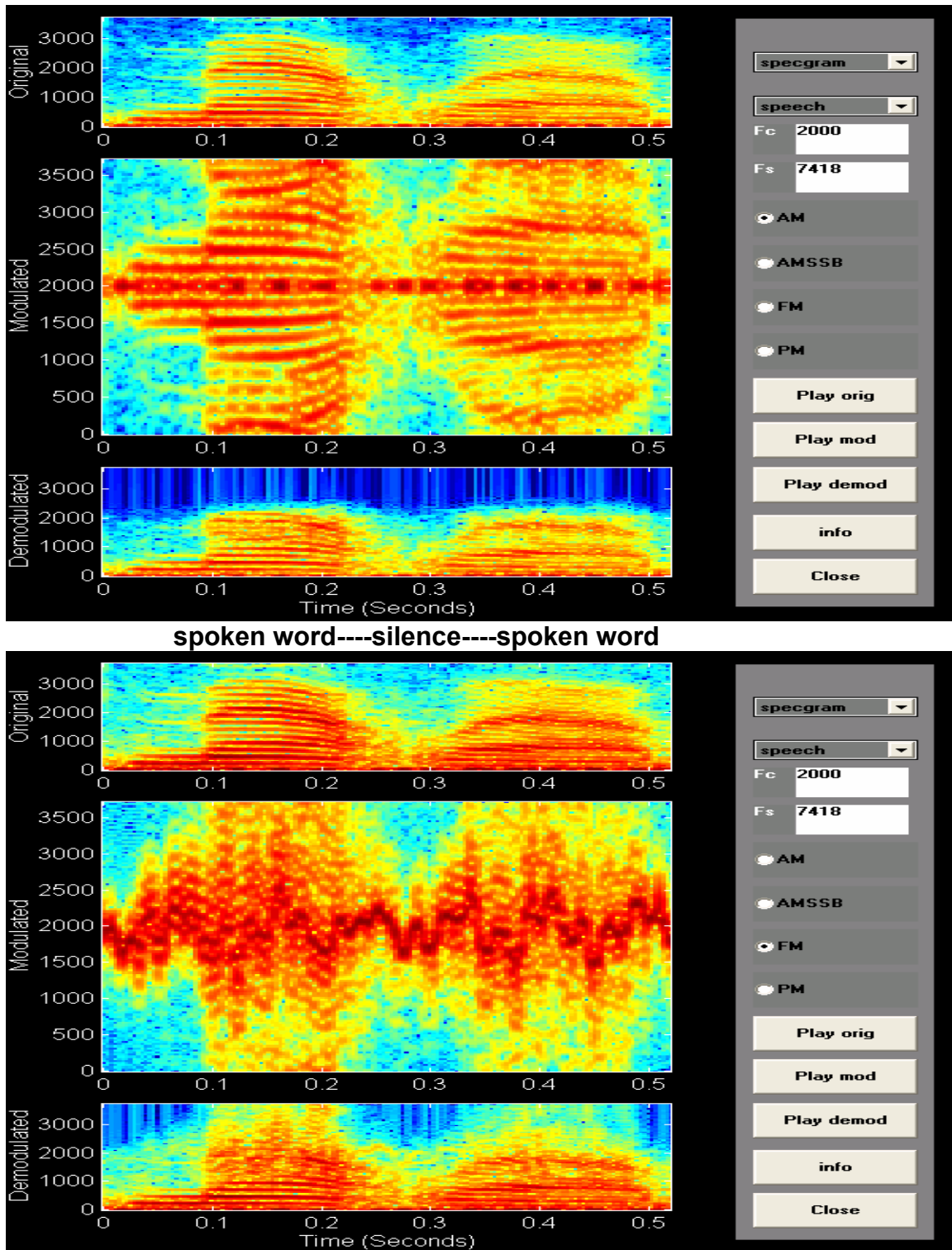


Figure 8. Spectrogram (time-xaxis vs. frequency-yaxis plot) of a typical digitized speech signal (e.g. two consecutive spoken digitized words separated by silence) modulated & demodulated by a Computer using Amplitude and Frequency modulation techniques.

Some aspects of Digital Image Processing

Digital image processing can be divided into 2 parts,

- **image processing of 2 dimensional images or video**

2 dimensional images are usually formed by a camera source which scans (interlaced – field by field or progressive format – frame by frame) a live scene, sends this analog information to the digitizer which converts this information into RGB or a YUV true color pixels (where each RGB/YUV pixel is 24 bits thus giving a total of 16 million colors). The digitized image or video thus obtained depending on the scanning format is called interlaced (typical TV display) or progressive (typical computer monitor display) image.

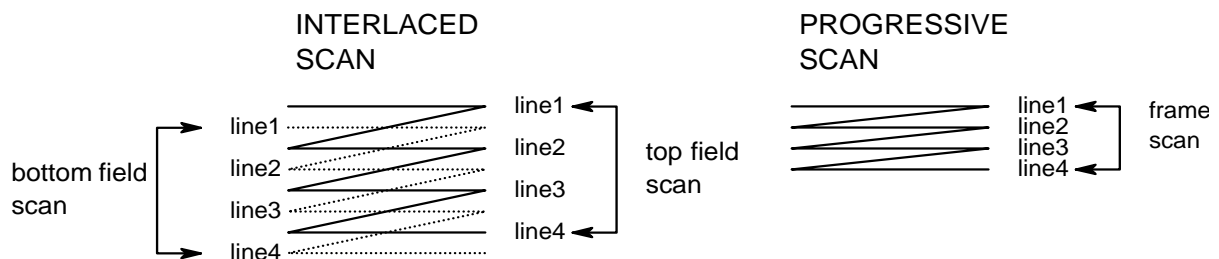


Figure 9. Camera scan generating Interlaced & Progressive pictures

- **image processing of 3 dimensional images or 3D data sets**

Three dimensional images (or a 3D data set) are usually formed in the medical profession by CT (computer aided tomography), MRI (magnetic resonance imaging), PET (positron emission tomography) and Ultrasound devices which can capture non-invasively (using Xrays, gamma rays or high freq sound waves) the 3D aspects or internal structure of any body part (head, stomach or any internal organ).

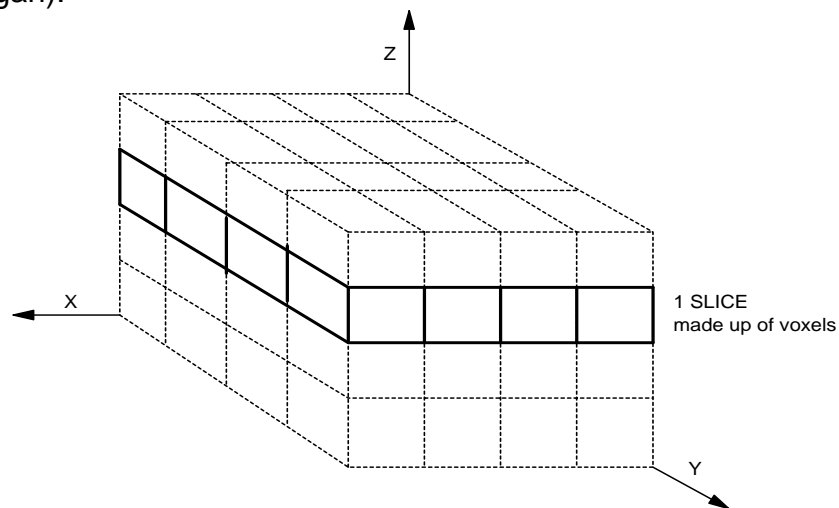


Figure 10. 3D data set used in medical imaging (slices made up of voxels)

The resulting image is a 3D array made up of voxels. Each voxel has a x,y,z representation in space (x & y being the 2D space, z being the voxel thickness) and has a accuracy of 8 bits (or 256 gray levels).

In both the 2D and 3D cases image processing becomes important to process and manipulate the images digitized before presenting them to the human visual system. Some of the image processing techniques which apply to both 2D and 3D images are listed below.

- **Point Processes** – these are image processing functions which apply only to the current 2D image or the current slice in a 3D image. These processes need to be applied to the gray scale pixels OR all the Red, Blue, Green OR all the Y, U, V pixels in a color image frame. Some of them are listed below,
 - a. **Image Negation**
 - b. **Image Brightness**
 - c. **Image Contrast stretching**
 - d. **Image Equalization using histograms**
 - e. **Pseudo Coloring of Gray Scale Images**
- **Area Processes** – these are image processing functions that use groups of pixels to find information about an image. Basically a area process uses a neighborhood of MxN pixels around the current pixel of interest to process that pixel. Some of them are listed below,
 - a. **Image embossing or giving depth to a 2D image**
 - b. **Detection of edges in a image (sobel, laplace)**
 - c. **Tracing contours in a image depending on intensities**
 - d. **Applying spatial filters (low pass or high pass)**
 - e. **Convolving Images with masked (MxN) filters for removal or enhancing of certain features**
- **Frame Processes** – these are image processing functions which use 2 or more image frames in a image sequence while operating on the current frame. Some of them are,
 - a. **Blitter functions which take 2 images, combine them into a third image using a digital operator (AND, OR, XOR, etc.).**
 - b. **Motion detection functions which use the motion information between blocks within (2 or more frames) to perform interpolation operations (e.g. interlaced to progressive translation) on the current image OR generating motion estimation vectors in compression techniques.**

- **Geometric Processes** – these are image processing operators which change the orientation or position of the pixels in an image (without changing their intensity values). Some of them are listed below,
 - a. **Image Scaling (Interpolation & Decimation)**
 - b. **Image Rotation or Mirroring**
 - c. **Changing the Aspect Ratio of a Image ($4/3 \leftrightarrow 16/9$)**

Some 2D image processing functions are shown below,

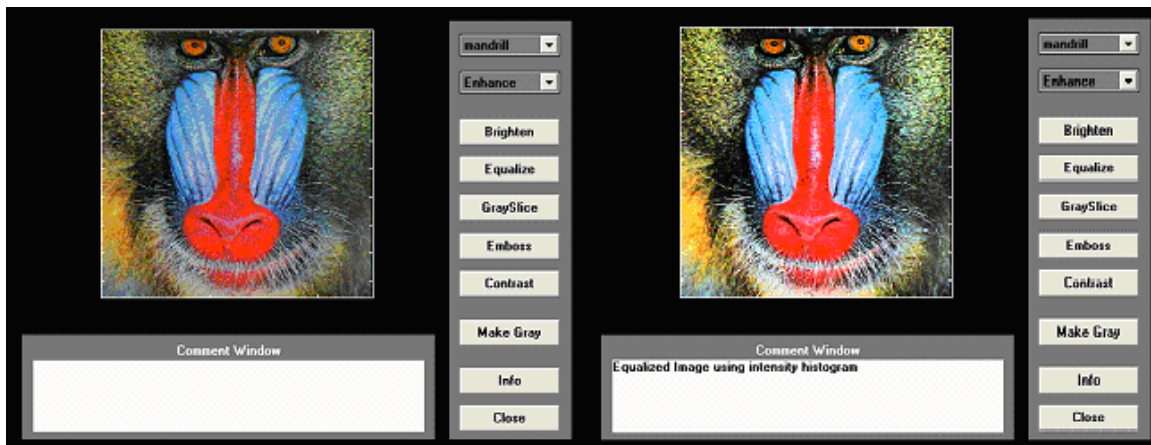


Figure 11. Original & Histogram Equalized Image

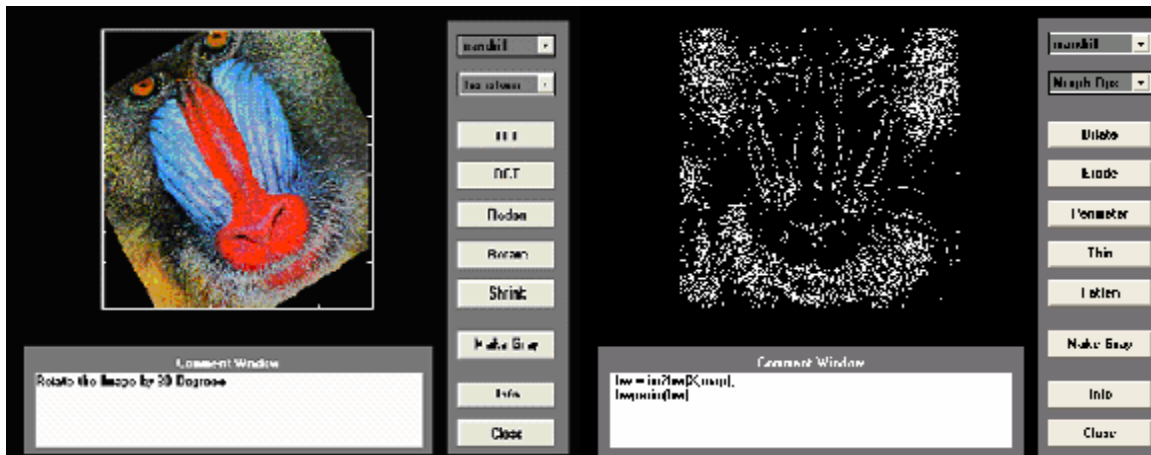


Figure 12. Rotated & Edge Detected Image

A Typical 3-Dimensional volumetric data set used in medical imaging applications is shown below. Most 3D digital operators are really 2D operators applied to each 2D slice across the entire volume of the image. The 2D operator applied to every slice may be the same or different.

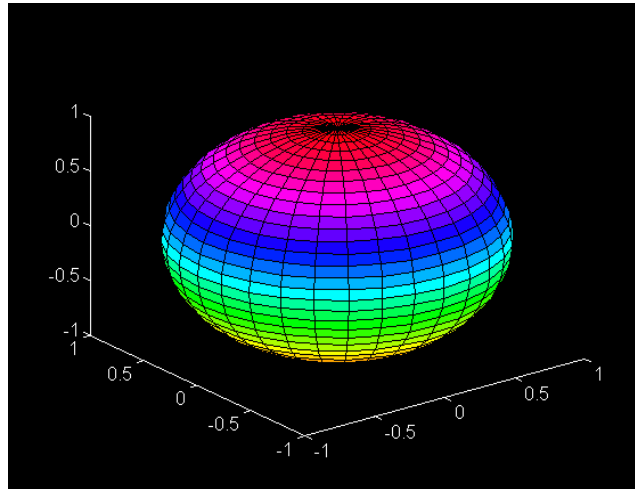


Figure 13. Three Dimensional Voxel Data Set

DIGITAL FILTERS

There are 3 types of Digital Filters,

- **FIR (Finite Impulse response)** – this filter has a output response which is of finite duration. These positive features of these filters are a simple architecture, stability and exhibit linear phase vs. frequency (phase angle is a linear function of frequency). The negative features are a inability to achieve a sharp cutoff magnitude response in the frequency domain. Most of these filters do not have a feedback structure, only a feedforward structure.
- **IIR (Infinite Impulse response)** – this filter has a output response which may persist for a infinite period of time. The IIR filter does not have phase linearity w.r.t frequency, but has an improved magnitude response in the frequency domain (sharp cutoffs). This class of filters have feedback structure wherein the output feeds back into the input.
- **FIR/IIR Cascade Structures** – these filters have a cascade structure consisting of FIR (feedforward) and IIR (feedback) structures, which lets them model complex systems better.

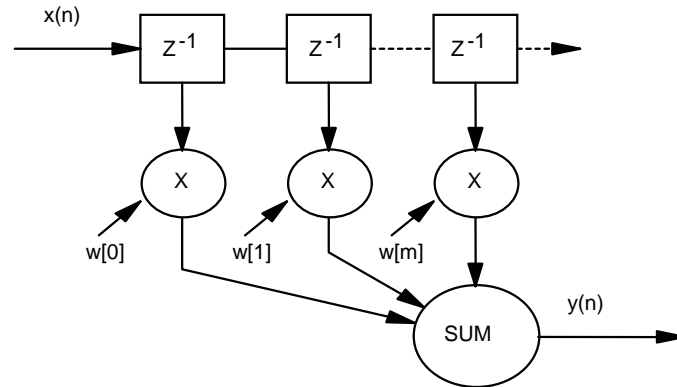
FIR filters are further classified into,

- **Windowing functions like hanning, hamming, blackman and kaiser**
- **Parks McClellan optimal equi-ripple FIR filter**
- **Linear phase FIR filter using least-squares error minimization**
- **Linear Convolution of two sequences**
- **Cross-correlation and Autocorrelation of sequences**
- **Interpolation & Decimation using FIR filters**

Any 1D FIR filter has the below structure. Given a 1D input $x(n)$, the goal now is to provide the weights (w) for different types of filters to obtain the output $y(n)$.

$$y(n) = \sum_{m=0}^{N-1} [\text{weights}(m) * x(n-m)], \text{ or } H(z) = \sum_{n=0}^{N-1} [h(n) * z^{-n}]$$

As you can see a typical FIR response contains ALL ZEROS (no POLES). Some of the commonly used FIR filters are listed next,



Hanning & Hamming

$\text{weights}(m) = a + (1-a) \cdot \cos \left[\frac{2\pi m}{N} \right]$, when $|m| \leq \frac{(N-1)}{2}$
 $\text{weights}(m) = 0$, when m is something else

$a = 0.54$ gives a hamming window, $a = 0.5$ gives a hanning window

Blackman

$\text{weights}(m) = 0.42 + 0.5 \cdot \cos \left[\frac{2\pi m}{(N-1)} \right] + 0.08 \cdot \cos \left[\frac{4\pi m}{(N-1)} \right]$, when $|m| \leq \frac{(N-1)}{2}$
 $\text{weights}(m) = 0$, when m is something else

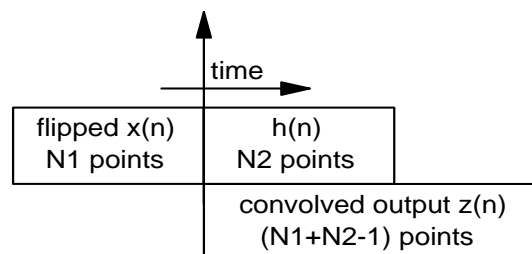
Kaiser

$\text{weights}(m) = I_0 \left\{ B \left[1 - \left(\frac{2m}{(N-1)} \right)^2 \right]^{1/2} \right\}$, when $|m| \leq \frac{(N-1)}{2}$
 $\text{weights}(m) = 0$, when m is something else

I_0 represents the zeroth order Bessel function and the parameter B establishes a trade off between the main lobe width and the side lobe width.

Linear Convolution

Linear Convolution is used to retrieve the actual or wanted signal series from the input or recieved signal series which was passed over any media. If $x(n)$ is the input time series with N_1 points, $h(n)$ is unwanted time series (media channel inv. response) with N_2 points, then convolving $x(n)$ with $h(n)$ gives us the wanted time series $z(n)$ with N_1+N_2-1 points or elements. Basically both the time series are padded with zeros to obtain (N_1+N_2-1) points, then time series $x(n)$ is flipped over, shifted over the time series $h(n)$ one element at a time, multiplied and accumulated to get each element in $z(n)$. This is the identical to padding the time series with zeros to get (N_1+N_2-1) points, DFT each one of them, and multiply them.



$z(n) = \sum_{m=0}^{N_1+N_2-1} [x(n) \cdot h(n-m)]$, where the time series $z(n)$ has a total of (N_1+N_2-1) elements.

Cross-correlation

Cross-Correlation is used to find the similarity between 2 time series, say $x(n)$ and $y(n)$. The most useful application of cross-correlation is to extract "wanted" features from a obtained time series and is denoted by $R_{xy}(k)$. Its another form of convolution.

$$R_{xy}(k) = \frac{1}{N} * \sum_{i=0}^{N-1} [x(i)*y(i-k)]$$

Auto-correlation

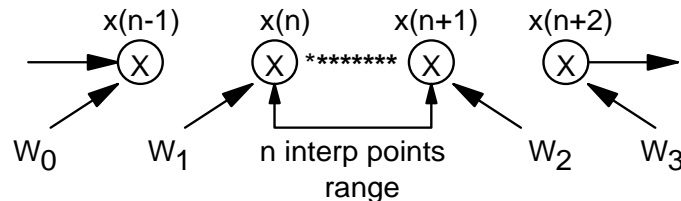
Auto-Correlation is used to find how well a signal (time series), say $x(n)$, maintains its structure over a given interval of time. The auto-correlation function is a cross-correlation between the signal and itself. It can be used to find important features of a signal and is denoted by $R_x(k)$. Its another form of convolution.

$$R_x(k) = \frac{1}{N} * \sum_{i=0}^{N-1} [x(i)*x(i+k)]$$

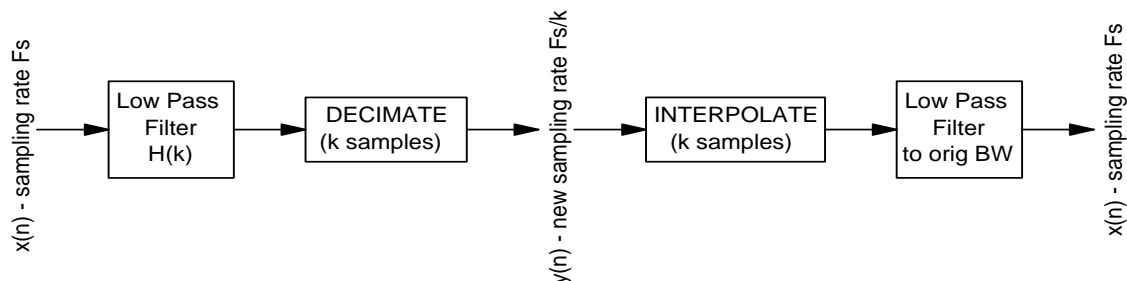
Interpolation & Decimation

All most all interpolation algorithms (due to improvements in processing power) use Cubic Spline which are basically 3rd order FIR filters which use 4 original elements in a sequence for generating a new element. Interpolation (or increasing the sample rate) involves the addition of more elements to a sequence $x(n)$ followed by a low pass filter.

A interpolated point, $y(n) = W_0*x(n-1) + W_1*x(n) + W_2*x(n+1) + W_3*x(n+2)$, where the weights selected depend on how many elements need to be added between 2 consecutive elements of $x(n)$. These can easily be generated using any math package supporting 3rd order curve fitting.



Decimation (or decreasing the sample rate) involves first passing a low pass filter over a sequence $x(n)$ before removal of some of its elements. Decimation is relatively simple after low pass filtering because it involves throwing away unwanted samples from the original sequence. Decimation ratios of 1:2, 1:3, 1:4, 1:5, etc. are achieved by removing "x samples" from the original set.



Some 1D FIR filters with their order, passband, stopband and ripples in these bands are shown below,

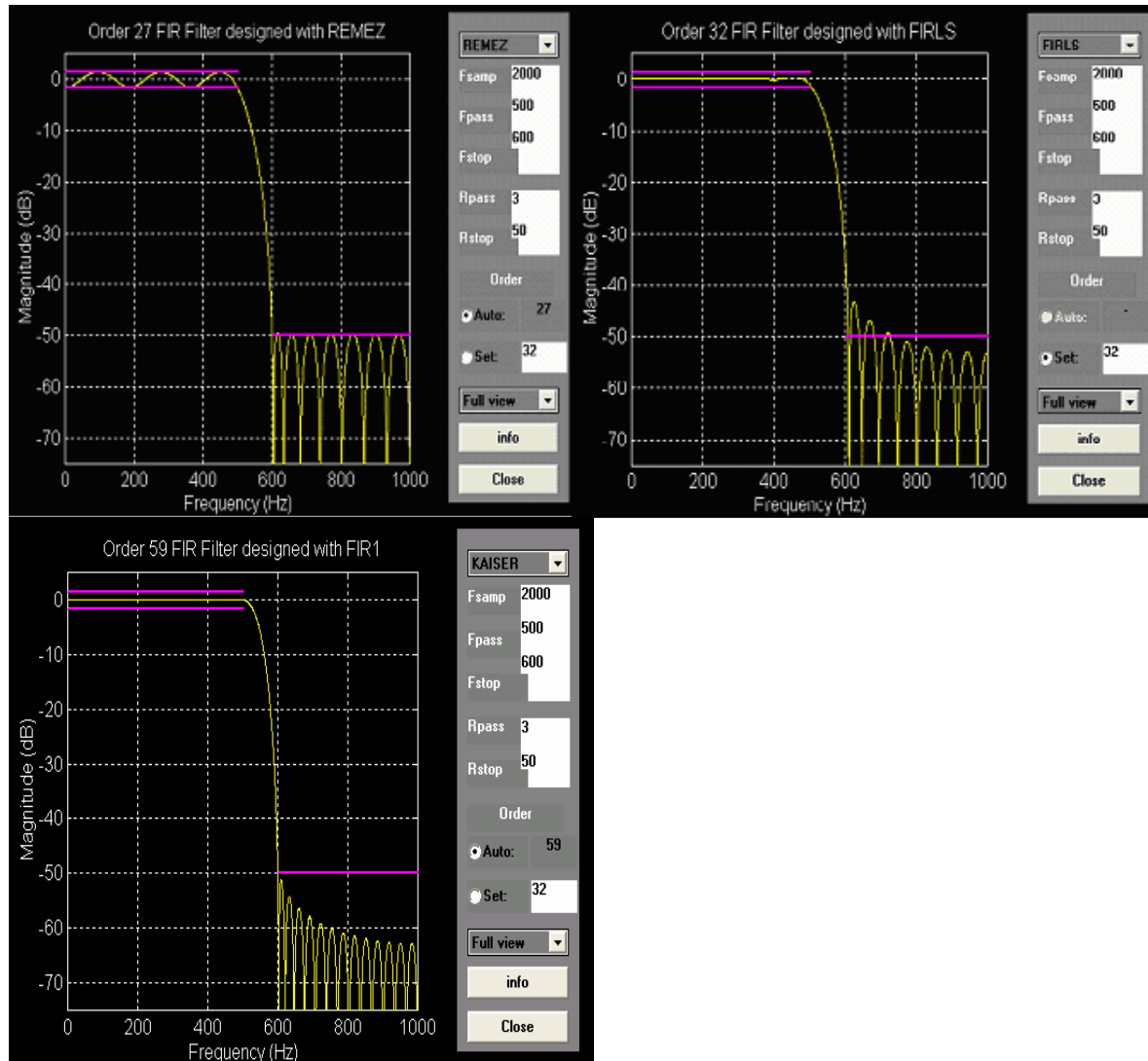


Figure 14. Some FIR Filters

IIR & FIR/IIR filters are further classified into,

- **Butterworth**
- **Chebyshev**
- **Elliptical**

Any 1D IIR filter or FIR/IIR cascade filter has the below structure,

$$H(z) = B(z)/A(z) = \text{Summation}_{n=0}^M [b_n * z^{-n}] / 1 + \text{Summation}_{n=1}^N [a_n * z^{-n}]$$

thus in the time domain,

$$y(n) + a_1*y(n-1) + a_2*y(n-2) + \dots + a_N*y(n-N) = b_0*x(n) + b_1*x(n-1) + b_2*x(n-2) + \dots + b_M*x(n-M)$$

$$y(n) = [b_0*x(n) + b_1*x(n-1) + b_2*x(n-2) + \dots + b_M*x(n-M)] - [a_1*y(n-1) + a_2*y(n-2) + \dots + a_N*y(n-N)]$$

As you can see the IIR or the cascaded FIR/IIR structure contains a feedforward path (zeros on the unit circle) and a feedbackward path (poles on the unit circle). The goal now becomes to find the weight vector a (having N points) and b (having M points) to develop the filter response.

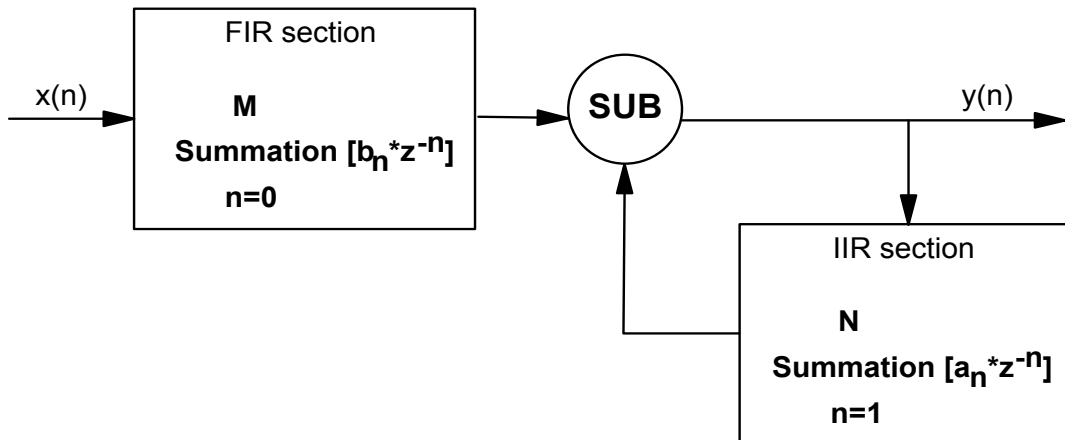


Figure 15. FIR/IIR filter structure

Some typical 1D IIR filters are shown below, as you can see the ORDER (points or taps) needed to achieve the same sharp cutoffs as their FIR counterparts are much smaller.

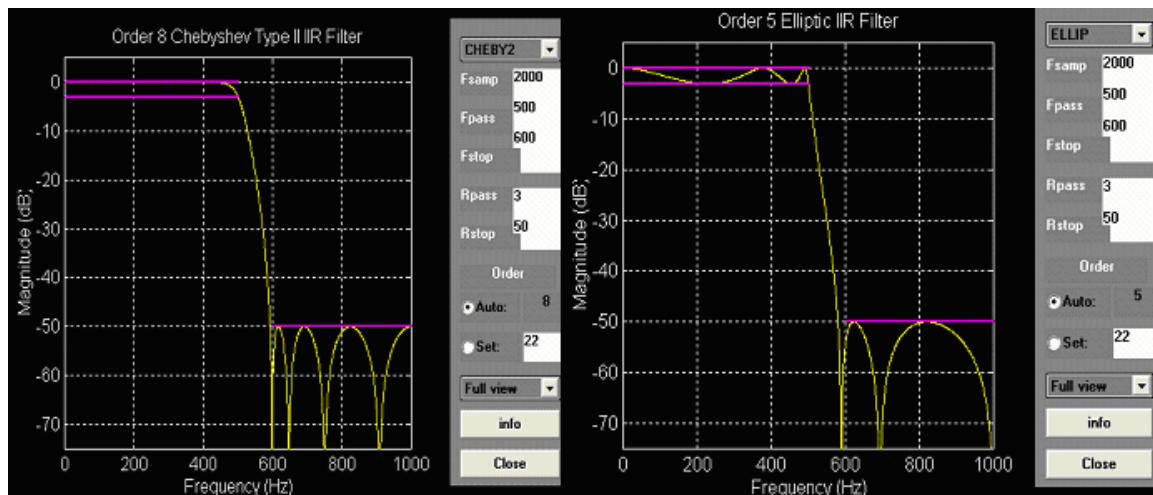


Figure 16. Some IIR Filters

Two dimensional Filters are usually MxN masks which are superimposed on a image and specific algorithms are applied to the pixels in that mask.

- a. **Median filter**
- b. **Sobel edge detector**
- c. **Embossing filter**
- d. **Averaging filter**
- e. **Low and high pass filters**
- f. **Cross-correlation**
- g. **Auto-correlation**
- h. **Bi-cubic interpolation**

Below is a 3x3 convolution mask which is used for typical masked filters,

A	B	C
D	E	F
G	H	I

1. In a Median filter a “quick sort” algorithm is used to find the intensity values of all the 9 points (PtA, PtB, PtC, PtD, PtE, PtF, PtG, PtH, PtI) in accending order and the “median” value is used to replace point E. This mask is then passed over the entire image column by column and row by row. The median filter effectively removes certain types of noise in a image (e.g. salt noise).
2. In a Sobel filter the below algorithm is used with a certain user “THRESHOLD”, to detect edges within that image.

```
LineAEIAveBelow = (PtD+PtG+PtH)/3;
LineAEIAveAbove = (PtB+PtC+PtF)/3;
LineAEIMaxDif = abs (LineAEIAveBelow-LineAEIAveAbove);
```

```
LineBEHAveBelow = (PtA+PtD+PtG)/3;
LineBEHAveAbove = (PtC+PtF+PtI)/3;
LineBEHMaxDif = abs (LineBEHAveBelow-LineBEHAveAbove);
```

```
LineCEGAveBelow = (PtF+PtH+PtI)/3;
LineCEGAveAbove = (PtA+PtB+PtD)/3;
LineCEGMaxDif = abs (LineCEGAveBelow-LineCEGAveAbove);
```

```
LineDEFAveBelow = (PtG+PtH+PtI)/3;
LineDEFAveAbove = (PtA+PtB+PtC)/3;
LineDEFMaxDif = abs (LineDEFAveBelow-LineDEFAveAbove);
```

```
/* Find the maximum value of the absolute differences from the four possibilities. */
```

```
MaxDif = max (LineAEIMaxDif, LineBEHMaxDif);
MaxDif = max (LineCEGMaxDif,MaxDif);
MaxDif = max (LineDEFMaxDif,MaxDif);
```

if (MaxDif >= THRESHOLD), then the “current original pixel” is replaced with a “constant” value, this pixel is now on the “edge function – a edge” for that image.

3. An embossing filter which gives depth to a image uses the following algorithm,

Pixel = PtA*(+1)+PtB*(+1)+PtD*(+1)+PtF*(-1)+PtH*(-1)+PtI*(-1);
Pixel = Pixel + 128;

Now replace the old Pixel with this new Pixel.

4. A typical low pass filter has the following convolution kernel,

Pixel = PtA*(1/9)+PtB*(1/9)+PtC*(1/9)+PtD*(1/9)+PtE*(1/9)+PtF*(1/9)+PtG*(1/9)+ PtH*(1/9)+ PtI*(1/9);

5. A typical high pass filter has the following convolution kernel,

Pixel = PtA*(-1)+PtB*(-1)+PtC*(-1)+PtD*(-1)+PtE*(9)+PtF*(-1)+PtG*(-1)+ PtH*(-1)+ PtI*(-1);

6. Bi-cubic interpolation uses the below 4x4 convolution mask which is passed over the entire image to obtain all the “interpolated pixels”. The algorithm in pseudo code follows,

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

```

for (ImageRow=Row, ImageRowI=0; ImageRow<Height; ImageRow++, ImageRowI+=interp)
for (ImageCol=Col, ImageColI=0; ImageCol<Width; ImageCol++, ImageColI+=interp)
{
    //interp is the interpolation ratio in both the horizontal & vertical direction
    //interp_row = 1/interp
    //interp_col = 1/interp

    //offset1 = -1
    //offset2 = 0
    //offset3 = +1
    //offset4 = +2

    for (i=0.0,ii=0; ii<interp; i+=interp_row,ii++)
    for (j=0.0,jj=0; jj<interp; j+=interp_col,jj++)
    {
        Pixel = Pt11*func(offset1-i)*func(j-offset3) + Pt12*func(offset2-i)*func(j-offset3) +
        Pt13*func(offset3-i)*func(j-offset3) + Pt14*func(offset4-i)*func(j-offset3) +
        Pt21*func(offset1-i)*func(j-offset2) + Pt22*func(offset2-i)*func(j-offset2) +
        Pt23*func(offset3-i)*func(j-offset2) + Pt24*func(offset4-i)*func(j-offset2) +
        Pt31*func(offset1-i)*func(j-offset1) + Pt32*func(offset2-i)*func(j-offset1) +
        Pt33*func(offset3-i)*func(j-offset1) + Pt34*func(offset4-i)*func(j-offset1) +
        Pt41*func(offset1-i)*func(j-offset4) + Pt42*func(offset2-i)*func(j-offset4) +
        Pt43*func(offset3-i)*func(j-offset4) + Pt44*func(offset4-i)*func(j-offset4)

        Put this Pixel at (ImageColI+jj, ImageRowI+ii) location on the screen.
    }
}

```

Some 2D filters are shown below,

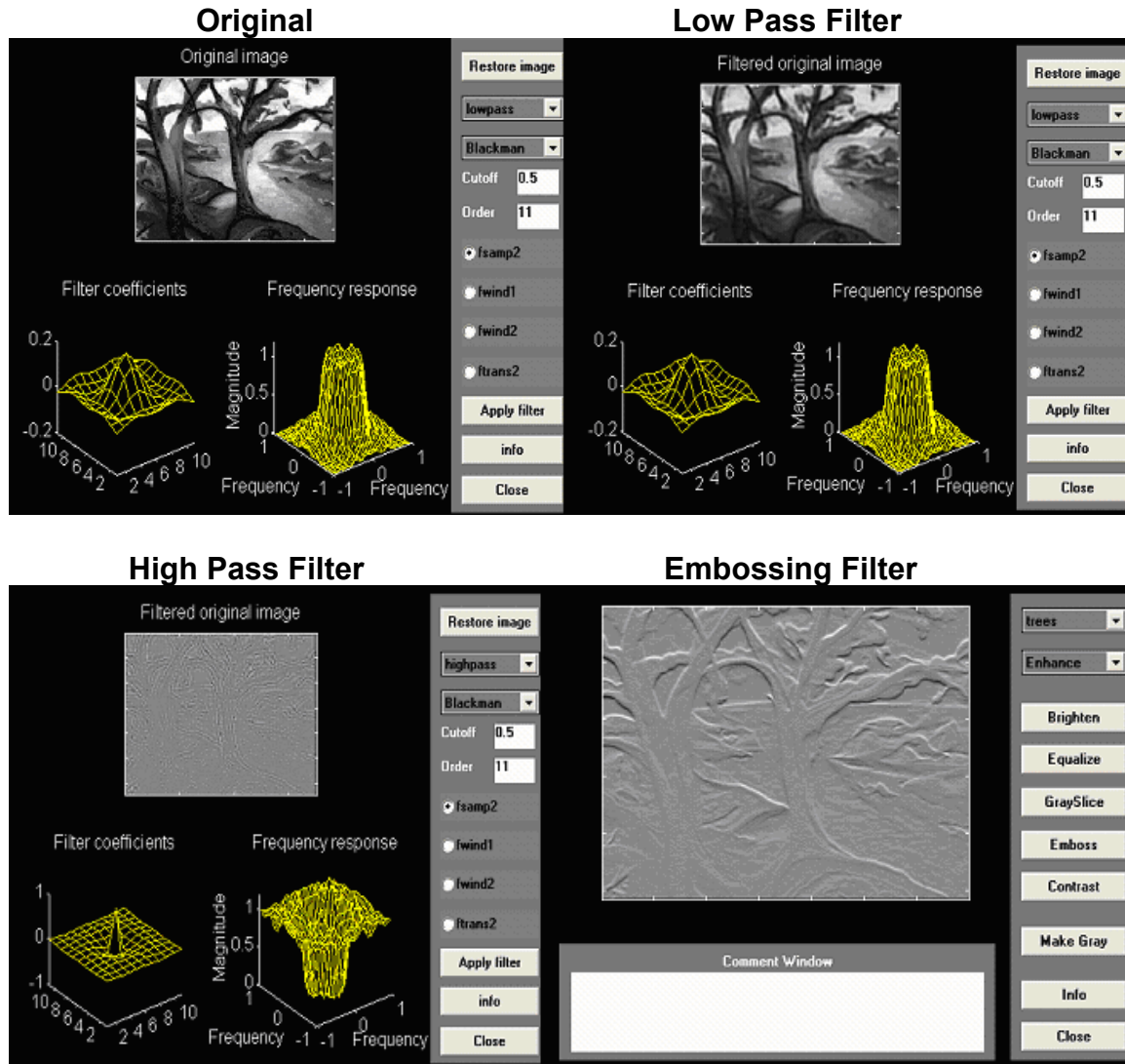


Figure 17. 2D FIR Filters

Notes:

1. For all the above filter cases if the image is colored (RGB or YUV) and not gray scaled, then these operators have to be applied to all the 3 colors or all color components.

ADAPTIVE FILTERS

Adaptive filters are used in a branch of Digital Signal Processing called **Adaptive Signal Processing**. These filters are used to model (imitate) dynamic systems whose behavior (characteristics or response) is **unknown**. Basically an adaptive filter adjusts itself (adjusts its weights using an adaptive algorithm) such that its output is the best least-squares fit to that of the unknown system. An adaptive filter could have both FIR and IIR sections (zeros & poles on the unit circle) which are adapted using individual or separate adaptive algorithms.

We will discuss the following 5 adaptive filter structures,

- **Transversal**
 - a. Asymmetric
 - b. Symmetric
- **FIR/IIR**
- **Lattice**
- **Subband**
- **Frequency domain**

Also we will discuss the following 2 adaptive algorithms,

- **LMS (Least Mean Squares)**
 - a. Normalized least mean square algorithm
 - b. Leaky least mean square algorithm
- **RLS (Recursive Least Squares)**

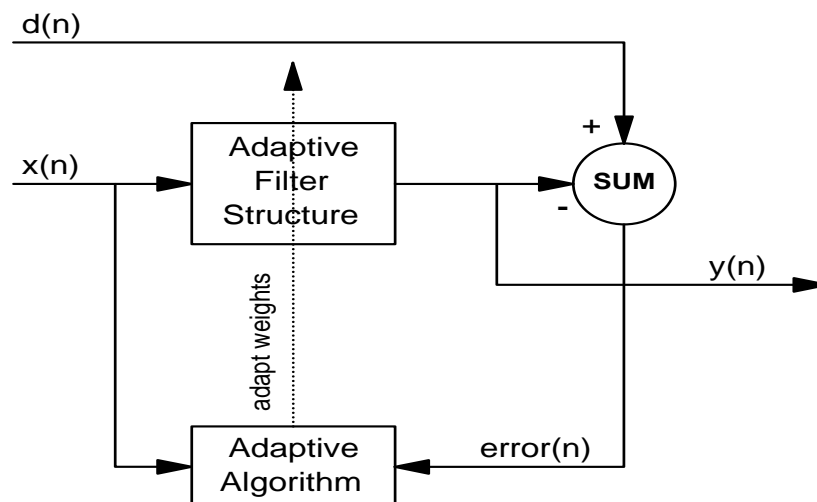
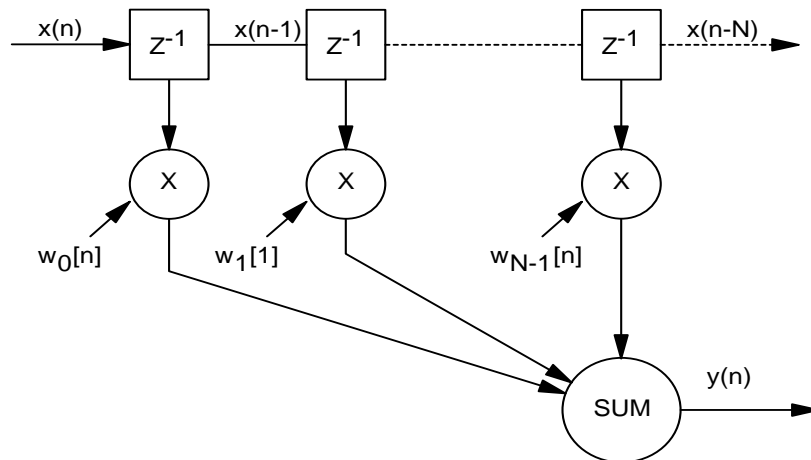


Figure 18. A typical Adaptive Filter System

In figure 18, we show a typical adaptive system which uses a adaptive filter to model the inverse response of a “unknown system”. Basically a adaptive algorithm adjusts the weights of the adaptive filter such that its response makes the output signal $y(n)$ as close as possible to the desired signal $d(n)$ (by minimizing the least squared error).

- **Transversal Filters**, use either un-symmetric or symmetric structures as shown next. These can be quite useful in digital speech processing because they maintain a linear phase response w.r.t frequency (all frequency components or tones in the speech signal are phase shifted or delayed by the same amount – **no phase distortion**)

Transversal filters are all zero FIR filters and very stable as discussed before. The output response of such a filter is the convolution of the input signal $x(n)$ with the adapted weights $w_i(n)$ (or the impulse response)



$$y(n) = \mathbf{w}^T(n) \mathbf{x}^T(n) = \sum_{i=0}^{N-1} [w_i(n) * x(n-i)]$$

where,

Input signal vector, $\mathbf{x}^T(n) = [x(n), x(n-1), \dots, x(n-(N-1))]$

And weight vector, $\mathbf{w}^T(n) = [w_0(n), w_1(n), \dots, w_{N-1}(n)]$

A **symmetric transversal filter** is shown in figure 19. It has a symmetric impulse response around the center tap which results in reduction of the tap order and thus computational complexity.

$$y(n) = \sum_{i=0}^{N/2-1} [w_i(n) * [x(n-i) * x(n-N+i+1)]]$$

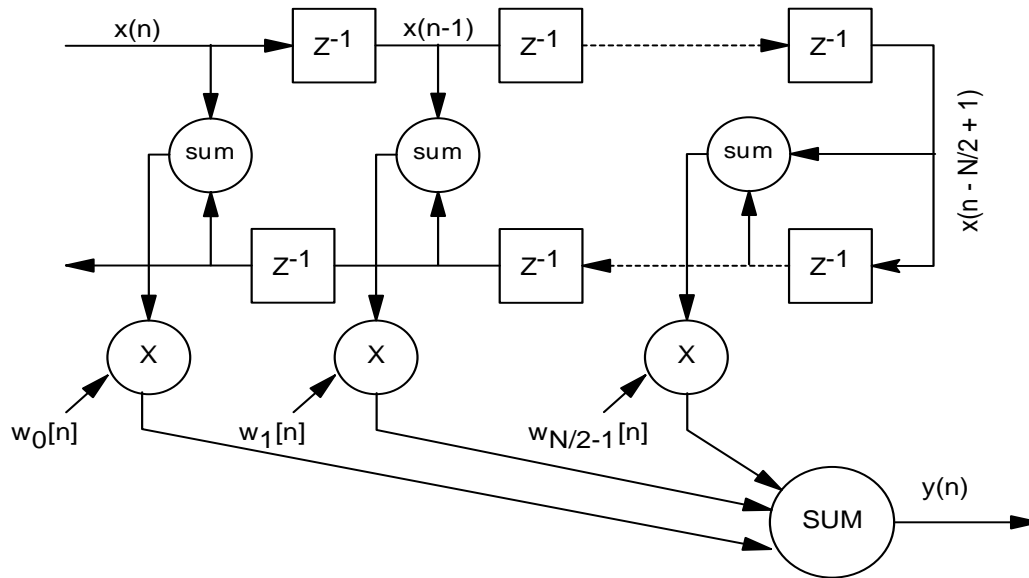


Figure 19. Symmetric Transversal FIR Filter

- A typical **FIR/IIR structure** is shown in figure 15. The response of this filter is given by the combination of the weighted present input samples plus the weighted past input samples. This results in a pole, zero design which makes it possible to achieve sharp cutoffs in the stop band using a small number of taps (weights).
- A **Lattice structure** is a cascade of single tap prediction error filters. This structure as shown in figure 20, splits the signal into sets of forward (f) and backward (b) residual signal samples with delays added to the backward channel. These signals are multiplied by Parcor coefficients, $k(n)$, which correspond to the reflection coefficients of a discrete lattice. These coefficients are equal for digitized sampled data. These structures are very useful in speech coders because they orthogonalize the signal, thus de-correlating the speech.

The calculation of the Parcor coefficients is shown below,

1. $f_0(n) = b_0(n) = x(n)$
2. $f_m(n) = f_{m-1}(n) - [k_m(n) * b_{m-1}(n-1)]$
3. $b_m(n) = b_{m-1}(n-1) - [k_m(n) * f_{m-1}(n)]$
4. $k_m(n+1) = k_m(n) + \mu * [f_m(n) * b_{m-1}(n-1) + b_m(n) * f_{m-1}(n)]$

where $f_m(n)$ is the forward prediction error, $b_m(n)$ is the backward prediction error, $k_m(n)$ is the parcor coef, m is the stage index and M are the number of cascaded stages. "mu" is the step size of the adaptation.

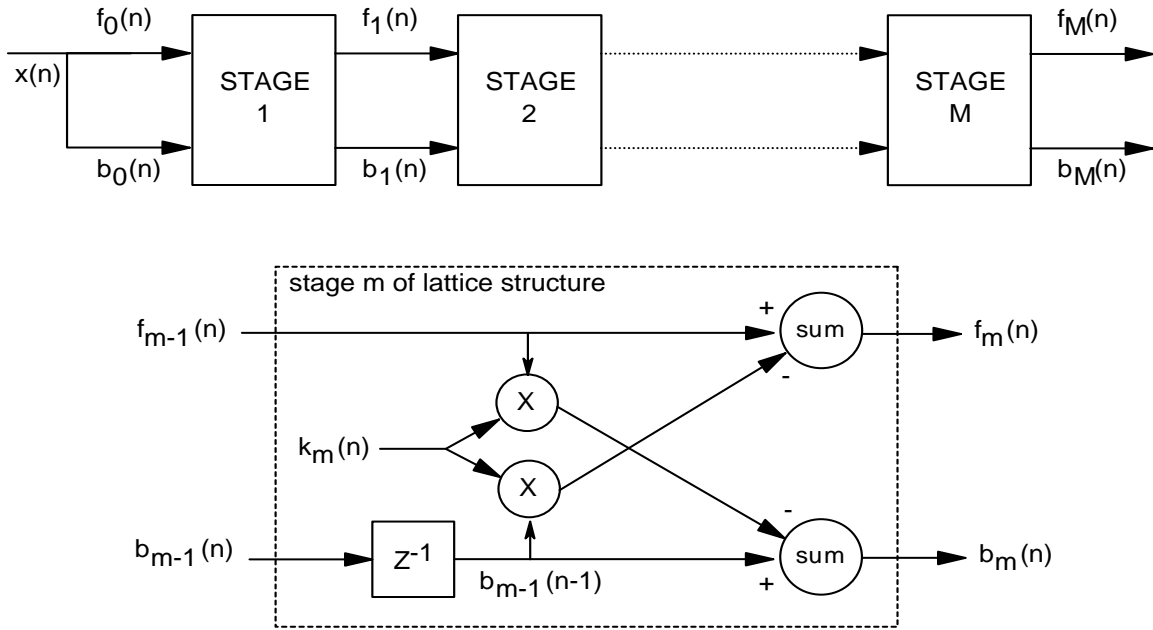


Figure 20. Lattice Filter Structure

$y(n) = \sum_{m=0}^M [g_m(n) * b_m(n)]$, where $y(n)$ is the output of the lattice filter in figure 21.

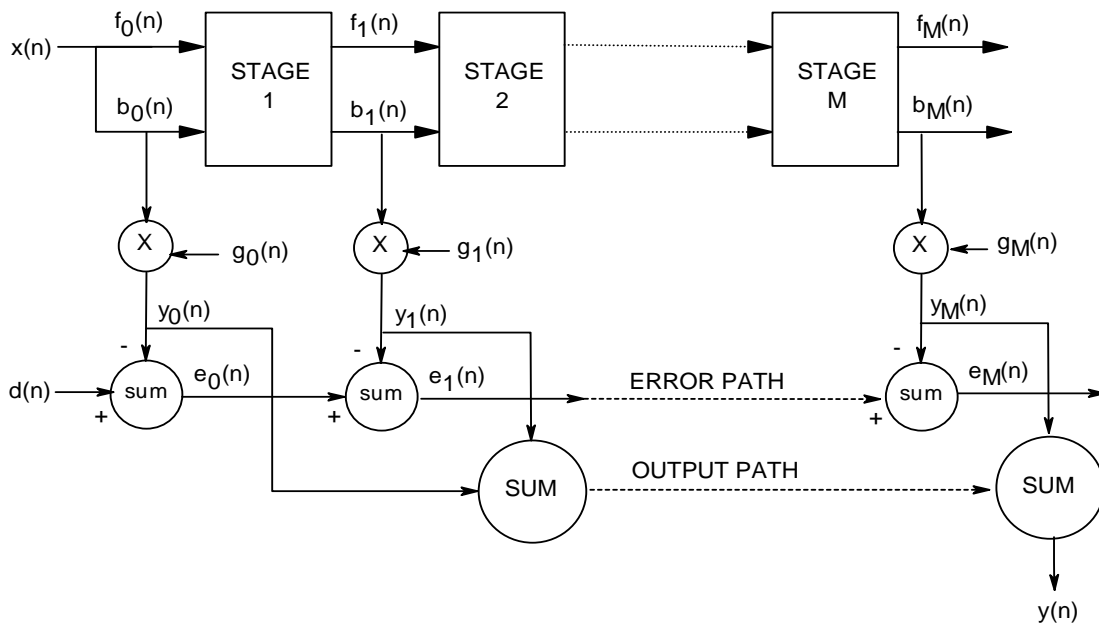


Figure 21. Lattice Error and Output paths in a M stage adaptive filter

- Subband filters** basically split the input signal using lowpass & highpass **Quadrature Mirror Filters** (QMF – prevents aliasing of freq components from one band into the other) into two frequency subbands (decimation or freq rate downconversion). Each subband output is now treated as an individual signal $x_{\text{subband}}(n)$. A subband has its own adaptive filter with its own weight vector adapted separately by an adaptive algorithm, and has its own output $y_{\text{subband}}(n)$ which is subtracted from the subband filtered desired signal $d_{\text{subband}}(n)$ to give its error signal $e_{\text{subband}}(n)$ as shown in figure 22. These error signals are then interpolated (freq rate upconversion) using identical QMF filters. The advantages of using the subband method is lowering the signal bandwidth due to downsampling, de-correlating the signal due to the filtering and faster convergence of the adaptation algorithm. The subband stages can be increased to M stages as shown in (b) subsection.

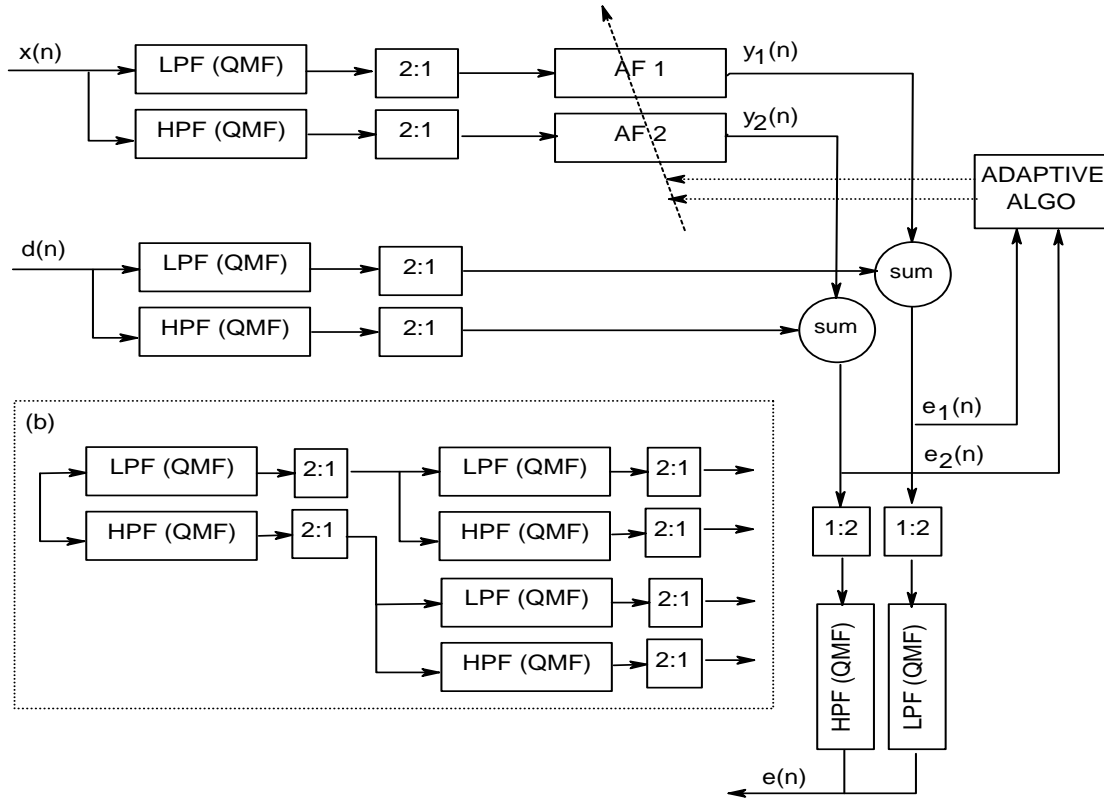


Figure 22. Subband Adaptive Filter Structure

Weight vector update for this structure (with M subbands) is done as follows,

$$\mathbf{w}_1(n+1) = [\text{beta} * \mathbf{w}_1(n)] + [[\mu/P_1(n)] * \mathbf{e}_1(n) * \mathbf{x}_1(n)]$$

...

$$\mathbf{w}_M(n+1) = [\text{beta} * \mathbf{w}_M(n)] + [[\mu/P_M(n)] * \mathbf{e}_M(n) * \mathbf{x}_M(n)]$$

beta is the leaky factor, mu is the filter gain factor, $P_x(n)$ is the subband power estimation,

$$P_x(n) = [\text{alpha} * P_x(n)] + [(1-\text{alpha}) * [x_x(n)]^2], \text{ where alpha is the "forgetting factor"}$$

- Frequency Domain Adaptive Filters** are filters whereby the input signal $x(n)$ and the desired or reference signal $d(n)$ are both transformed from the time domain into the frequency domain using a frequency transform (complex FFT or the real FHT or the real DCT) and an adaptive weight is placed between each of their frequency points (basically the output $Y(n)$ is a **circular convolution** between the weight vector $W(n)$ and the transformed input vector $X(n)$). This technique (orthogonal transforms reduce or compress the eigenvalue spread of the input signal – whitening of the signal – de-correlation) is specifically used to improve the speed of convergence of the adaptive algorithm and also to reduce the number of taps used in the adaptive filter.

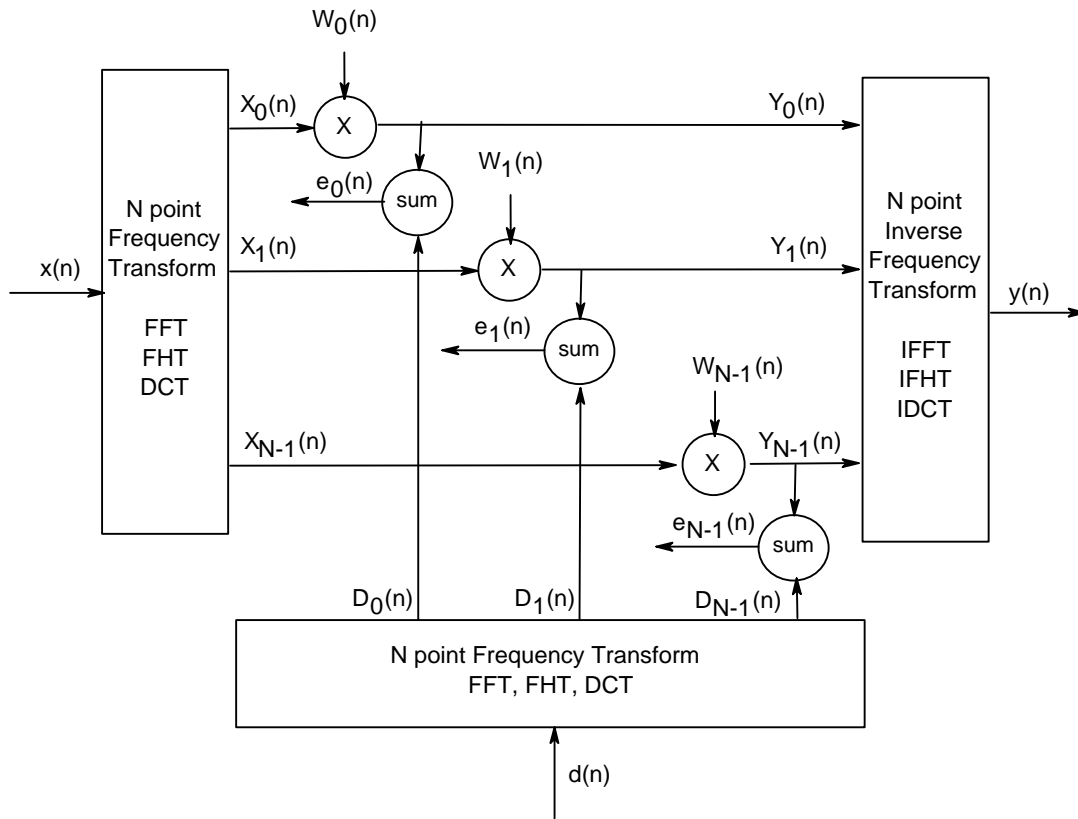


Figure 23. Frequency Domain Adaptive Filter Structure

$Y_k(n) = W_k(n) * X_k(n)$, where $k = 0, 1, \dots, N-1$, basically the elements of the output vector Y in the frequency domain (before transformation into the time domain - using an inverse of the original transform) are obtained by multiplying a single element of the weight vector W with a single element of the input vector X (convolution in time domain is multiplication in the frequency domain).

The frequency transform could be one of the real transforms like the FHT or the DCT, in that case all the operations are real (real part – a). In the case of using a complex transform like the FFT, all the vector elements of X , D , W and the output vector Y are complex and all operations are complex (real & imaginary parts - $a+ib$).

- **LMS or the least mean square adaptive algorithm** and its 2 variants are the most widely used adaptive algorithms for updating the weights in adaptive filters. This algorithm minimizes or goes towards or finds the mean square error using the method of steepest descent. Basically this algorithm changes the weight vector at each step of the adaptation process in the direction of the negative gradient of the mean square error surface. These are used in a variety of applications ranging from adaptive line enhancers to echo cancellers to active noise cancellation.

The LMS algorithm updates the weight vector using the below formula for each input sample $x(n)$, where $n = 0, 1, 2, \dots, \text{infinity}$,

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) + [2\mu e(n)x(n)] \dots \text{where, } k = 0, 1, 2, \dots, N-1 \text{ [N = total weights]}$$

μ is the step size of the adaptation ($\mu < 1$), the lower the “ μ ” the slower the convergence rate, the higher the “ μ ” the higher the chances of the adaptive system becoming unstable.

$e(n)$ or $[d(n) - x(n)]$ is the instantaneous error at time n .

$x(n)$ is the sampled signal element at time n .

Initial weight vector $\mathbf{W}(n)$ should be set to 0 at time $n=0$.

a. Normalized least mean square algorithm – the stability, convergence time and the fluctuation of the adaptation process are governed by 2 things,

- step size μ – selection of the largest μ depends on the largest eigenvalue of the input signal's autocorrelation matrix.
- input power of the signal – an automatic AGC circuit may be needed to keep the signal power envelope constant before being fed into the adaptive filter.

Both these can be achieved using the normalized LMS which uses a variable convergence factor $\mu(n)$,

$$\mu(n) = \alpha / \text{var}(n),$$

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) + [\mu(n)e(n)x(n)]$$

α is the convergence parameter, $\text{var}(n)$ is the estimate of the input signal's average power at time n ,

$$\text{var}(n) = [(1-\beta) * \text{var}(n-1)] + [\beta * [x(n)]^2], \dots \text{where, } 0 < \beta < 1$$

β being the smoothing factor, for all practical purposes, $[\alpha = \beta]$.

b. Leaky least mean square algorithm – is used when the autocorrelation matrix of the input signal has one or more zero

eigenvalues, causing some of the associated modes of the adaptive algorithm to be undriven or undamped. This can cause some of the uncoupled nodes of the adaptive algorithm to remain unconverged and thus the adaptive algorithm cannot converge to a unique solution. The solution is to add some amount of white noise into the filter input to “quiet” the coefficients. A practical method is to use coefficient leakage as follows,

$$\mathbf{w}_k(n+1) = \alpha \mathbf{w}_k(n) + 2\mu \mathbf{e}(n) \mathbf{x}(n) \dots \text{where, } \alpha \text{ is usually “0.99”}.$$

Thus, if there is no error or input signal the updated coefficients will “decay” to zero gradually.

- **RLS or the recursive least squares** – One limitation of the LMS algorithm is that it does not use all the information contained in the input data set for converging. The RLS algorithm uses the Newton’s adaptation method to find or go towards the minimum of the mean square error surface. So it computes recursively, the inverse of the mean square error function or inverse of the “input correlation matrix”, $R = E[\mathbf{x}(n)\mathbf{x}^T(n)]$, or R^{-1} . The RLS algorithm uses the Matrix Inversion Lemma technique to compute R^{-1} . The most important feature of the RLS algorithm is that it uses all the information in the input data set, extending back to the time the algorithm was initiated, thus the convergence speed is an order of magnitude faster than the LMS algorithm.

The RLS algorithm is given below for each incoming input sample $\mathbf{x}(n)$, where $n = 0, 1, \dots, \text{infinity}$,

$$y(n) = \mathbf{w}^T(n) \mathbf{x}(n),$$

$$\alpha(n) = d(n) - [\mathbf{w}^T(n) * \mathbf{x}(n)],$$

$$\mathbf{k}(n) = [\beta^{-1} * \mathbf{P}(n-1) * \mathbf{x}(n)] / [1 + \beta^{-1} * \mathbf{x}^T(n) * \mathbf{P}(n-1) * \mathbf{x}(n)],$$

$$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) + [\mathbf{k}(n) * \alpha(n)],$$

β is the weighting or forgetting factor ($0 < \beta \leq 1$), usually 0.99, is used to forget data samples in the distant past, the initial weight vector $\mathbf{W}(n)$ should be set to 0 at time $n=0$. $\mathbf{P}(n=0) = \delta^{-2}$ where δ is a small positive constant.

Notes:

1. Both the LMS and the RLS algorithms can be used to adapt the weights in any of the adaptive filter structures shown in this chapter.

FREQUENCY TRANSFORMS

Frequency transforms are widely used in all areas of signal and image processing applications,

- a. Conversion from the time domain into the frequency domain for further processing of the individual spectral components (or frequencies) of a signal.
- b. Conversion into the frequency domain to de-correlate a highly correlated signal like speech.
- c. Conversion into the frequency domain, for analyzing its individual frequency components for characterization and testing.
- d. Phase shifting certain or all spectral components within a signal.
- e. Calculating the Spectral Power Density (PSD) of the input signal.

There are 5 frequency transforms commonly used and are listed next,

- **HILBERT Transform** – this transform is mainly used for phase shifting a “real I” signal by 90^0 degrees (a phase shift of $\pm \pi/2$ is applied to all spectral components, creating the imaginary part Q). It finds its applications in areas of signal processing when an exact 90^0 phase shifted replica of the original signal is to be created at the receiver’s end. Single side band modulation schemes (e.g. 8/16 vestigial side band modulation to transmit ATSC signals – digital terrestrial broadcast in the US) transmit only the real part (I component) of the signal, so the receiver has to create its complex part (90^0 imag. Q component) for further processing on its end.

$x(n) = 0$, when n is even

$x(n) = 2 \cdot \sin^2[\pi \cdot (n/2)] / (\pi \cdot n)$, when n is odd

Usually, FIR filters are used to realize the Hilbert transform filter, thus limiting the number of samples in the impulse response.

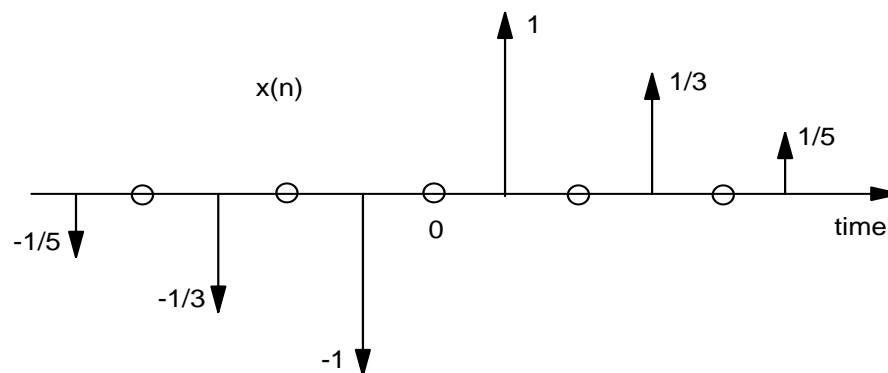


Figure 24. Impulse response of the Hilbert Transform Filter

- **FFT (Fast Fourier Transform)** – the Discrete Fourier Transform (DFT) transforms a sequence of real numbers from the time domain into a sequence of complex numbers in the frequency domain. The N point discrete fourier transform of a time sequence, $x(n)$, has the below formula,

$$X_f(k) = \text{sqrt}(1/N) * \sum_{n=0}^{N-1} [x(n) * e^{-j * [(2 * \pi) / N] * n * k}], \text{ where } X_f(k) \text{ is the DFT of } x(n)$$

The Fast Fourier Transform (FFT) algorithm makes use of two properties in the DFT equation to reduce the number of computations,

1. **The kernel of the fourier transform is periodic.**
2. **The shift rule which states that shifting a function in the time domain corresponds to multiplying a function in the frequency domain by a complex exponential factor**

The FFT uses the following equations to compute the fourier transform of a N point sequence (N has to be a power of 2) from two (N/2) point sequences,

$$X_f(k) = X_{f1}(k) + e^{-j * [(2 * \pi) / N] * k} * X_{f2}(k), \dots \text{ for } 0 \leq k \leq (N/2) - 1$$

$$X_f(k) = X_{f1}(k - N/2) - e^{-j * [(2 * \pi) / N] * [k - N/2]} * X_{f2}(k - N/2), \dots \text{ for } (N/2) \leq k \leq N - 1$$

The butterfly for a 8 point (can be extended to N points), FFT is shown below. Basically the FFT starts by “bit-reversing – not shown in figure 25” all the input data samples, and then performs a 2 point transform on the input sequence. It then takes the output from the 2 point transform and performs a 4 point transform on it. This process is repeated till N points are reached.

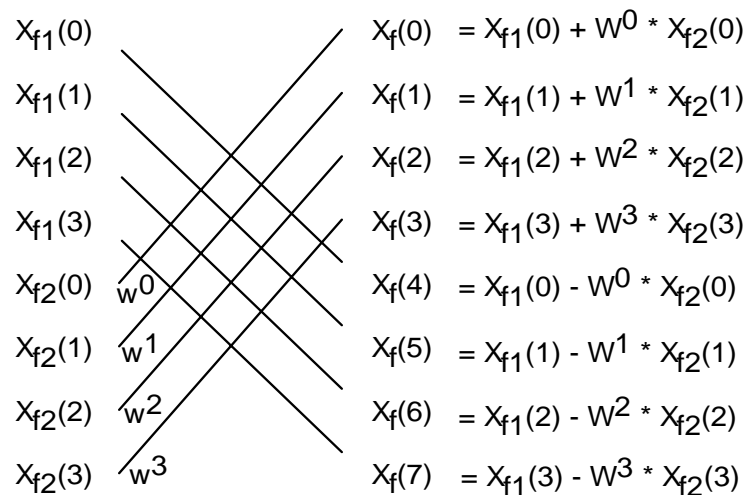


Figure 25. FFT butterfly for 8 points, where $W^k = e^{-j * [(2 * \pi) / N] * k}$

Convolution in the Fourier Space is the following,

$$W_f(k) = X_f(k) * Y_f(k)$$

Auto-correlation in the Fourier Space is the following,

$$W_f(k) = X_f(k) * X_f^*(k), \text{ where } X^* \text{ is the complex conjugate of } X.$$

Cross-correlation in the Fourier Space is the following,

$$W_f(k) = X_f(k) * Y_f^*(k), \text{ where } Y^* \text{ is the complex conjugate of } Y.$$

Hilbert Transform in the Fourier Space is the following,

$$W_f(k) = +i * X_f(k), \text{ when } k \geq 0$$

$$W_f(k) = -i * X_f(k), \text{ when } k < 0$$

- **FHT (Fast Hartley Transform)** – since the Discrete Fourier Transform transforms a sequence of real numbers into a sequence of complex numbers, half of the complex numbers in the frequency domain are redundant because the information in the negative frequencies repeat the information in the positive frequencies. If $X_f(k)$ is the fourier transform of a real sequence of real numbers, then,

$$\text{Real } [X_f(k)] = \text{Real } [X_f(-k)], \text{ and } \text{Imag } [X_f(k)] = \text{Imag } [-X_f(-k)]$$

The multiplication of complex numbers on a computer is a computationally intensive task, so unless the input sequence contains complex numbers the DFT or its equivalent FFT is not efficient. This is where the **Discrete Hartley Transform (DHT)** can be used.

The DHT transforms a sequence of real numbers in the time domain into a sequence of real numbers in the frequency domain. The Hartley transform is an algorithm that removes the redundancy in the fourier domain by repacking the numbers as follows,

$$X_h(k) = \text{Real } [X_f(k)] - \text{Imag } [X_f(k)]$$

The fourier space can be reconstructed back as follows,

$$\text{Real } [X_f(k)] = [X_h(k) + X_h(-k)]/2, \text{ and } \text{Imag } [X_f(k)] = [X_h(k) - X_h(-k)]/2$$

Thus the DHT is computationally more efficient than its DFT equivalent and its equation is as follows,

$$X_h(k) = \sqrt{1/N} * \sum_{n=0}^{N-1} [x(n) * \text{cas } [(2\pi/N)*n*k]]$$

where, $X_h(k)$ is the DHT of $x(n)$.

where, $\cos [((2\pi)/N)*n*k] = \cos [((2\pi)/N)*n*k] + \sin [((2\pi)/N)*n*k]$

The Fast Hartley Transform (FHT) for a N point sequence (N has to be a power of 2) can be derived from the DHT (who has similar properties to the FFT) using two N/2 point sequences and represented as follows,

$$X_h(k) = X_{h1}(k) + \cos [((2\pi)/N)*k] * X_{h2}(k) + \sin [((2\pi)/N)*k] * X_{h2}(-k) \dots \dots \dots 0 \leq k \leq (N/2)-1$$

$$X_h(k) = X_{h1}(k-N/2) - \cos [((2\pi)/N)*(k-N/2)] * X_{h2}(k-N/2) - \sin [((2\pi)/N)*(k-N/2)] * X_{h2}(-k+N/2) \dots \dots \dots (N/2) \leq k \leq N-1$$

The butterfly for a 8 point (can be extended to N points), FHT is shown in figure 26. Basically the FHT starts by “bit-reversing – not shown in figure 26” the input data samples, and then performs a 2 point transform on the input sequence. It then takes the output from the 2 point transform and performs a 4 point transform on it. This process is repeated till N points are reached.

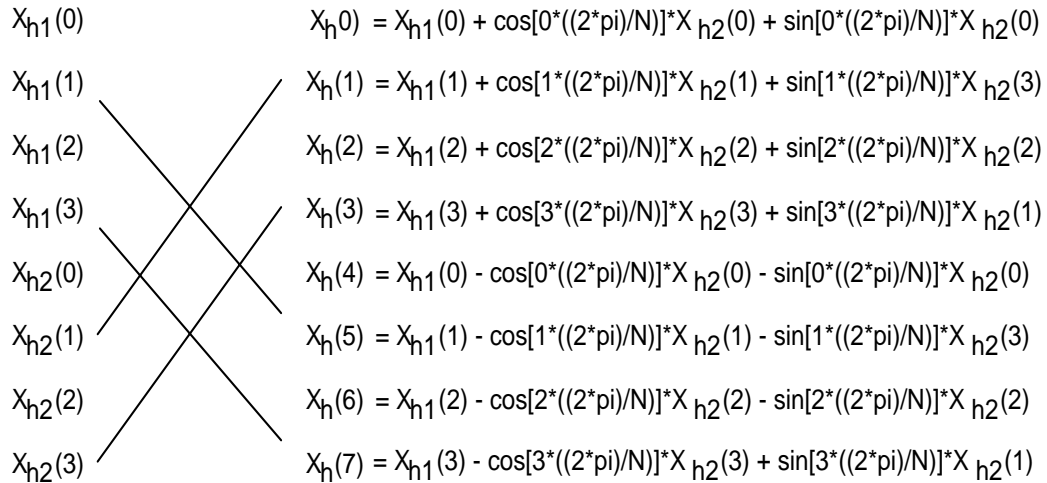


Figure 26. FHT butterfly for 8 points

Convolution in the Hartley Space is the following,

$$W_h(k) = 0.5 * [X_h(k)*[Y_h(k) + Y_h(-k)] + X_h(-k)*[Y_h(k) - Y_h(-k)]]$$

Auto-correlation in the Hartley Space is the following,

$$W_h(k) = 0.5 * [[X_h(k)]^2 * [X_h(-k)]^2]$$

Cross-correlation in the Hartley Space is the following,

$$W_h(k) = 0.5 * [X_h(k)*[Y_h(k) + Y_h(-k)] - X_h(-k)*[Y_h(k) - Y_h(-k)]]$$

Hilbert Transform in the Hartley Space is the following,

$$W_h(k) = -X_r(-k), \text{ when } k \geq 0$$

$$W_h(k) = +X_r(-k), \text{ when } k < 0$$

Notes:

1. When the FFT or the FHT is to be computed over a data set using the windowing technique (e.g., spectrum analysis - basically a overlap window of x samples is used to move the FFT of N points over the entire data set), and if the last set of data samples windowed are not a power of 2, then these samples are **zero padded to N points** over which the FFT is then applied.
2. The FFT and the FHT are separable transforms, thus the **2D FFT** or the **2D FHT** can be obtained by applying the 1D FFT (or FHT) first to all rows, and then to all columns.
3. The **Inverse FFT** (IFFT) is obtained by taking a “**conjugate**” of the FFT transformed data and passing it back into the FFT to obtain the original signal samples. The **Inverse FHT** is identical to the forward FHT (FHT \leftrightarrow IFHT)
4. The output of the FFT (or IFFT) and FHT (or IFHT) should be scaled by the **square root of N points** for practical implementations.
5. If the image is colored (RGB or YUV), then the transform needs to be applied to all color components.
6. Bit-reversal algorithm applied to FFT/FHT input samples or IFFT/IFHT input coefficients,

```

j = 1;
n = Npoints;
n1 = Npoints - 1;

for (i=1; i<=n1; i++)
{
    if (i>=j)
        goto lap1;

    InputBuffer [j-1] = InputBuffer [i-1];
    InputBuffer [i-1] = InputBuffer [j-1];

lap1:
    k = n/2;

lap3:
    if (k>=j)
        goto lap2;
    else
        j = j - k;

    k = k/2;

    goto lap3;

lap2:
    j = j+k;
}

```

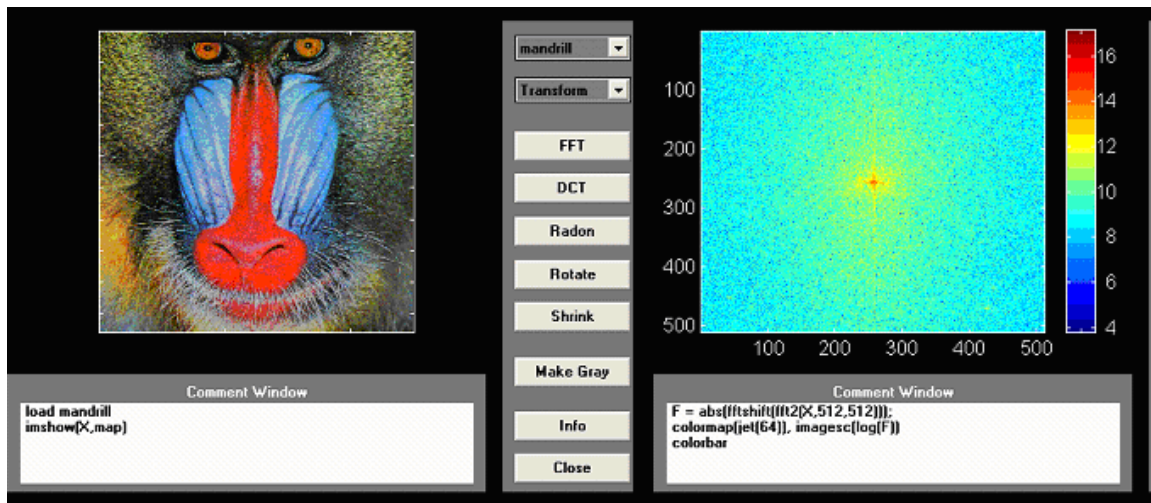


Figure 27. 2D FFT applied to the image of a “mandrill”. In this image the origin (or the spectrum) is shifted towards the center. In both the FFT/FHT, the frequency components near the center (DC component of the image) are the low frequency components and those away from the center are the high frequency components. Thus directional filters based on this information can be built for “lossy image coding” by discarding some of the frequency coefficients in the transformed image.

- **DCT (Discrete Cosine Transform)** – this transform is widely used in speech and image coding including the popular JPEG, MPEG and DOLBY coding techniques. Its advantages are given below,
 1. DCT is preferred for compression because it concentrates the largest percentage of signal energy in a small percentage of coefficients, especially in signals which have high spatial correlation.
 2. DCT is a real transform, no complex number calculations.
 3. Useful in compression because the coding efficiency is high, thus good compression ratios can be obtained with minimum picture distortion.
 4. Ortho-normal transform, thus the Inverse DCT can be obtained by passing the DCT transform over the transformed image to get back the original image.

The equations for the **1D DCT** and the **2D DCT** are given below,

$$X_d(k) = \sqrt{2/N} * \beta(k) * \sum_{n=0}^{N-1} [x(n) * \cos [((2n+1)*k*\pi)/2N]] \dots \text{where, } k = 0, 1, \dots, N-1$$

where, $X_d(k)$ is the 1D DCT of $x(n)$, $\beta(0) = 1/\sqrt{2}$, and $\beta(k) = 1$, for $k > 0$

$$X_d(k,l) = [2/N * \beta(k) * \beta(l)] * \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} [x(m,n) * \cos [((2m+1)*k*\pi)/2N] * \cos [((2n+1)*l*\pi)/2N]]$$

where, $X_d(k,l)$ is the 2D DCT of $x(m,n)$, $\beta(0) = 1/\sqrt{2}$, and $\beta(k) \& \beta(l) = 1$, for $k, l > 0$
 where, $x(m,n)$ image always has $N \times N$ dimensions, basically $k=0, 1, \dots, N-1$ and $l=0, 1, 2, \dots, N-1$

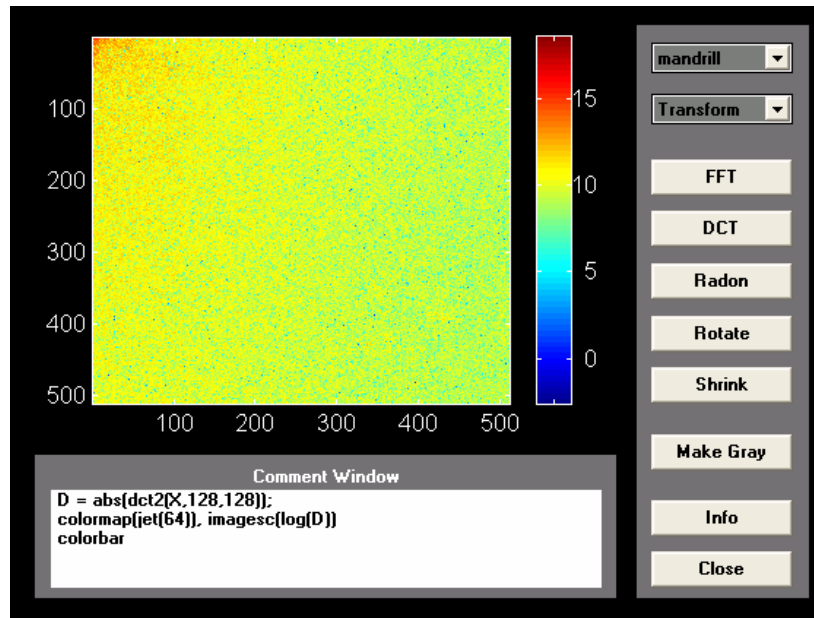
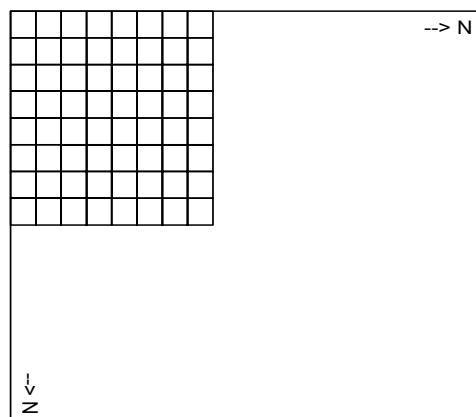


Figure 28. 2D DCT [128x128 points] applied to the image of a “mandrill”. In this image you can clearly see the DC component & the low frequency components towards the “top left” of the image. In Figure 29, you can see the effect of keeping just the DC component or the DC component along with some low frequency components and then applying a inverse 2D DCT.

In Speech (or a one dimensional signal) the 1D DCT is applied over the entire length of the signal using a windowing technique (1D DCT window is moved over the signal).

In (m,n) images the 2D DCT is applied using a moving (axb) block/window over the (m,n) image as shown below. Zero-padding is used to extend an imperfect image to NxN dimensions for the (axb) window to fit correctly. This method is used to keep the DCT/IDCT computations simple for digital image processing hardware and to localize image coding to 8x8 blocks.



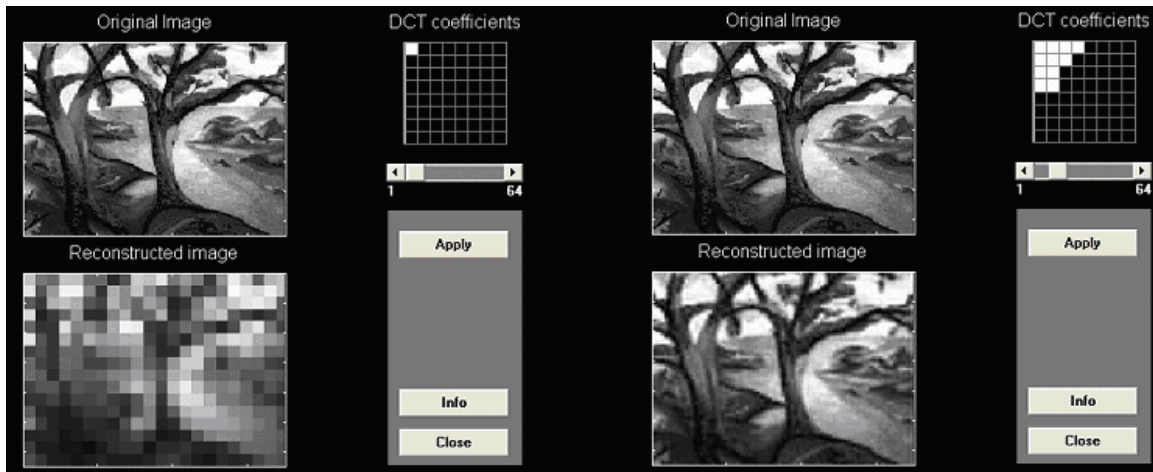
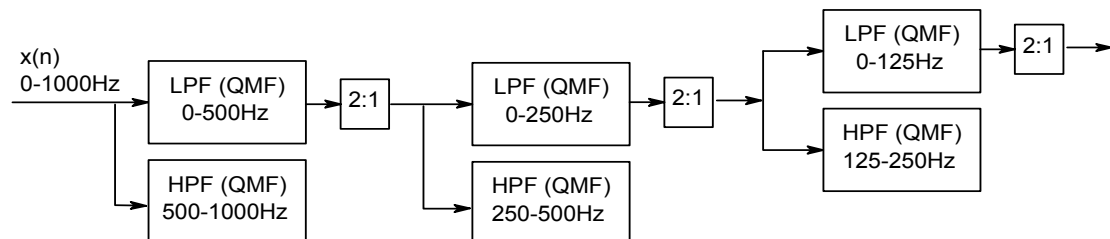


Figure 29. Inverse 2D DCT [8x8 points] applied to the image of a tree after discarding some of the high frequency coefficients. As you can see the image looks low pass filtered.

- **Wavelet Transform** – one of the disadvantages of the DCT is that it is block based transform (divides a image arbitrarily into $a \times b$ blocks over which the transform is applied), so the resulting image could appear blocky after going through a compression → decompression operation using a built in DCT/IDCT stage. The other disadvantage of the Cosine, Fourier and the Hartley transform is that they are both purely frequency transforms (they tell a user what the frequency components are in a signal or image).

The wavelet transform is gaining popularity in all types of coding and compression applications because it can provide a **time-frequency representation of a signal OR time localization of the spectral components**. This can be very useful if information about specific frequency components that occur in specific instants of time is needed (e.g. EEG, where an event happening in time is of specific importance – response of brain to a specific stimulus). A wavelet transform is obtained by passing the time domain signal through various highpass and lowpass quadrature mirror filters, and each time a portion of the signal corresponding to certain frequencies is removed (signal is decomposed). Reversal of this **decomposition** process is called **reconstruction**.



Above we have decomposed a 0-1000Hz signal into 4 frequency bands using QMF's (0-125Hz, 125-250Hz, 250-500Hz & 500-1000Hz). This process is continued till a certain level of decomposition is reached. If these frequency bands are plotted on a 3D graph, we will have time, frequency and amplitude on each of the 3 axes. This will clearly show **which frequency bands exist at certain time intervals**.

Basically Quadrature Mirror Filters with impulse response $h(n)$ are used for convolution with the signal $x(n)$ to decompose or reconstruct that signal.

$$y(n) = \sum_{k=0}^{2N-1} [h(k) * x(2n-k)]$$

A total of 4 filters (high pass and low pass filters are designed so that the aliasing introduced by the decimations are exactly canceled in reconstructions) are needed for the decomposition and reconstruction and are calculated from a “**scaling filter W**” which is classified as follows,

1. FIR
2. length $2N$
3. sum of coeffs is 1
4. normalized to $1/\sqrt{2}$
5. low pass filter

$$Lo_R = W/\text{norm}(W), Lo_D = \text{wrev}(Lo_R), Hi_R = \text{qmf}(Lo_R), Hi_D = \text{wrev}(Hi_R)$$

where, “norm” is $1/\sqrt{2}$, “qmf” is such that Hi_R & Lo_R are quadrature mirror filters, and “wrev” flips the filter coefficients left to right.

$$Hi_R(k) = -1 * k * Lo_R(2N+1-k) \dots\dots\text{where, } k=0,1,2,\dots,2N-1$$

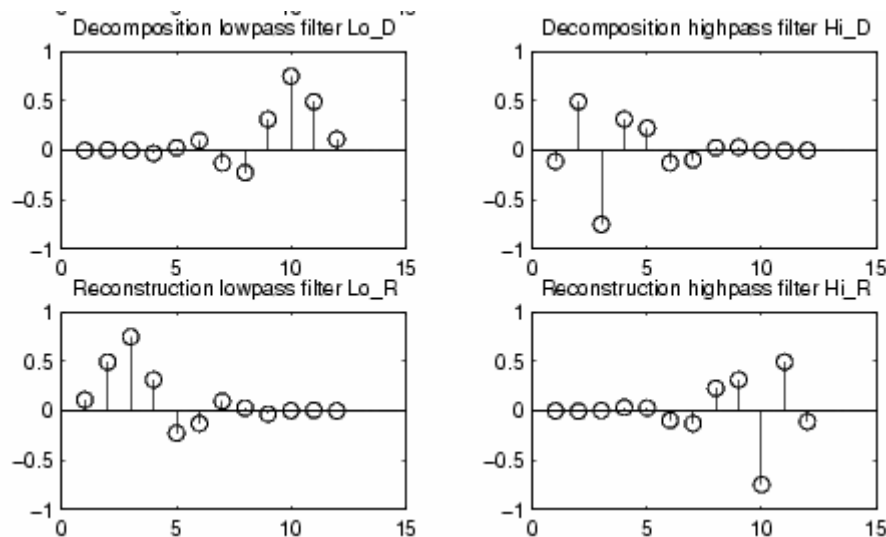
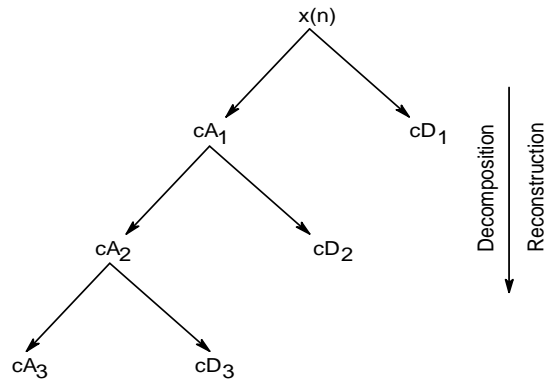


Figure 30. Four typical filters obtained for decomposition & reconstruction from a typical scaling filter W.

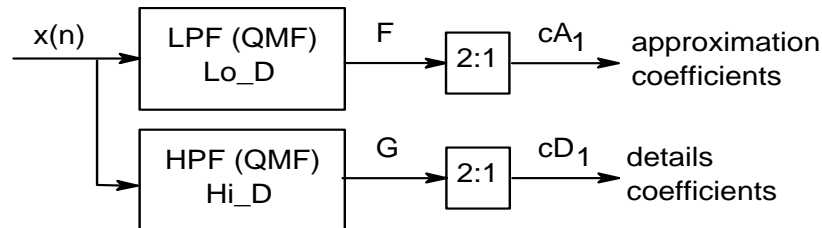
Lets now look at the algorithms used for decomposing or reconstructing a N point input sequence $x(n)$.

Given a **signal $x(n)$ of length N** , the **DWT** consists of **$\log_2 N$ stages or levels**. Starting from $x(n)$, the first step produces two sets of coefficients, **approximation coefficients cA_1** , and **details coefficients cD_1** . These vectors are obtained by convolving $x(n)$ with the low-pass filter **Lo_D for approximation**, and with the high-pass filter **Hi_D for details**, followed by a **dyadic decimation** (down-sampling by 2 and keeping only the even indexed elements) stage.

1. STAGES or LEVELS (when $j = 3$) in the One-Dimensional DWT



2. FIRST STAGE in One-Dimensional Discrete Wavelet Transform



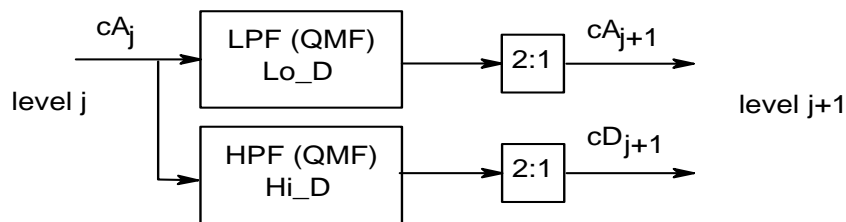
The length of each QMF is equal to $2N$.

The length of filtered (or convolved) signals F & $G = \text{length}[x(n)] + 2N - 1$.

The length of coefficients cA_1 & $cD_1 = \text{floor}[(n-1)/2] + N$

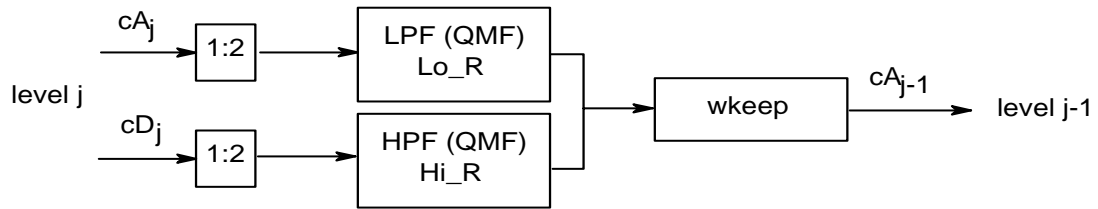
The decimation (or down conversion) by a factor of 2 keeps only the even indexed elements (a.k.a, dyadic decimation).

3. NEXT STAGE (j) in DECOMPOSITION (One-Dimensional Case)



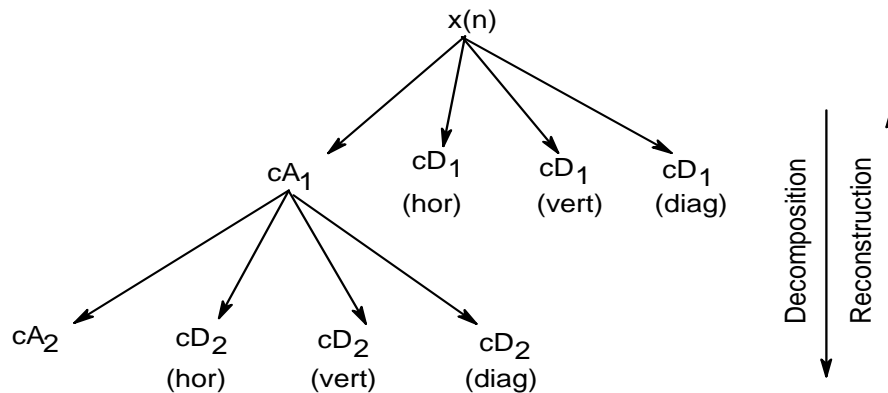
Now only the approximation coefficients CA_j ($j=1$) are fed into the following decomposition stages for 1D DWT as shown.

4. NEXT STAGE (j) in RECONSTRUCTION (One-Dimensional Case)



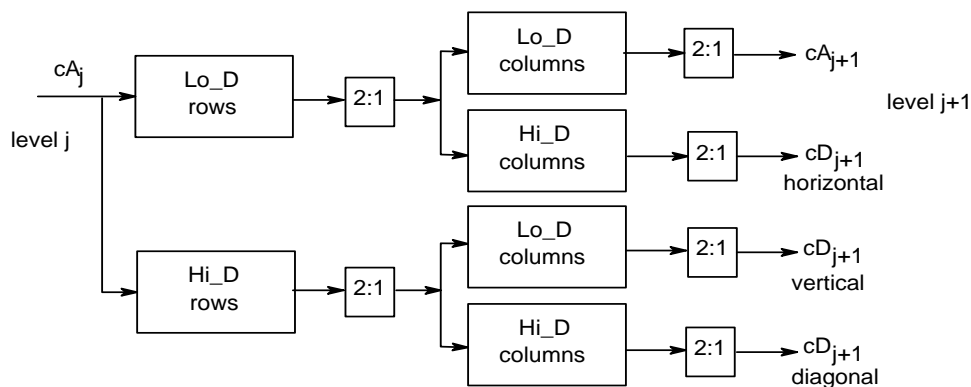
Upsample by inserting zeros at odd indexed elements, convolve with the reconstruction filters and (wkeep) keep the central samples of the convolution.

5. STAGES or LEVELS (when j=2) in the Two-Dimensional DWT



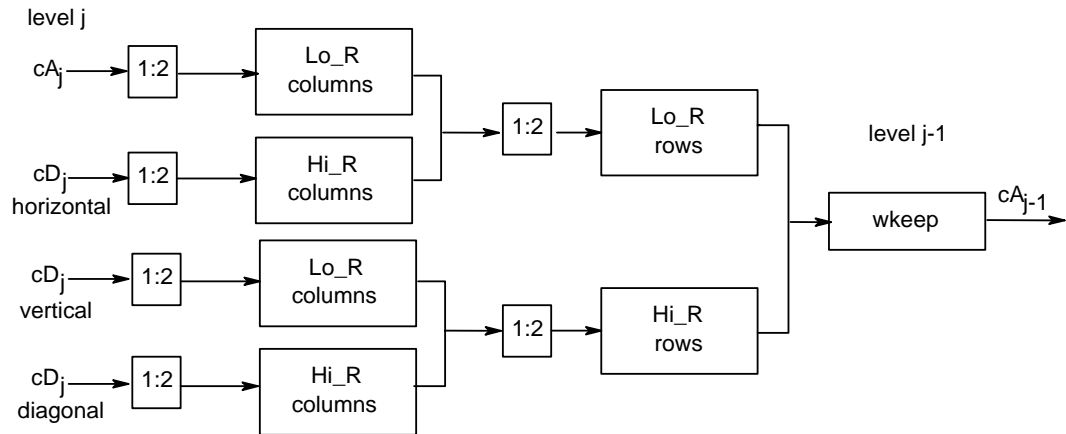
For bi-orthogonal wavelets, the same algorithms hold but the decomposition filters and the reconstruction filters are obtained from two distinct scaling filters (one for rows and the other for columns). In the 2D case, the filters for decomposition and reconstruction are of different odd lengths. By zero-padding, the 4 filters can be extended in such a way that they will have the same even length.

6. Any STAGE (j) in DECOMPOSITION (Two-Dimensional Case)



The 2D DWT leads to a decomposition of approximation coefficients at (level j) in 4 components, the approximation at (level j+1) and details in three other orientations (horizontal, vertical & diagonal). The 2:1 down sampling stage decimates and keeps only the "even indexed" rows & columns.

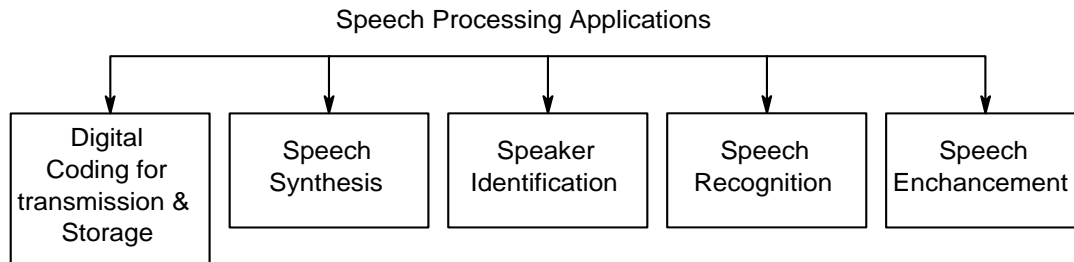
7. NEXT STAGE (j) in RECONSTRUCTION (Two-Dimensional Case)



The 1:2 up-sampling stage up samples rows and columns by inserting zeros into “odd indexed” columns and rows. The function “wkeep” keeps only the central samples from the convolution.

APPLICATIONS in SPEECH PROCESSING

The applications under speech processing can be broadly classified as follows,



- Digital Coding of speech finds applications in areas like digital transmission of audio over any media like terrestrial, cable or satellite where digitized speech is first compressed, coded, packetized and modulated for transmission over that media. Coding is also employed when digitized speech is stored on media like a hard disk or a memory stick for future use.
- Speech synthesis systems find use in automated computer based voice response systems.
- Speaker identification is a technique to identify a specific person whose acoustic signature for a certain template is digitized and compared (correlated) against a LUT of templates of numerous speakers.
- Speech recognition systems are mostly the speech to text conversion systems and have 2 parts, speaker independent and speaker dependant systems. Speaker independent systems take a digitized acoustic signature from a unknown speaker, split it into various parts (words or characters), compare (correlate) it with stored acoustic signatures for those words or characters and output text corresponding to that signature. Speaker dependant systems involve an additional step in which the individual speaker's voice is first trained for certain acoustic signatures (words or characters) and then the speech to text system is custom built for that speaker.
- Speech enhancement involves the filtering of speech with standard or adaptive digital filters and is very useful for removal of noise, telephone line echo cancellation and acoustic echo cancellation found in voice conferencing systems.

Speech Coding Systems

We will discuss 2 speech coding techniques,

- **ADPCM (adaptive digital pulse code modulation)** – since speech is highly correlated (each signal sample does not change rapidly from one to the other) the difference between two consecutive samples has a lower variance than the actual signal sample, so the difference can be quantized with lesser levels (or bits) than the actual signal.

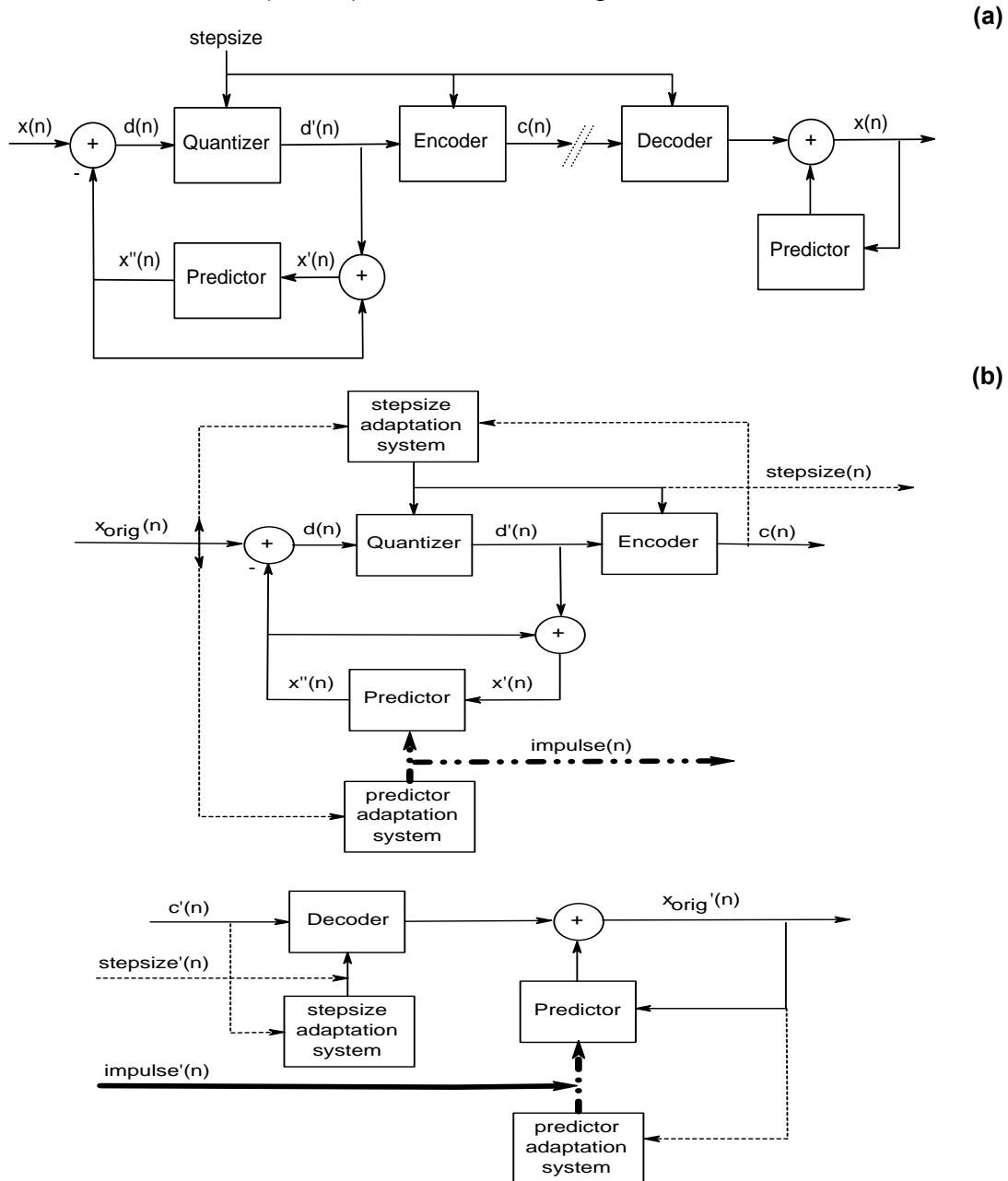


Figure 31. Differential quantization (a) and ADPCM (b)

Now, in the above differential quantization scheme when both the quantizer stepsize and the predictor filter taps are changed adaptively, the resulting scheme is called ADPCM. This becomes important to accommodate different speakers, different speech material and variations in the signal levels.

The step size can be controlled adaptively as shown to be proportional to the variance of the input to the quantizer in the case of a **feedforward** system or proportional to the variance of the difference signal in the case of a **feedbackward** system.

The predictor coefficients are time dependant, and it is common to assume that the properties of the speech signal remain fixed over a short interval of time. Thus the predictor coefficients are adapted to minimize the average squared prediction error over a short time interval (N taps).

$$x''(n) = \sum_{k=0}^{N-1} \text{impulse}_k(n) * x'(n-k)$$

To find the predictor coefficients **[impulse_k(n)]** a method called **linear predictive analysis** of speech is used which suggests that a speech sample can be approximated by a linear combination of past speech samples. Basically by minimizing the sum of the squared differences (over a finite interval) between actual speech samples and the linearly predicted ones, a unique set of predictor coefficients can be obtained.

- **LPC (linear predictive coding of speech)** – the general set of linear predictive analysis techniques to determine the predictor coefficients is called LPC of speech. LPC techniques are mainly used for system estimation and system identification in that once the predictor coefficients have been obtained, the system can be modeled as an all pole linear system. Linear prediction methods for analyzing the speech waveform to obtain the predictor coefficients are classified into 3 main categories,

1. **Covariance method**

2. **Autocorrelation method**

3. **Lattice method** – all most all digital signal processor implementations of LPC, today use the lattice method to adaptively determine the predictor coefficients or weights.

The speech model shown in figure 32, has the below response,

$$H(z) = S(z) / U(z) = G / [1 - \sum_{k=0}^{N-1} (a_k z^{-k})]$$

This model is excited by an impulse train for voiced speech OR a random noise sequence for unvoiced speech. Thus, the parameters of this model are, unvoiced/voice classification, pitch period for voiced speech, gain parameter [G], and the filter coefficients $[a_k]$. All these parameters vary slowly with time.

Thus, the speech samples $s(n)$ are related to the excitation $u(n)$ by the following equation in the below model,

$$s(n) = \sum_{k=0}^{N-1} [a_k * s(n-k)] + G * u(n)$$

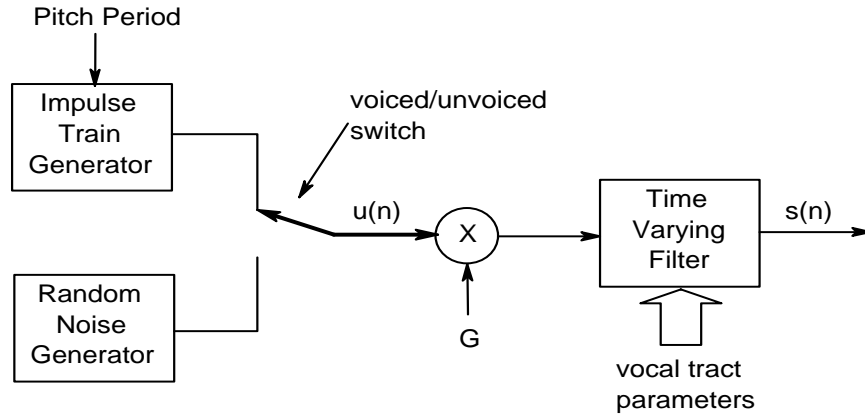


Figure 32. Speech production model for the LPC technique

A linear predictor with prediction coefficients $[impulse_k]$ is depicted as follows,

$$s'(n) = \sum_{k=0}^{N-1} [impulse_k * s(n-k)] \text{ OR, } N(z) = \sum_{k=0}^{N-1} [impulse_k * z^{-k}]$$

the prediction error $e(n)$ is defined as,

$$e(n) = s(n) - s'(n) = s(n) - \sum_{k=0}^{N-1} [impulse_k * s(n-k)]$$

the transfer function $E(z)$ of this error sequence is,

$$E(z) = 1 - \sum_{k=0}^{N-1} [impulse_k * z^{-k}]$$

Thus if the speech signal follows the Speech Production Model of figure 32, and if $impulse_k = a_k$ then, error $e(n) = G * u(n)$ and the prediction error filter, $E(z)$, will be an inverse filter for the speech model's transfer function,

$$H(z) = G / E(z)$$

The basic problem which the linear predictive analysis model has to solve is to obtain or determine sets of predictor coefficients $[impulse_k]$ directly

from the speech signal by obtaining good estimates of the spectral properties of short speech segments. The basic approach is to find a set of predictor coefficients that will minimize the mean-squared error over short speech segments. Most commonly used method for this is the **lattice adaptive predictor** as explained in chapter 4 (Adaptive Filters)

The most popular algorithm for finding the **pitch period** and making **voiced/unvoiced decisions** is the **SIFT** algorithm as shown in figure 33.

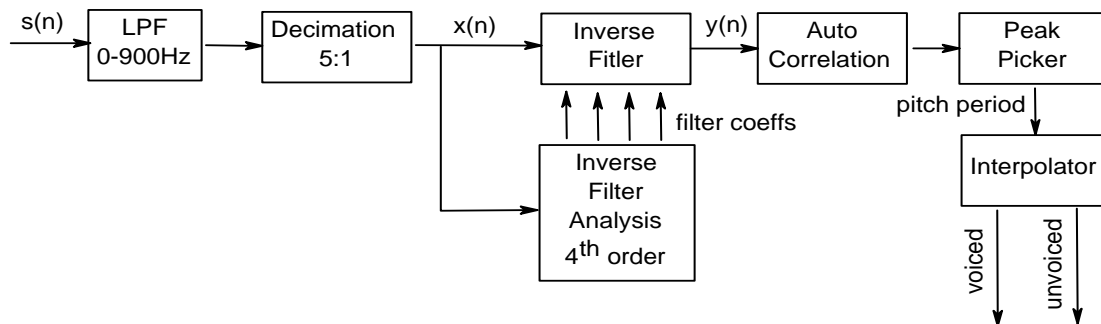


Figure 33. Speech production model for the LPC technique

Basically the input speech signal $s(n)$ is low pass filtered around 900Hz, then the sampling rate is reduced by a factor of 5. The decimated output is then analyzed (range 0-1 KHz) using the autocorrelation method using a 4th order filter. The signal $x(n)$ is then inverse filtered to give $y(n)$ which has a flat spectrum. The **short time autocorrelation function** is computed and the **largest peak** is the **pitch period**. Additional resolution around the pitch period is obtained by interpolating the signal around the largest peak. An **unvoiced** decision can be made when the level of the autocorrelation function falls below a certain threshold.

The **predictor coefficients** along with the **pitch period & voiced/unvoiced** classification can be used in the speech production model shown in figure 33.

Both ADPCM and LPC are time domain speech coding systems used in low bit rate (< 100kbts/sec) speech encoding applications. Most high bit rate speech or multi-channel audio coding or compression systems are “frequency transform based” and an entire chapter has been devoted to the most popular of them, namely MPEG & DOLBY.

Automated Digital Voice Response Systems

In figure 34, we have shown a digital voice response system in which the vocabulary (spoken words) elements were coded using ADPCM.

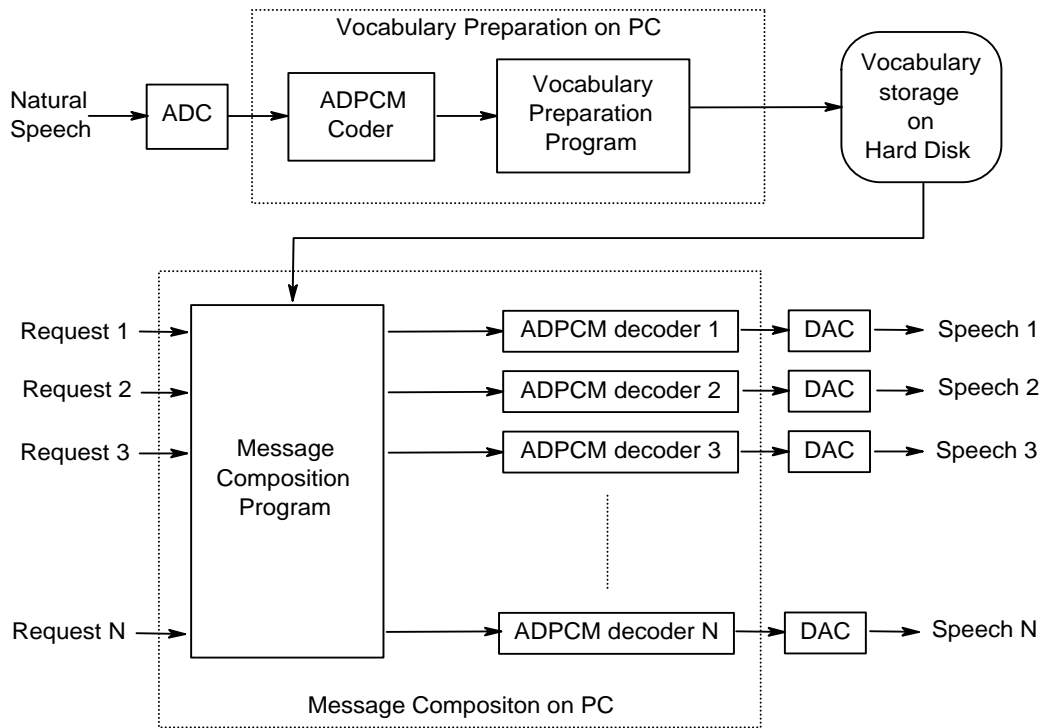


Figure 34. Automated Voice Response System for N users

The beginning and end points of each word is automatically located by an algorithm which computes the “short time energy” of the ADPCM code words. The entire needed vocabulary is stored on a hard disk (random access media) for fast retrieval by the message composition program. The message composition program can take multiple message requests from users typically over a phone line and index the vocabulary stored on the hard disk. This vocabulary can then be cached or buffered in the computer RAM (for supporting multiple user requests) and then DMA’ed to separate ADPCM decoders. In the above figure 34, we have shown the digital voice response system (or information retrieval system) supporting N users.

Speaker Identification Systems

Both speaker identification systems and speaker verification systems (determining a speaker’s identity) employ some form of pattern recognition technique which matches (correlates) extracted features from the digitized speech signal of the speaker with stored reference patterns of N speakers. Basically using digital speech processing techniques a pattern vector x is extracted which preserves the features of the speech signal relevant to the speaker and this pattern is compared with previously prepared (and stored) reference patterns of N speakers and distance measure dx_i is calculated for the i_{th} speaker. Decisions are then made based on these distance measures for each speaker.

If we denote the probability distribution for the pattern vector x for the i^{th} speaker as $p_i(x)$, then a **speaker verification system** does the following,

verify speaker i if $p_i(x) > e_i * p_{av}(x)$
reject speaker i if $p_i(x) < e_i * p_{av}(x)$

where e_i is a constant for the i^{th} speaker, which determines the probability of error for the i^{th} speaker, and $p_{av}(x)$ is the average probability distribution for the extracted pattern x for all speakers.

And a **speaker identification system** chooses a speaker with the minimum probability of error,

choose speaker i , such that, $p_i(x) > p_j(x)$, where $j = 1, 2, 3, \dots, N$ speakers

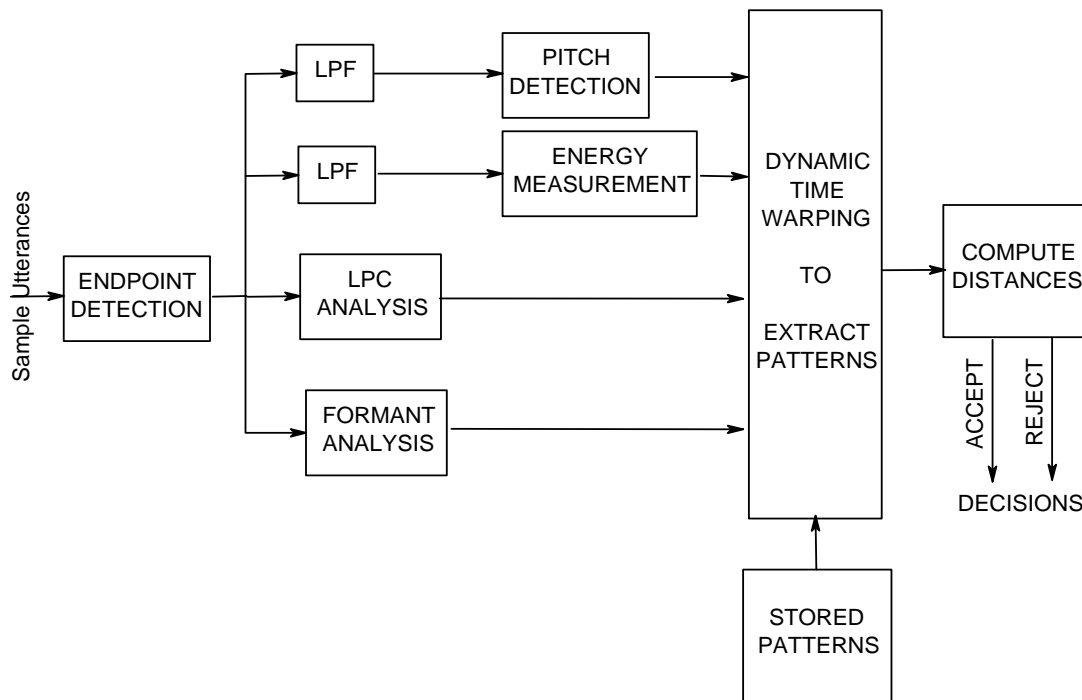


Figure 35. Speaker Identification OR Speaker Verification

Speech Recognition Systems

- A. Most **speaker independent speech recognition systems** have a very limited vocabulary and usually work well when the digits or words spoken are unconnected (isolated). If at all a speech recognition system is to be developed for connected digits or words, they are limited to 2-3 connected digits or words. These systems are usually used for automated banking (banks, credit cards, etc.) where the vocabulary used is extremely limited.

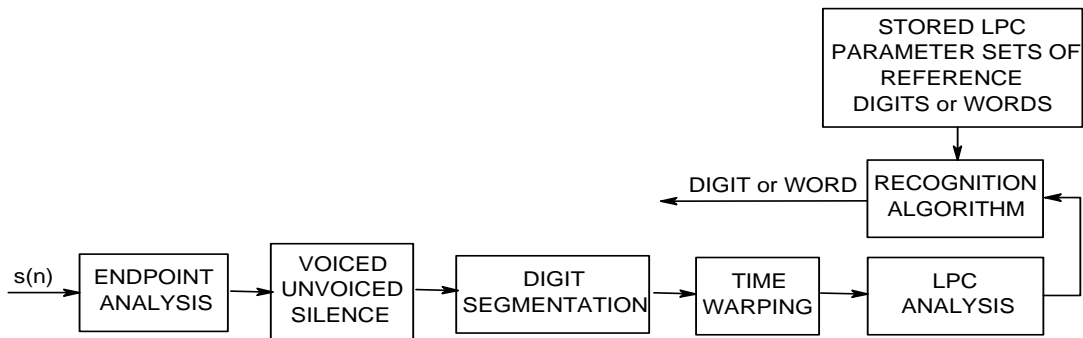


Figure 36. Continuous 2-3 Digit Recognition System

The recorded digit or word string is first subjected to endpoint analysis to determine where in the recording interval the speech utterance occurs.

Endpoint algorithm: detects the beginning and ending of a speech utterance using a time domain analysis technique based on the “average zero-crossing rate across N samples” and the “average magnitude across N samples”.

1. **Average zero crossing** is computed using the number of times the **SIGN** between 2 speech samples change during N samples and determining a average zero crossing threshold.
2. **Average magnitude** is usually computed by moving a window of N samples over the speech samples (summation between window coefficients and speech samples over N samples) and determining a average magnitude threshold.

After endpoint analysis the speech signal is subjected to the following measurements at a rate of 100 times per second.

- 1) **Zero crossing rate**
- 2) **Log energy**
- 3) **LPC coefficients**
- 4) **LPG log error**
- 5) **First auto-correlation coefficient**

These parameters are then processed using a pattern recognition technique which classifies each 10msec interval as silence, unvoiced or voiced. This is then used to segment or disconnect each digit or word. Knowledge of how many digits/words in the input string is necessary for correct segmentation. For each segmented digit/word the voiced region is analyzed by the LPC technique and then a recognition algorithm takes the LPC parameters sets of the voiced regions for the incoming digits/words and compares them to the stored LPC parameters of the reference digits/words.

One important in between step is to **time-warp** the segmented individual digit/word to the time interval of the stored reference digits/words. This enhances the pattern recognition algorithm’s ability to match digits/words.

- B. Most **speaker dependant speech recognition systems** (e.g. speech to text) are used in word processing packages which specifically train the package running on a computer or handheld device with a small subset of the spoken language used by the speaker operating that device.

The calculated distance measure is an idea as to how much each incoming word pattern differs from the stored word pattern.

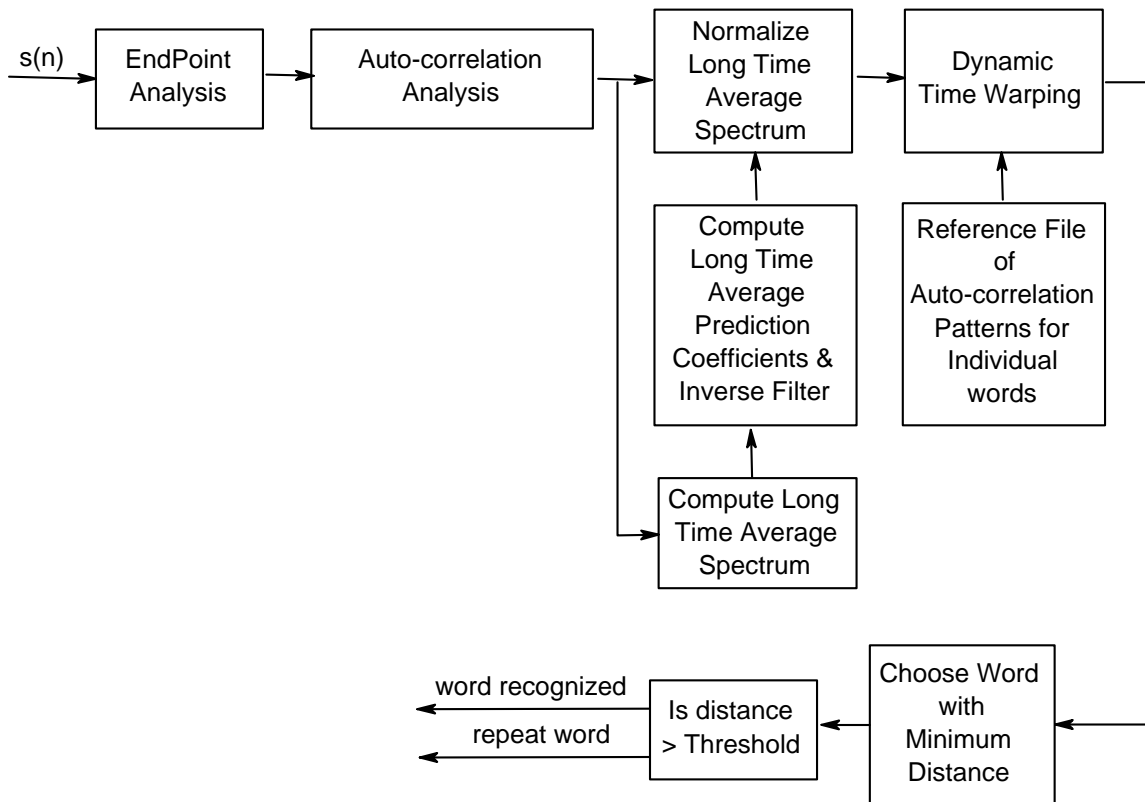


Figure 37. Speech Recognition System typical of those used in Word Processing Packages supporting large vocabularies.

Speech Enhancement Systems

Speech enhancement systems are basically filtering techniques which remove unwanted signal sources from the wanted speech signal. They can be achieved using,

1. **Standard Filters** – these include FIR or IIR sections which perform filtering tasks like LPF, HPF, notch and band pass operations and whose weights (or coefficients) are fixed.

2. **Adaptive Filters** – these include either transversal (FIR/IIR) sections or lattice sections as discussed before which perform filtering operations like removal of inter-symbol interference (equalization) in digital telephones, line echo cancellation in phones and acoustic echo cancellation in conference rooms.

We will discuss the following filtering schemes using adaptive filters,

- **Equalization** – transmission over voice bandwidth channels used by telephones and modems, time dispersion results in extending the time interval used to represent each modulated symbol. This causes the overlay of transmitted symbols at the receiver (a.k.a. inter-symbol interference). An adaptive equalizer is designed to remove this inter-symbol interference caused by time-dispersion, using adaptive filters and algorithms.

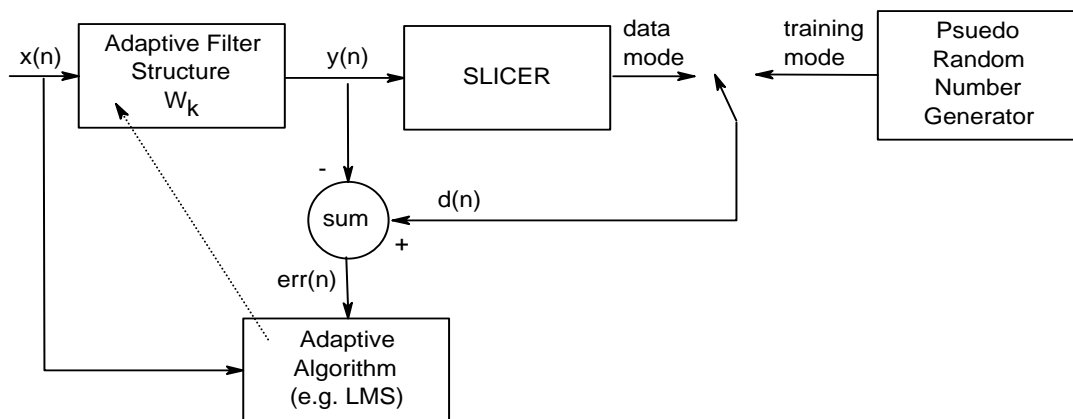


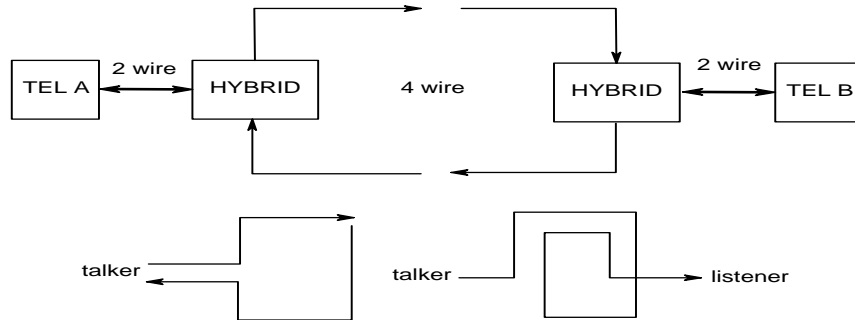
Figure 38. Typical Equalizer for Voice Channels

$x(n)$ is the received signal, $d(n)$ is the detected data signal (in data mode) or a pseudo random number (in training mode), $y(n)$ is the equalized signal and $err(n)$ is the residual inter-symbol interference. Basically bursts of white noise are transmitted over the channel in the beginning. The adaptive algorithm adapts the weights (W_k) with $d(n)$ in **training mode** to this white noise $x(n)$. Now the adaptive filter forms an **inverse** of the channel response. The adaptation is stopped and the actual signal (telephone or modem) is sent over the channel and $d(n)$ is now connected in **data mode**.

- **Line Echo Canceller** – this method is used to control or cancel echoes arising in a typical telephone network. This along with a **double talk detector** is today the most widely used method for full duplex transmission over a telephone network (telephone or data modem).

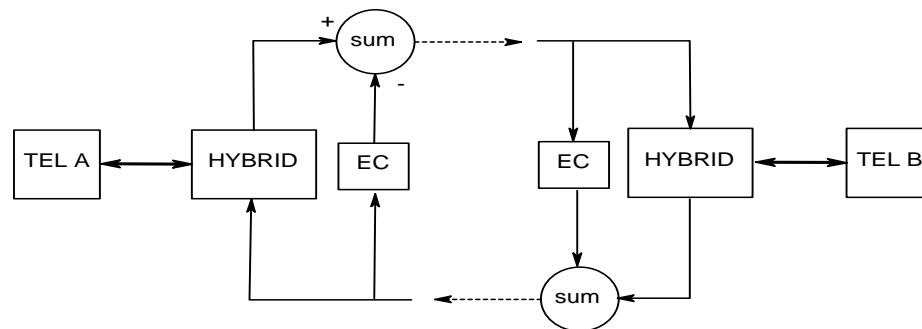
The generation of echoes is mostly caused due to **impedance mismatches** in the hybrid, which sits between and separates the 2 wire

telephone local loop from the 4 wire carrier facility. Ideally a hybrid couples all the energy on the incoming branch of the 4 wire carrier circuit onto the 2 wire local loop. Thus, none of the incoming 4 wire signal is transmitted on the outgoing branch of the 4 wire circuit. But, since the hybrid has fixed impedance a component of the received signal leaks through and is coupled onto the outgoing branch. This is the echo returned to the source.



The 2 different kinds of echoes are,

1. Talker hearing a delayed version of his/her speech.
2. Listener hearing a delayed version of the talker's speech.



To overcome these echo problems, an adaptive echo canceller is installed at both ends of the telephone network in the 4 wire network circuit path as shown above.

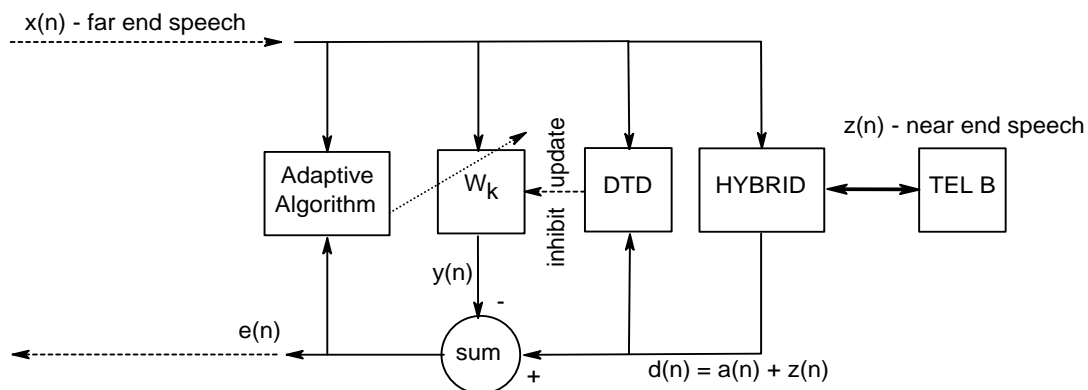


Figure 39. Typical Telephone Line Echo Canceller

The far end signal $x(n)$, passing through the echo path results in the far end echo signal $a(n)$, $d(n)$ is the result of the **near end speech $z(n)$ + far end echo signal $a(n)$** . A replica $y(n)$ of the far end echo signal is generated by sending the far end signal $x(n)$ to the adaptive filter whose weights (W_k) are updated using a normalized leaky LMS adaptive algorithm. Thus the adaptive filter learns and adapts to the response of the hybrid making $y(n) \sim a(n)$ and error signal $e(n) \sim z(n) -$ IN THEORY.

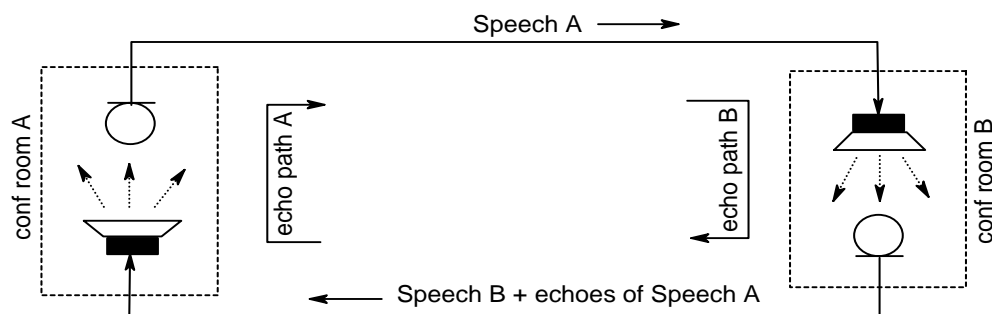
The adaptive echo canceller above estimates the impulse response of the echo path using its input $d(n)$ and output signal $y(n)$, by minimizing the error signal $e(n)$. So now during **double talking** (talker at far end & talker at near end talking at the same time) the input signal $d(n)$ will contain not only the echo signal $a(n)$, but also the near end speech $z(n)$. So the AF will take the response of not just $a(n)$, but $a(n) + z(n)$ during double talk. A double talk detector is used to detect double talk (by calculating the energy of the 2 paths into it) and **freeze** the **adaptation of the AF weights** during the presence of near end signal $z(n)$.

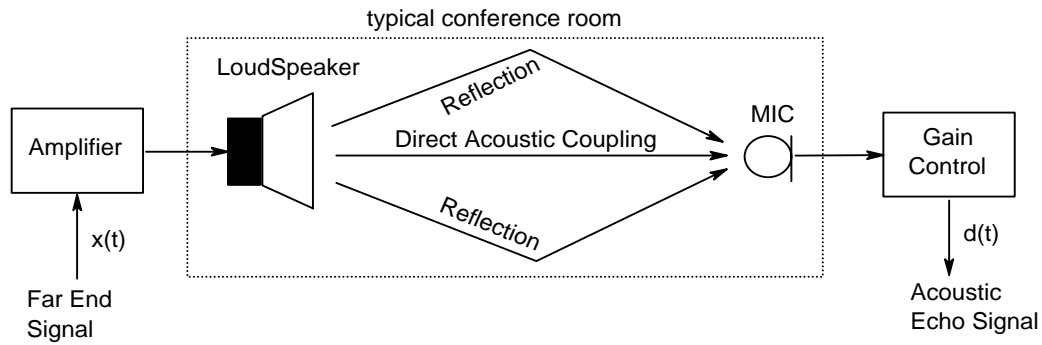
- **Acoustic Echo Canceller** – this technique is used in speaker phone and teleconferencing applications where acoustic echoes are dominant. In this environment there are 2 main components that can cause echoes,

1. The acoustic energy coupling between loudspeakers and microphones
2. Multi-path sound reflections of far end speech signals

An acoustic echo has several different characteristics than line echoes,

1. The acoustic echo can be long (~400 ms), thus the number of filter taps can be large (~5000) depending on F_s and adaptive method.
2. The acoustic echo path is non-stationary (e.g. motion of people in a conference room), thus faster convergence algorithms are needed.
3. Since the input is a speech signal (highly correlated), to improve the convergence rate we have to employ methods to de-correlate speech before the AF technique.
4. Near end speech detection is difficult for a DTD usage.





As you can from the model of a typical conference room the sound from the loudspeaker couples with the microphone via 3 paths. Also in the 2 conference room model (connected by telephone wires) or a speaker phone internal loop the loudspeaker in conf room A will be coupled not just with Speech B, but with **Speech B + echoes of Speech A**. Below we have shown a acoustic echo canceller in teleconferencing applications.

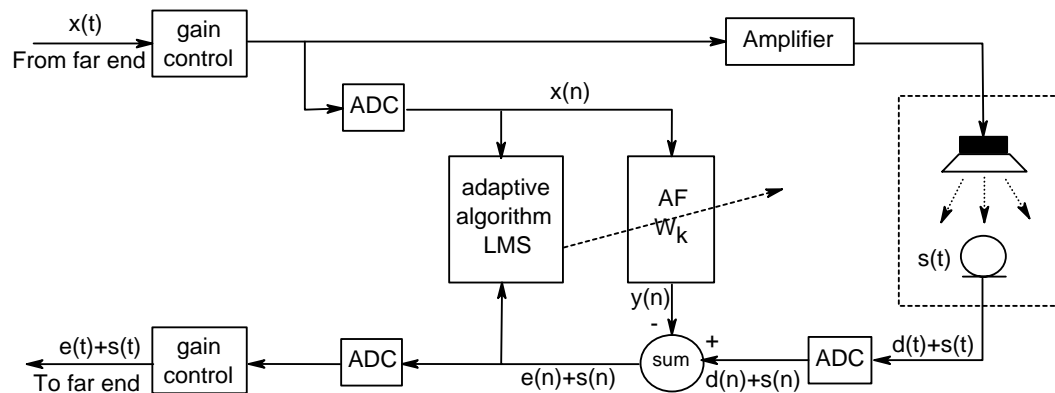
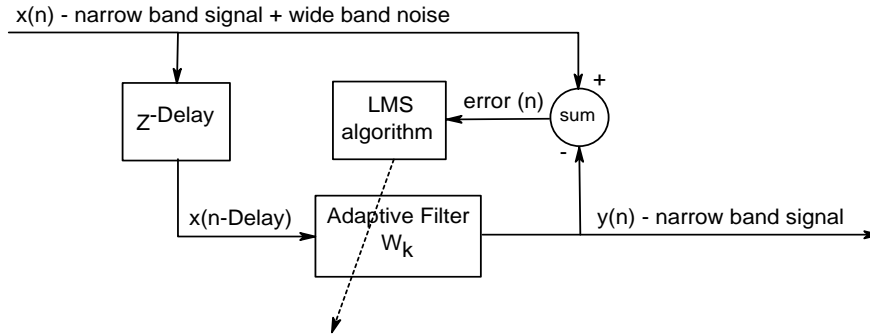


Figure 40. Typical Acoustic Echo Canceller

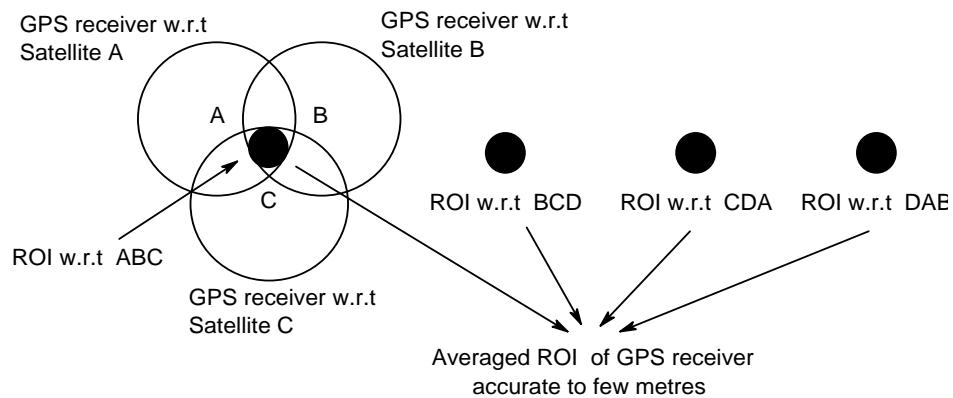
As shown in figure 40, $d(t)$ is the echo signal, basically echo of far end speech microphone $x(t)$, which gets coupled into the near end microphone speech $s(t)$ and sent to the far end. The acoustic echo canceller using a transversal, lattice or frequency domain adaptive filtering technique reproduces a replica of the signal $\sim d(n)$ which is subtracted from $d(n)+s(n)$. The goal is to replace the echo impulse response such that $e(n)$ is minimized.

Notes:

One additional application which we have not discussed is the **adaptive line enhancer (ALE)** which is very useful in **tracking narrowband signals** which get superimposed by **wide band noise**. This application (ALE) using the AF is finding its use not only in wide band noise cancellation, but also in **GPS (Global Positioning Satellite) receivers**.



In a GPS application, the GPS receiver keeps a record of the geo-stationary positions of all the **24 US military navigational satellites**. When it is in **line-of-sight** with a satellite, it calculates its relative position on the earth based on a clock reference, extracted from the radio signal transmitted by that satellite. **4 such satellites** are used to accurately determine the GPS 3D location on earth as shown below.



An adaptive filter using the LMS algorithm (similar to the ALE configuration) is used to **adaptively track the averaged position** (4 line of sight satellites give 4 sets of regions of interest – 3 satellites give 1 region of interest) **of a GPS receiver** on earth when the receiver **loses lock** with any of the 4 line of sight satellites. The **Russian** equivalent of GPS is **GLOSNASS** whose receivers use similar techniques. Also **Euro-Land** is launching (by 2008) its own version of a **commercial 30 satellite navigation system**.

APPLICATIONS in IMAGE PROCESSING

Some of the applications falling under Image Processing can be broadly classified as shown below,

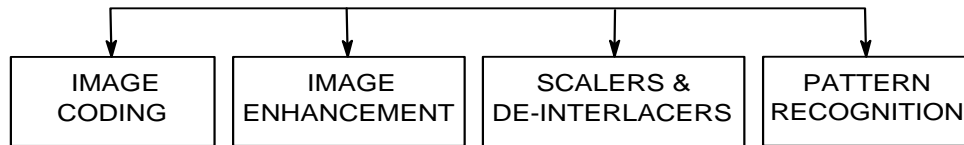


Image Coding

We will discuss 2 types of imaging coding techniques,

- **Run Length Encoding** – is a lossless algorithm that only offers good compression ratios in some types of text, line art and raw image data. It replaces strings of the same data values within a series by a count number and a single value.

Given below is a uncompressed string of data (17 bytes) which has to be compressed using RLE,

ABBBBBBBBCDEEEEF

The compressed string takes (10 bytes) and is shown below,

A *8B C D *4E F, where (*) is the control character.

As you can see, repetitive strings of data are replaced by a control character (*) followed by the number of repeated characters and the repetitive character itself. If the control character appears in the string to be compressed then one additional character is coded.

To decode a RLE compressed string the algorithm basically looks for the number after the **control character** and replicates the following character or numerical following it number times (**character/numerical * number**)

As you can see, RLE encoding is only effective if there are sequences of 4 or more repeating characters because 3 characters are used to conduct RLE, so coding two repeating characters would even lead to an increase in file size.

We have shown the basic principle of RLE encoding. There are several implementations of RLE, most of which are adapted to the type of data to be compressed.

RLE encoding has the following features,

1. **Easy to implement without excessive CPU power.**
 2. **Used in compressing text files and line art images.**
 3. **Used in compressing RGB or YUV raw images.**
- **Huffman Encoding** – this is the most popular lossless encoding scheme for images in use today and finds applications in transform based image compression schemes like JPEG and Wavelet. This form of variable length encoding is widely used for compressing quantized frequency components after a frequency transformation of time domain data.

Huffman coding replaces fixed bit characters (or coefficients) with variable length bits. The length of the bit string is **inversely proportional** to the **frequency of occurrence** of that character or coefficient.

Example: length of bit string $\approx \log_2(\text{character probability})$

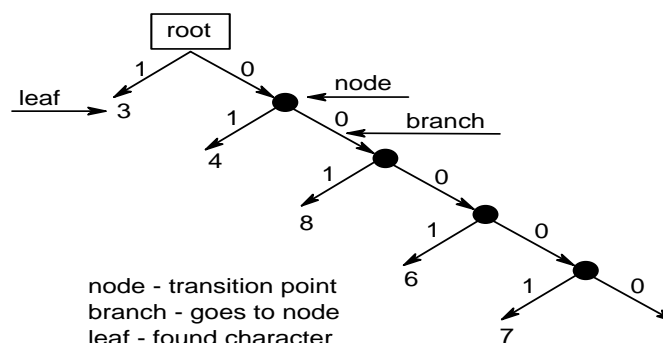
The most standard Huffman encoding scheme encodes 8 bit characters (0xYY – YY being 1 character).

In the encoding process every character is assigned a unique number of bits in order of decreasing probability (**more bits for less probable characters, less bits for more probable characters**).

Let us see how a Huffman tree is created. Suppose you have characters or quantized frequency components as follows, **34346789833**.

In this sequence 3 occurs 4 times, 4 & 8 occur 2 times, 6/7/9 occur 1 times. Since 3 occurs 4 times, we allocate it the least number of bits (to increase coding efficiency) and 6/7/9 is allocated the most number of bits.

Huffman encoding tree, with nodes, branches and leafs is shown below,



Thus, character 3 is “1”, character 4 is “01”, character 8 is “001”, character 6 is “0001”, character 7 is “00001” and character 9 is given the string “00000”.

If the bit strings are concatenated together for the encoded bytes then,

$$4 \times '1' + 2 \times '01' + 2 \times '001' + '0001' + '00001' + '00000' = 4 + 4 + 6 + 4 + 5 + 5 = 28.$$

So instead of encoding 3 4 3 4 6 7 8 9 8 3 3 = 11 bytes * 8 bits = 88 bits, we encode 28 bits.

This gives a encoding efficiency of $88/28 = 3.14$

In Huffman decoding the decoding algorithm has to know the bit strings for each character encoded before decoding the Huffman encoded characters. This information is sent before transmitted the encoded characters.

On obtaining the bit string information, the decoding algorithm from the **root** (start) searches for **ones and zeroes**. A 1 is a leaf and means we have reached the code for a character. A 0 means we are on a branch going towards a node. A node in turn could spring a branch (0) or a leaf (1). This way the entire encoded tree can be searched using a **linked list of nodes** with **each node having a search direction** (leaf or branch).

Image Enhancement

- **Using Standard filters** – standard filters are basically fixed impulse response masked [MxM window] filters which are applied to an image whose **a priori** information is known (basically the user has prior knowledge of the distortion to be removed), to perform functions like bi-leveling, low pass filtering, high pass filtering, median filtering, wiener filtering, edge detection, etc.
- **Using Adaptive Filters** – two most popular adaptive filtering techniques used in image or video processing are listed next,
 1. **Two Dimensional LMS algorithm** – this two dimensional filter is a MxM adaptive filter whose weights are computed adaptively at every input point $x(m,n)$ in the image, using a adaptive algorithm like leaky LMS. This MxM weight window is then convolved with the image MxM window around point $x(m,n)$ to give a output point $y(m,n)$, which is then inserted into the new output image.

This type of filter becomes very important when the signal (image) and the noise source are non-stationary. This filter can be used to enhance images using a typical adaptive line enhancer (ALE) configuration.

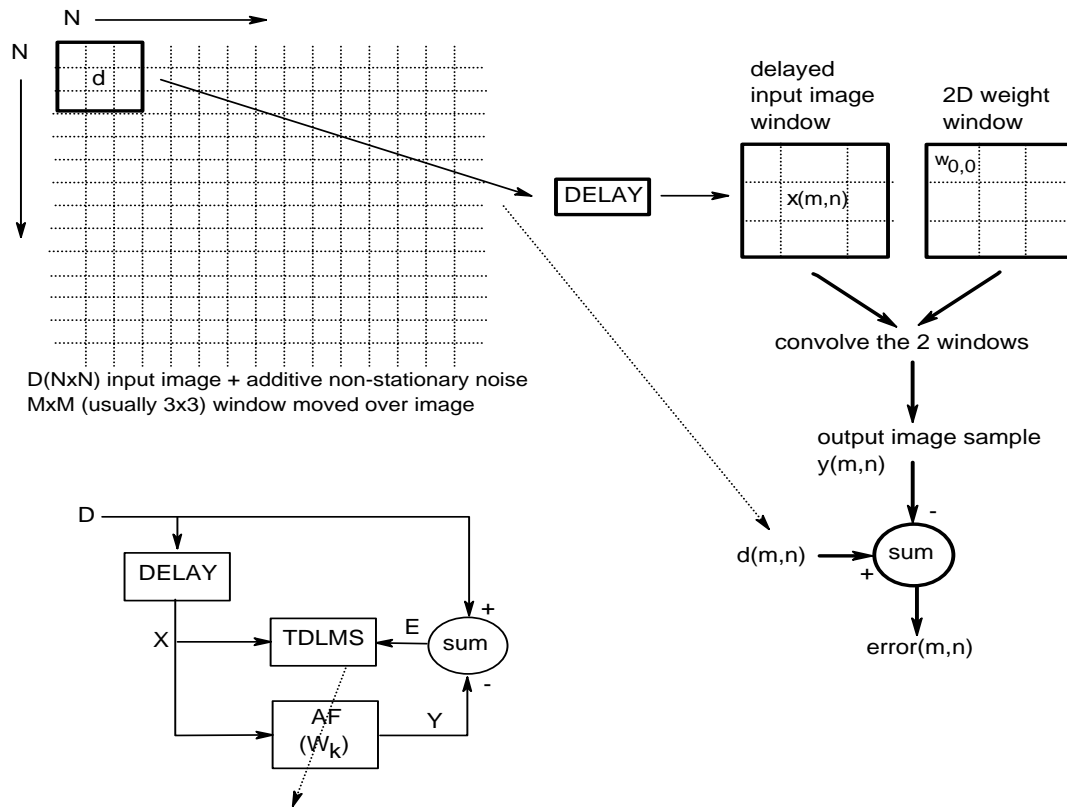


Figure 41. TDLMS applied to an image $D(M,M)$ with additive nonstationary noise.

The equations for the TDLMS algorithm are given below,

$$y(m,n) = \sum_{a=0}^{M-1} \sum_{b=0}^{M-1} [w_i(a,b) * x(m-a,n-b)]$$

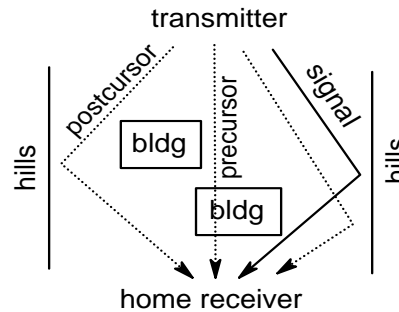
$$error(m,n) = d(m,n) - \sum_{a=0}^{M-1} \sum_{b=0}^{M-1} [w_i(a,b) * x(m-a,n-b)]$$

$$w_{i+1}(a,b) = w_i(a,b) + 2 * \mu * error_i(m,n) * x(m-a,n-a)$$

The output samples $y(m,n)$ is calculated per moving window position over the input image and a final output image ($M \times M$) is made.

2. Ghost Cancellation – this technique is another form of adaptive equalization which is employed in receivers compatible with the ATSC digital transmission used in the US. In this transmission system standard & high definition MPEG compressed video along with Dolby

AC3 compressed audio is packetized, modulated using 8 VSB and sent over the terrestrial (off the air) channel to home receivers. Ghosts can be formed due to the following scenarios shown below.



A transmitted ATSC data frame and its associated frame segment sync is shown below,

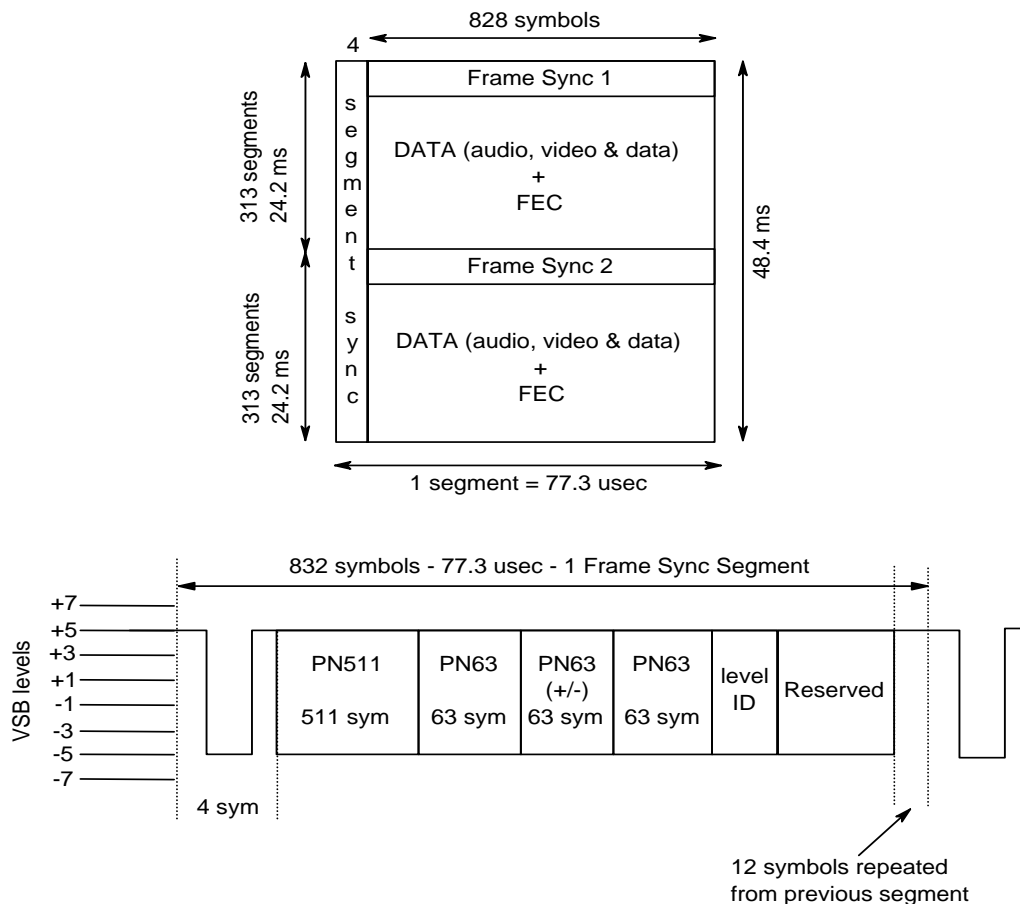


Figure 42. ATSC frame and Frame Sync Segment

When the transmitted ATSC data frame comes over the air as shown from the transmitter to the receiver, it suffers channel distortion which can cause postcursor ghosts (delayed w.r.t the actual signal in time) or

precursor ghosts (before the actual signal in time). The **equalizer** or **ghost canceller** has to remove these ghosts to provide **maximum eye opening** for optimal data slicing.

In this type of adaptive equalization scheme (a.k.a. ghost cancellation) we use the standard FIR transversal filter for 4 reasons,

- simple and deterministic response
- excellent for handling pre-cursor ghosts
- large taps possible today, thus handles long post-cursor ghosts
- IIR structure can become unstable

In a VSB system the clock recovery is done using the segment syncs and since the position of the frame sync is known, a **T spaced or T sampled** equalizer (basically a equalizer that is not over-sampled) is possible by spacing the equalizer FIR taps apart by **1/(data symbol rate)**.

We use the standard 2 modes of equalization namely training mode and data directed mode.

- **Training Mode** – the use of a known reference training signal allows the creation of an error signal that is always valid, regardless of the ghost size or noise amplitude, since no data slicing or estimation is needed. Even with completely closed eyes, the algorithm can still converge on a solution since the clock recovery loop is locked and the reference training signal location is known. The long **511 symbol pseudo-random sequence** at the beginning of the frame sync provides the equalizer a spectrally flat signal (flat across all frequencies) for correlating with an identical internal stored PN sequence. This pseudo-random sequence can cover up to 48usec ghosts. This is done every time a frame sync segment is obtained.
- **Data Directed Mode** – once the Training is completed and the filter weights have converged forming the inverse channel response, the system is switched into data directed mode wherein the sliced data is switched into the loop (this is when ghost cancellation occurs)

There is one more mode called **blind equalization** which can supposedly work **without the presence** of a **training signal** (widely used in QAM based cable systems in Europe or USA) and is sometimes used instead of the **training mode**. In this technique the equalizer models the received multi-level signal (could be VSB or QAM) as a binary data signal plus noise. With enough averaging the equalizer can open the data eyes to the point where data-directed equalization can be used. This technique however has increased susceptibility to spurious noise.

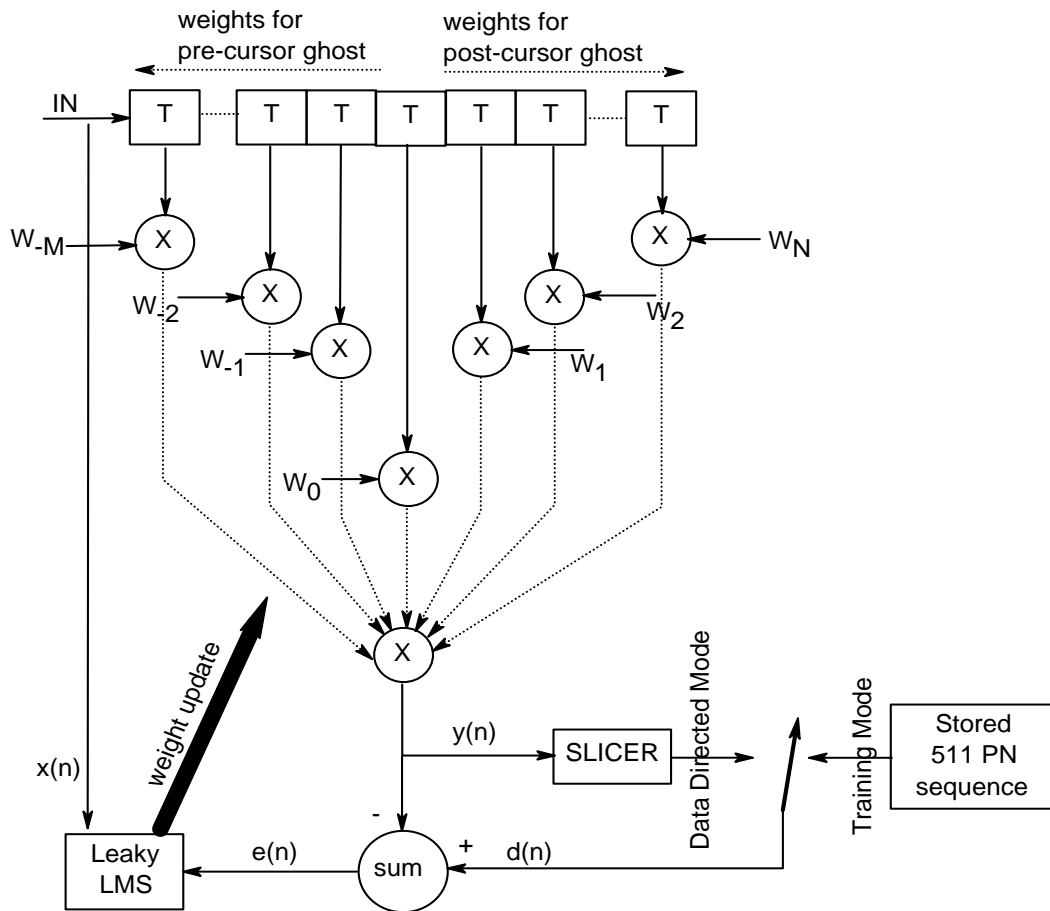


Figure 43. Equalizer used in ATSC receivers

The PN63 psuedo-random sequences are only used in cable systems where the transmission is 16VSB and the ghosts are in the range of ~6 microseconds. Cable systems using 16VSB have not been employed thus far. The most popular or widely used format for cable systems is 64 or 256 QAM.

Scaling, Format Conversion, De-interlacing, Line Doubling, Aspect Ratio Conversion and Color Space Conversion

Scaling is typically used to decimate (lower the sample rate) or interpolate (increase the sampling rate) of images or video in both the horizontal (pixels) and vertical (lines) directions. Scaling is typically applied in 2 areas,

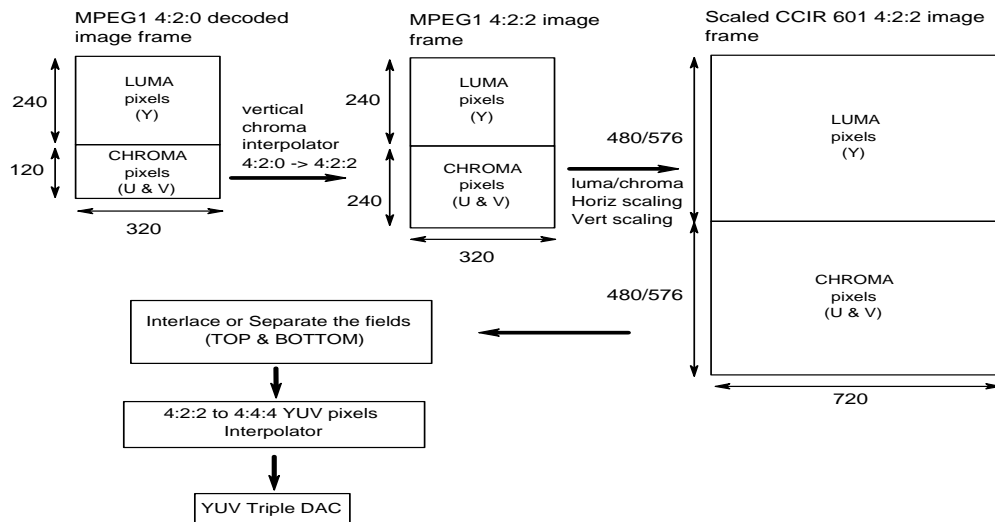
1. **Scaling of Computer Graphics** – this type of scaling is used when the display screen resolution is fixed (e.g. a computer display adapter selects a specific screen resolution – pixels & lines - by adjusting the horizontal

and vertical display scan rates in Hz). All other graphics is scaled within this screen resolution **without adjusting or changing** the display's **horizontal or vertical syncs**. Basically the background screen has a fixed number of pixels and lines. If the display adapter's screen resolution is changed by the user, then all graphics (including bitmapped images and video) superimposed on it either shrinks (if the resolution is increased) or expands (if the resolution is decreased).

So when a bitmapped frame/s is to be scaled horizontally and vertically, before being displayed on this screen, we apply either a moving bi-linear (2x2 pixel/line neighborhood) or a bi-cubic (4x4 pixel/line neighborhood) fit to that frame/s, remove (decimate) or add (interpolate) pixels or lines to/from the original frame/s. We have discussed the implementation of a bi-cubic interpolating filter before which can be applied to any bitmapped frame/s for display on a computer screen. This is the method in which a frame or video is scaled and displayed on a computer screen.

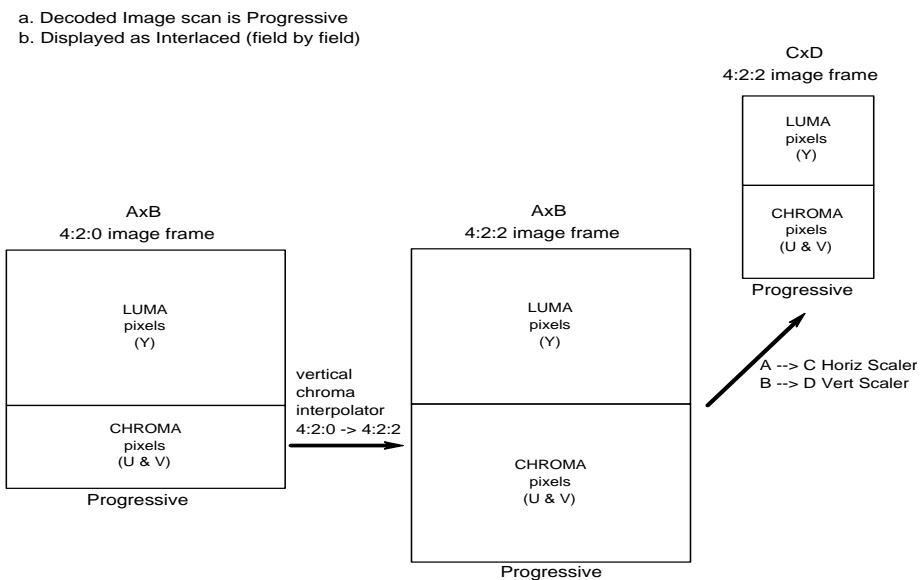
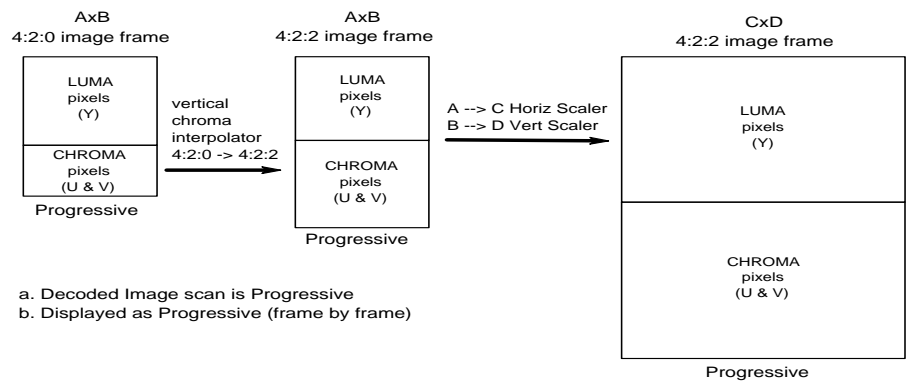
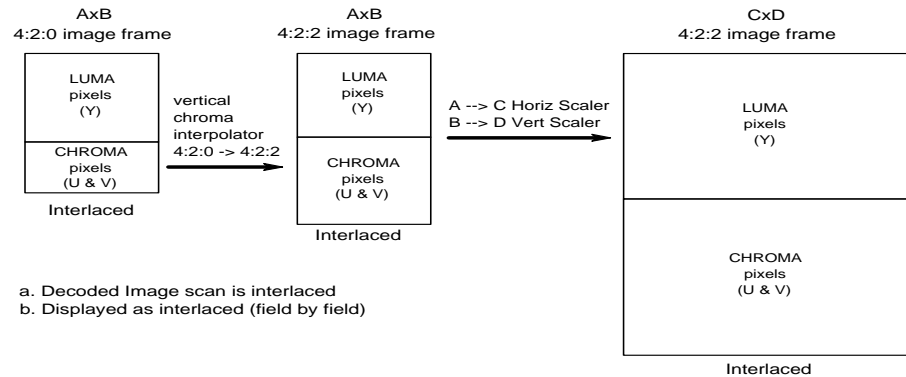
2. **Format Conversion of Video** – In this technique the scaled video is made to fill the screen dimension. The input resolution is converted to the output resolution by decimating or interpolating pixels & lines, such that the pixels & lines fit the new scan rate. The formats are changed by changing the scan rate. The video is usually scaled to fit the scan rate.

Example 1: If you have an input resolution typically found in MPEG1 decoded frames 320x240 progressive, and it needs to be displayed on a NTSC or PAL monitor display, then the pixels have to be interpolated to standard CCIR 601 resolution which is 720 pixels and the lines have to be interpolated to 480 lines (NTSC) or 576 lines (PAL). This output resolution is then interlaced (frames to fields) and fitted within the horizontal/vertical scan rates of NTSC or PAL which is fixed for both these standards.



Example 2: In ATSC, standard and high definition video could be in any input resolution and format, which may need to be converted into an appropriate output resolution and format,

1. AxB dimensions in interlaced format → CxD dimensions in interlaced format
2. AxB dimensions in progressive format → CxD dimensions in progressive format
3. AxB dimensions in progressive format → CxD dimensions in interlaced format



Above we have shown 2 interpolations and 1 decimation in Horizontal and V ertical directions.

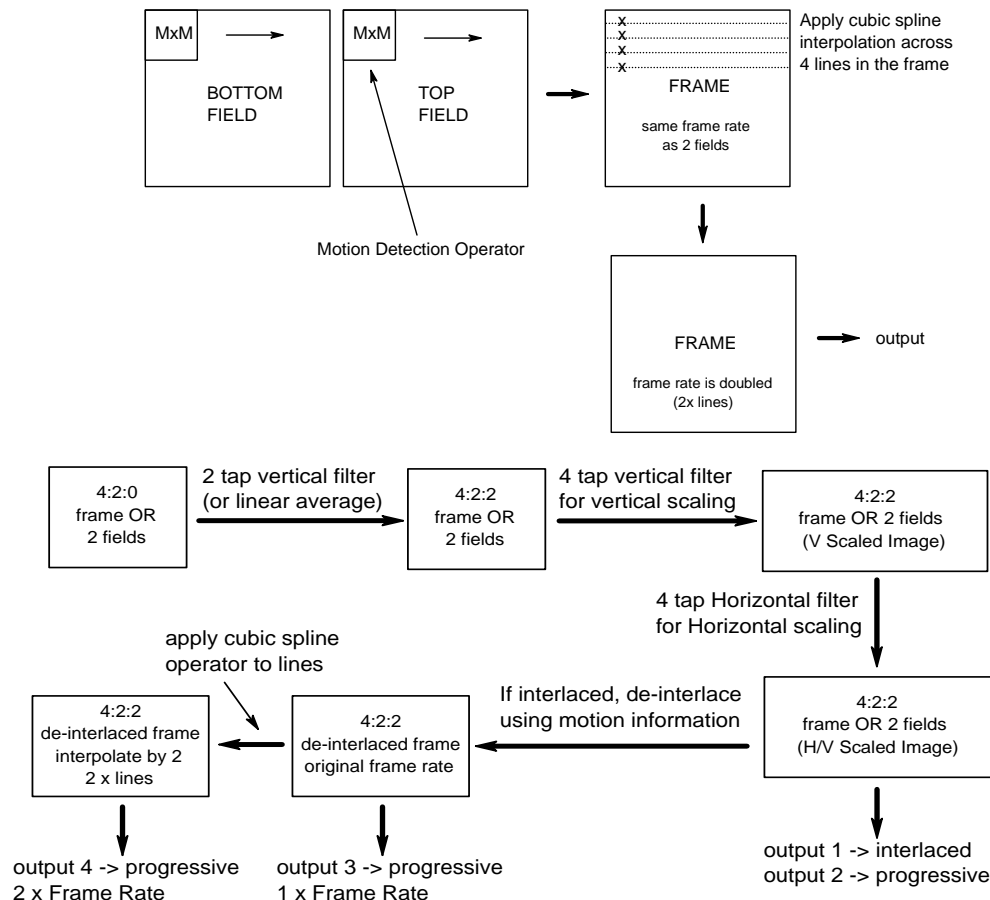
3. Interlaced to Progressive Scan or line doubling – both these techniques involve an interlaced to progressive scan conversion with or without frame rate up conversion (or line doubling).

1. $A \times B$ dimensions in interlaced format $\rightarrow C \times D$ dimensions in progressive format at the same original frame rate.
2. $A \times B$ dimensions in interlaced format $\rightarrow C \times D$ dimensions in progressive format at twice the original frame rate.

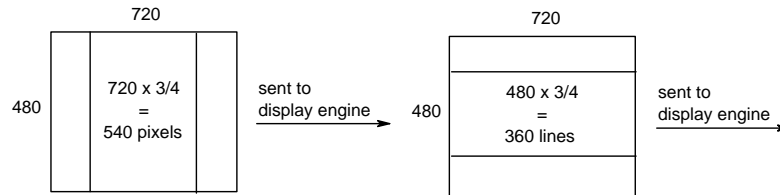
Both these techniques involve **MOTION-DETECTION** which works across the 2 interlaced fields (top & bottom) stored in memory to make a frame,

- A 2D window ($M \times M$ dimension block) is moved over both fields (LUMA only), the pixels in them are averaged and compared against a Threshold.
- If this comparison is greater than the Threshold Motion Value, then it is assumed that motion exists between those 2 luma blocks and the new pixel in the interpolated frame line is identical to the pixel in Top field.
- If this comparison is less than the Threshold Motion Value, then it is assumed no motion exists between the 2 luma blocks and the new pixel in the interpolated frame line is the average of the 2 pixels in the 2 fields.

In this way the 2 fields (top & bottom) are de-interlaced into a frame. If the frame rate is to be doubled, then after de-interlacing, the lines in the frame are interpolated by a cubic spline operator to generate twice the amount of original lines which are outputted at twice the frame rate.



- 4. Aspect-Ratio Conversion** – this is usually needed when an input picture having a source aspect ratio of 4:3 is to be displayed on a screen having an aspect ratio of 16:9 or vice-versa. In MPEG2, **pan & scan** vectors are provided in the picture display extension header to do these conversions.



720x480 (4:3) on a 16:9 TV screen **720x480 (16:9) on a 4:3 TV screen**

- 5. Color Space Conversion** – used to convert between the YUV and RGB color components. There are 2 conversion matrices (CCIR601 and CCIR709) out of which we have shown CCIR601,

CCIR 601 CSC matrix,

$$R = 1.0000 * (Y-16) + 0.0000 * (U-128) + 1.3828 * (V-128)$$

$$G = 1.0000 * (Y-16) - 0.3359 * (U-128) - 0.6992 * (V-128)$$

$$B = 1.0000 * (Y-16) + 1.7344 * (U-128) + 0.0000 * (V-128)$$

$$Y = 0.2988 * R + 0.5869 * G + 0.1143 * B$$

$$U = -0.1729 * R - 0.3389 * G + 0.5107 * B + 128$$

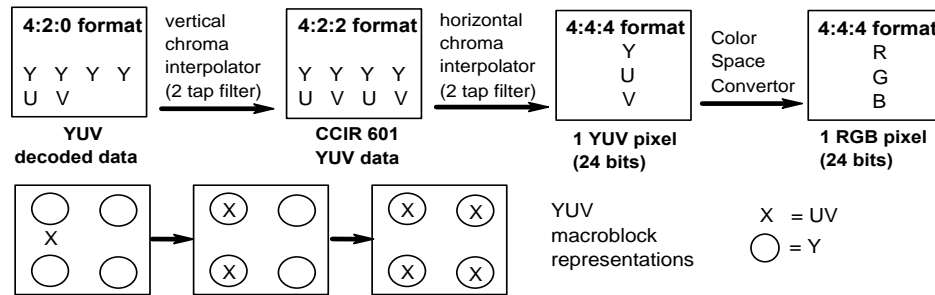
$$V = 0.5107 * R - 0.4277 * G - 0.0830 * B + 128$$

Notes:

1. CCIR 601 video is an 8 bit stream of 4:2:2 YUV components (or D1 component video) without the H/V sync information. The syncs are provided externally to any hardware accepting CCIR 601 video.
2. CCIR 656 video is CCIR 601 YUV components with embedded H/V syncs (as SAV – start of active video and EAV – end of active video).
3. High definition video can be outputted as a stream of 4:4:4 YUV/RGB 24 bit pixels with external H/V syncs.
4. High definition video can also be outputted as a stream of 4:2:2 16 bits YUV pixels with embedded H/V syncs (SAV & EAV) in the SMPTE274M or SMPTE295M format.

EAV code	Ancillary or Blanking codewords	SAV code	Active Video YUV data
-------------	------------------------------------	-------------	-----------------------

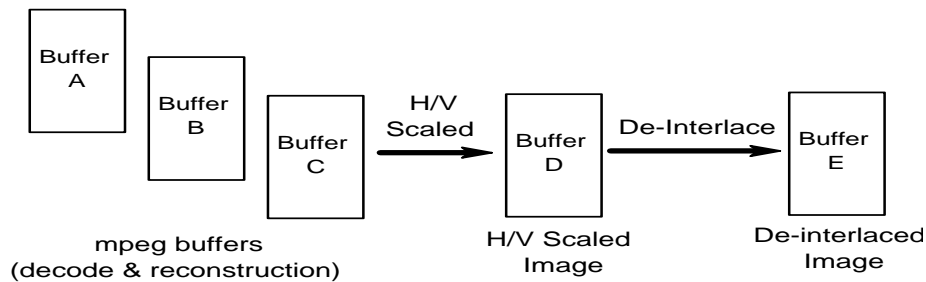
5. Below we have shown the construction of YUV pixels,



6. Most software/hardware decoders which decode SD/HD MPEG streams having interlaced/progressive resolutions of 1920x1088 pixels, use additional 2 full resolution 4:2:2 YUV buffers to store the intermediate stages for format conversion, de-interlacing and frame rate up-conversion. So the total number of buffers can be,

$$3 * 1920 \times 1088 * 12 \text{ bits} - (4:2:0 \text{ buffers})$$

$$2 * 1920 \times 1088 * 16 \text{ bits} - (4:2:2 \text{ buffers})$$



This configuration gives the best system performance (~32 Mbytes, with more than enough storage for additional full resolution graphics buffers to BLIT with the video frames. It also allows enough storage for code/heap/stack in an embedded application) without the need to store any “lines” internal to the SD/HD decoder.

7. Most memories connected to SD/HD mpeg decoders are DDRAM, which fetch data on both edges of the clock cycle, effectively doubling the data fetch rates.

Pattern Recognition

1. **Optical Character Recognition (OCR)** – is a technique by which typed text in any form is captured by a scanner or camera, pre-processed using some image processing techniques like noise removal, segmented into individual characters or numbers and detected by pattern recognition algorithms.

Optical character recognition using the **Walsh transform** can be split into the following steps,

- Capture the text using a scanner or camera
- Convert the image into a binary format (0/1) using a suitable threshold
- Segment the text into individual characters or numbers
- Apply the Walsh transform to extract features from the segmented text
- Use a pattern recognition technique to detect/recognize characters and numbers

Let us consider the below image which we have pre-processed using a noise removal filter, converted into a binary image using a suitable threshold and segmented using boundary parameters like size of the rectangle (this can be looking at the 1/0 delineation) within-holding them.

figyelő

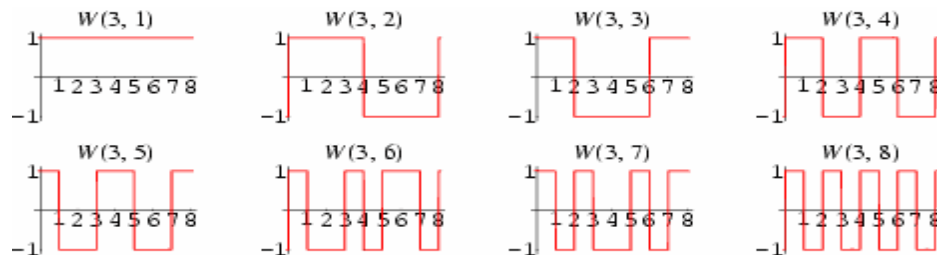
If the binary image $f(x,y)$ is of size $N \times N$, its Walsh transform is shown below,

$$W(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [f(x,y) * g(x,y,u,v)]$$

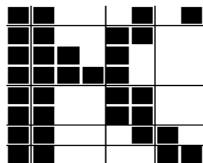
The Walsh Transform coefficients will be,

$$\begin{aligned} &W(0,1), W(0,2), \dots, W(0,N-1) \\ &W(1,1), W(1,2), \dots, W(1,N-1) \\ &\dots \\ &W(N-1,1), W(N-1,2), \dots, W(N-1,N-1) \end{aligned}$$

Walsh functions consist of square pulses (with the allowed states being -1 and +1), such that the transitions only occur at fixed intervals of a unit time step. There are 2^n Walsh functions of length n , illustrated for $n = 3$ (or $N=8=2^3$).



We now apply the Walsh Transform to a “partial 8x8 window” in the above image to get the 8x8 transformed coefficients as shown below,



The final step is detecting the feature set extracted by the Walsh transform using some form of correlation technique with reference characters or numbers.

- 2. Fingerprint Recognition** - fingerprints are made up of a series of ridges and furrows on the surface of the finger. The uniqueness of a fingerprint can be determined by the pattern of ridges and furrows as well as the minutiae points. Minutiae points are local ridge characteristics that occur at either a ridge bifurcation or a ridge ending.

Fingerprint matching techniques can be placed into two categories,

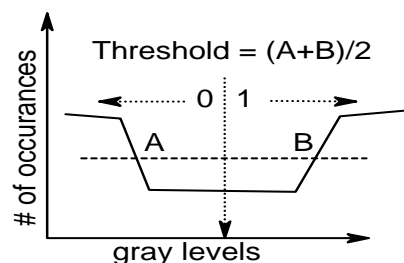
- 1. Minutiae based** - Minutiae-based techniques first find minutiae points and then map their relative placement on the finger. However, there are some difficulties when using this approach. It is difficult to extract the minutiae points accurately when the fingerprint is of low quality. Also this method does not take into account the global pattern of ridges and furrows. Fingerprint matching based on minutiae faces problems in matching different sized (unregistered) minutiae patterns. Local ridge structures can not be completely characterized by minutiae.
- 2. Correlation based** - The correlation-based method is able to overcome some of the difficulties of the minutiae-based approach. However, it has some of its own shortcomings. Correlation-based techniques require the precise location of a registration point and are affected by image translation and rotation.

The fingerprint detection algorithm shown below uses the **minutiae based method** and has the following steps,

- **Image enhancement (contrast stretch) and noise filter**
 - **Threshold the image into 2 levels (make binary)**
 - **Skeletonize the image**
 - **Chain coding**
 - **Creating a direction matrix**
 - **Creating a prototype set**
 - **Comparing the created prototype set with reference prototype set**
1. The Image is gray scaled, contrast stretched, a histogram is generated, a suitable threshold is chosen and the image is converted into binary form (1/0).



Before Contrast Stretching



After Contrast Stretching

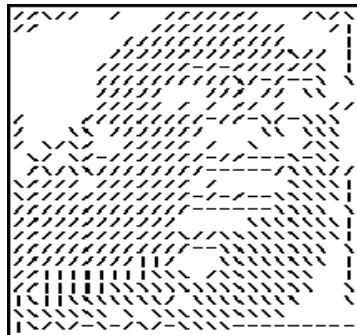
2. Skeletonize the fingerprint ridges by thinning them to 1 pixel wide.



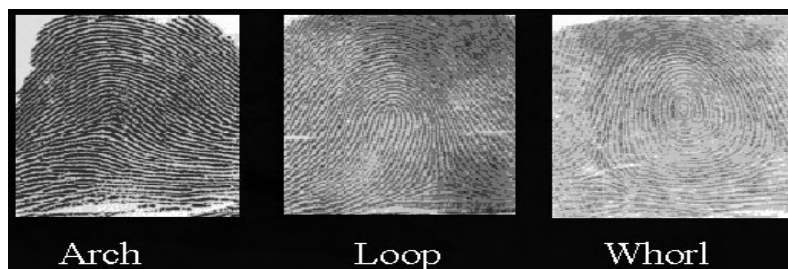
3. Now get information about the various directions in the skeletonized image, find skeletonization bugs like dis-connections, fake ridge branches and also find the occurrences of minutiae (bifurcations and ridge endings).
4. Chain code by assigning direction values to the pixels in the skeleton as follows.

0°	0	—
45°	1	/
90°	2	
135°	3	\
180°	0	—
225°	1	/
270°	2	
315°	3	\

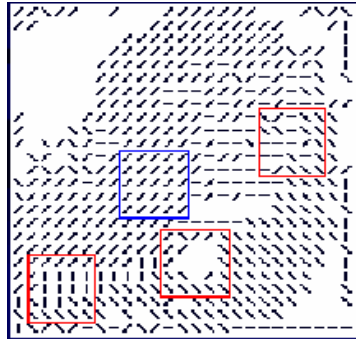
5. Average the chain code values to form a direction matrix for analysis.



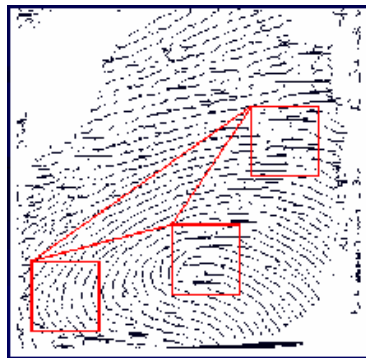
6. The direction matrix preserves the main feature types of a fingerprint.



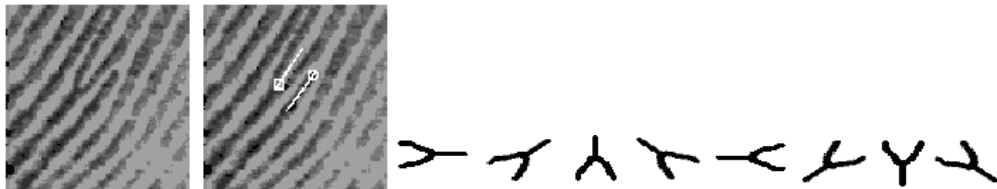
7. Search for locations (or minors) which have largest difference from their surroundings, use Hamming distance or Euclidean distance to calculate the distance values. BLUE is a minor where the surrounding differences are unchanged, whereas RED are minors where the hamming distances are large.



8. Check the minor's positions by counting ridges along the line segments.



9. Search for more detailed classifiers like bifurcations, their orientations and ridge endings.



10. Now our prototype set contains a “direction matrix”, “minor positions”, “minor position differences in ridge counts”, “bifurcations information” and “ridge endings information”.

With all these elements in the prototype set of the fingerprint just analyzed, we can compare it with **reference prototype sets of fingerprints**, to see how good of a match has been obtained.

3. **Handwriting Recognition** – this technique is widely used in handheld devices and consists of 2 modes, training mode and recognition mode. In the **training mode** the owner of the handheld is asked to input a wide

range of characters and numbers to extract a set of unique characteristics about his handwriting. These characteristics are then used in the **recognition mode** to match the characters and numbers inputted by the owner during normal usage of the handheld.

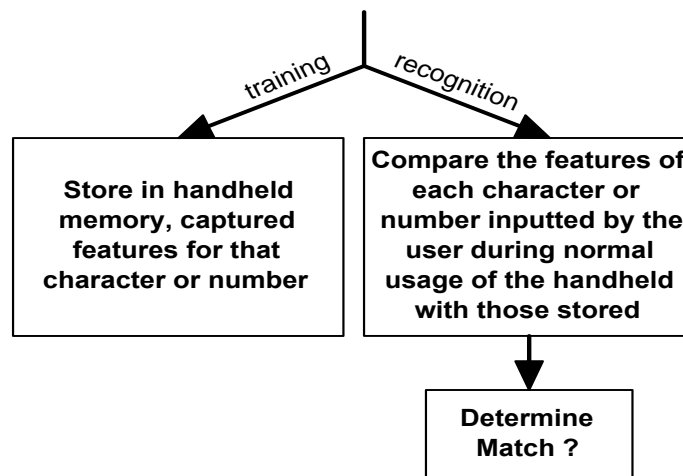
Both the “**training and recognition**” modes have the following steps,

1. Preprocessing (training & recognition)

- Digitizing the inputted character or number
- Applying a noise removal filter
- Contrast stretching the image
- Converting the image into 2 (1/0) binary levels
- Thinning the image to few pixels

2. Feature Extraction (training & recognition)

- Segmenting the image into individual characters/numbers
- Apply the Walsh transform to each segmented character/number
- Capture the major features of each character & number

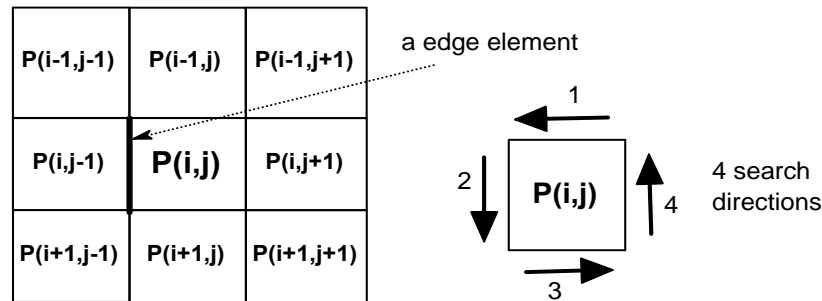


4. **Contour Tracing** – this technique is mainly used for tracing the contour of single or many objects captured by a camera. It basically involves drawing a 1 pixel wide contour around the perimeter of an object or objects. It finds its use in industrial applications to measure and classify objects on an assembly line, and also in medical applications to recognize shapes & sizes of different cells, tumors & lesions.

The steps involved in contour tracing are outlined next,

- Capture the image using a industrial camera
- Preprocess the image using a noise removal filter or pixel averaging using a 3x3 neighborhood

- Calculate the histogram of the image and choose a threshold value to bi-level the image into 2 binary levels (0/1)
- Trace the contour of the image using the below algorithm



The contour of an object in a $N \times M$ image is obtained by calculating the gradients over pixels in a 3×3 window placed over each reference pixel $P(i,j)$, and moving the 1st reference pixel east, west, north or south till the 1st reference pixel is reached or the end of the image is reached.

1. Find a FIRST reference point A on the bi-leveled image such that $A = P(i,j) = 0$
2. Set the 1st search direction to $S=2$
3. Now go over the entire image using the "search or find direction S", till end of image ($N-1, M-1$) is reached OR FIRST reference point A is reached

1. Gradient 1 = $\text{abs} [P(i,j-1) - P(i-1,j-1)]$
2. Gradient 2 = $\text{abs} [P(i,j) - P(i,j-1)]$
3. Gradient 4 = $\text{abs} [P(i-1,j-1) - P(i-1,j)]$

If Gradient 1 = 1, then set $S=1$ and current point $P(i,j)$ to $P(i,j-1)$

If Gradient 2 = 1, then set $S=2$

If Gradient 4 = 1, then set $S=4$ and current point $P(i,j)$ to $P(i-1,j-1)$

1. Gradient 1 = $\text{abs} [P(i+1,j-1) - P(i,j-1)]$
2. Gradient 2 = $\text{abs} [P(i+1,j) - P(i+1,j-1)]$
3. Gradient 3 = $\text{abs} [P(i,j) - P(i+1,j)]$

If Gradient 1 = 1, then set $S=1$ and current point $P(i,j)$ to $P(i+1,j-1)$

If Gradient 2 = 1, then set $S=2$ and current point $P(i,j)$ to $P(i+1,j)$

If Gradient 3 = 1, then set $S=3$

1. Gradient 2 = $\text{abs} [P(i+1,j+1) - P(i+1,j)]$
2. Gradient 3 = $\text{abs} [P(i,j+1) - P(i+1,j+1)]$
3. Gradient 4 = $\text{abs} [P(i,j) - P(i,j+1)]$

If Gradient 2 = 1, then set $S=2$ and current point $P(i,j)$ to $P(i+1,j+1)$

If Gradient 3 = 1, then set $S=3$ and current point $P(i,j)$ to $P(i,j+1)$

If Gradient 4 = 1, then set $S=4$

1. Gradient 1 = $\text{abs} [P(i,j) - P(i-1,j)]$
2. Gradient 3 = $\text{abs} [P(i-1,j+1) - P(i,j+1)]$
3. Gradient 4 = $\text{abs} [P(i-1,j) - P(i-1,j+1)]$

If Gradient 1 = 1, then set $S=1$

If Gradient 3 = 1, then set $S=3$ and current point $P(i,j)$ to $P(i-1,j+1)$

If Gradient 4 = 1, then set $S=4$ and current point $P(i,j)$ to $P(i,j-1)$

4. Once the contour is traced over the perimeter of an image, the points in it can be used for all sorts of measurements.

MODULATION TECHNIQUES

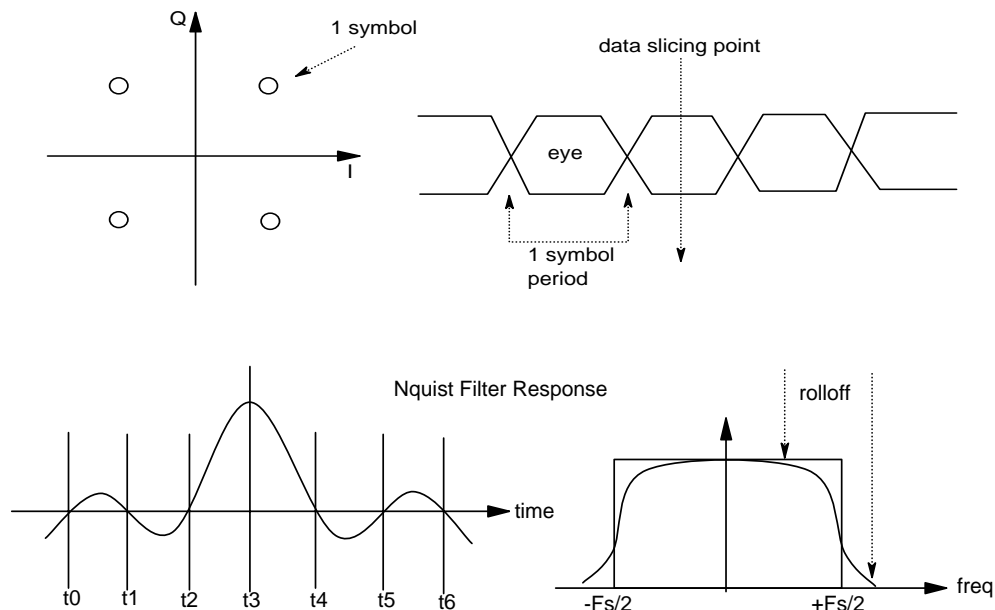
A number of modulation schemes exist for **up-converting** and **modulating digital base-band signals** (audio, video and data) for transmission over telephone lines, coaxial cable, satellite and terrestrial (over the air) medium. Each of these modulation schemes work best over a particular medium depending on the medium's bandwidth, echoes and distortion capability. Each of these modulation schemes consist of sophisticated error coding techniques and methods for neutralizing the channel response and degradation. We will discuss the following methods,

- **QPSK** – Quadrature Phase Shift keying
- **QAM** – Quadrature Amplitude Modulation
- **VSF** – Vestigial Side-Band Modulation
- **COFDM** – Coded Orthogonal Frequency Division Multiplexing

- **Forward Error Correction (FEC) using Concatenated Codes**
- **Turbo coding instead of Inner Coding in a FEC**

QPSK (Quadrature Phase Shift Keying)

In QPSK modulation we have 4 symbols (one per quadrant in the I/Q domain) and each symbol is 2 bits. The below figure shows a QPSK constellation and the eye diagram shows when the data points are sampled or sliced (maximum eye opening) to avoid inter-symbol interference.



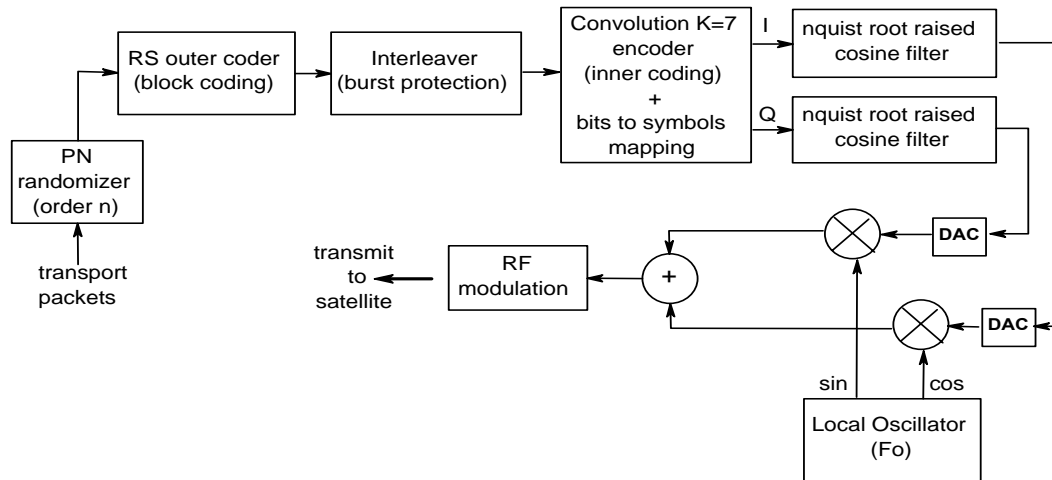
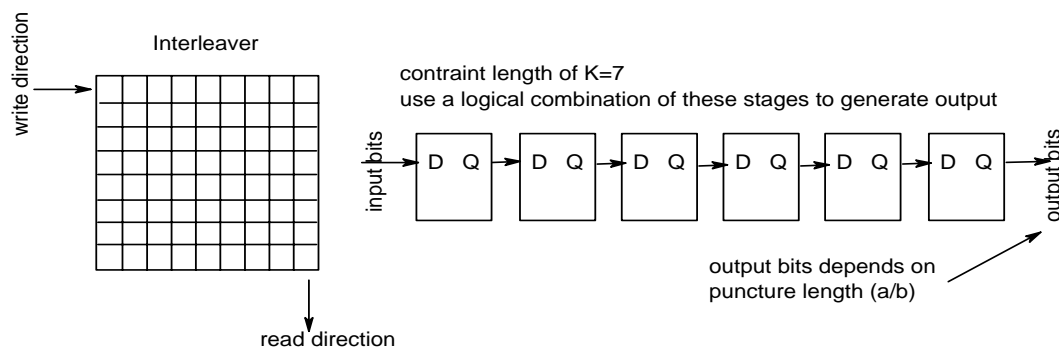


Figure 44. QPSK modulation scheme for transmission of MPEG2 packetized transport data

In a typical QPSK transmitter used in the digital broadcasting standard (DVB), the MPEG2 encapsulated transport data is first randomized using a PN randomizing polynomial (order n) and then passed through a outer coder (Reed Solomon) which is a block coder adding error codes (or redundant bytes) to certain lengths of data. The RS block code used is usually of the type (204,188) over the Galois Field ($2^{\text{bits_in_byte}}$), where 16 redundant/parity RS bytes can correct up to 8 byte errors. This data is then sent to an interleaver which interleaves the RS data symbols (rows to columns) depending on the interleaving depth specified by the transmission standard. This provides excellent protection for burst errors occurring in a channel. The goal is to take an error burst across a row of data and spread it across columns. From here the data bits go into an inner coder (convolutional encoder, or soft encoder) which has a constraint length of 7 ($k=7$). This adds additional redundant bits to the data depending on the puncture length (e.g. if puncture length or redundancy is $\frac{3}{4}$, then for every 3 input bits we generate 4 output bits). The constraint length of 7 implies that 7 consecutive input bits are used to generate each output bit.



The output bits are now mapped into QPSK I/Q symbols using a bits to symbol mapping technique. The resulting I & Q symbols are shaped using digital nyquist root raised cosine filters for better channel immunity. Each signal is now converted from the digital domain into the analog domain, IF modulated and up-converted using 90° phase shifted carriers and then summed together. Finally the resulting signal is RF modulated in the GHz range and transmitted from the head end to the satellite.

The satellite beams this signal down (point to point – satellite to dish) to a X inches dish installed in a user's home which is connected via a coaxial cable to an integrated receiver which demodulates this signal using a QPSK demodulating scheme as shown below,

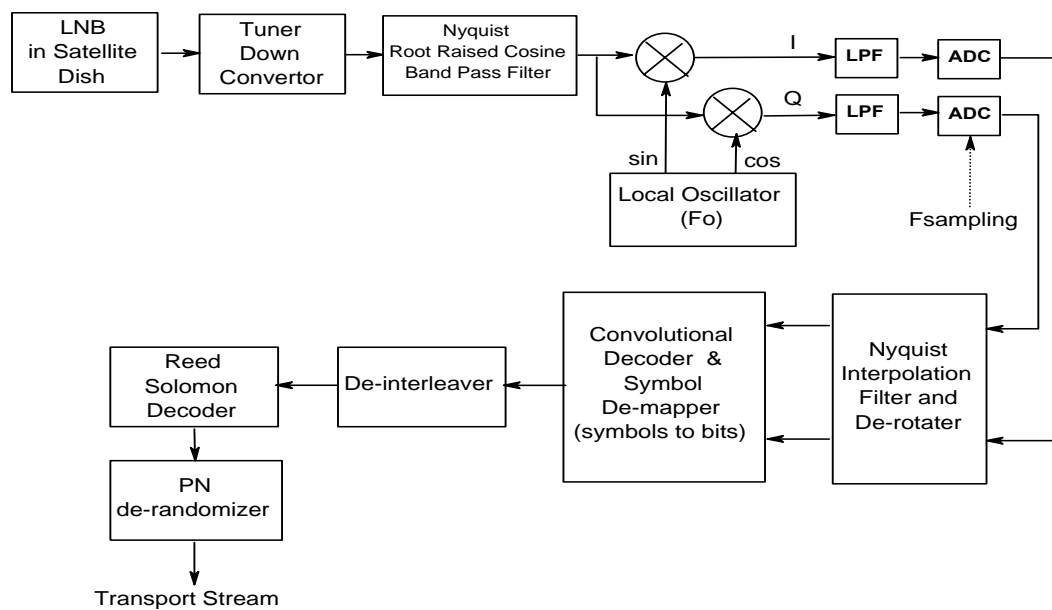


Figure 45. QPSK de-modulation scheme for reception of MPEG2 packetized transport data

In the satellite dish there is a LNB (low noise block) which receives the signal (GHz range) from the dish, down-converts it and sends it to the tuner for further down-conversion. The IF signal is passed via a SAW filter (BW country or region dependant) or sent directly (in case of ZERO IF demodulator) into a band-pass nyquist filter and then to a local demodulator which separates the I and Q components. The I and Q components are then low pass filtered and digitized. Before slicing, a possible simple equalization scheme could be applied for ISI removal. These components are then sent to a nyquist interpolation filter and then to the derotater which calculates the residual phase error in the I & Q components and compensates (rotates the I/Q components appropriately) for this phase error. The convolutional decoder (soft decoder) using the viterbi algorithm extracts (using soft decisions and trace memory) the data symbols and maps the symbols to data bits. This is then sent into the de-interleaver which de-

interleaves the columns into rows and sends it to the Reed Solomon decoder. The Reed Solomon decoder works on a block of bytes (transport bytes + RS added parity bytes) to detect and correct up to 8 byte errors in the transport stream. The transport stream is then de-randomized using the identical polynomial used at the transmitter.

QPSK bit-rate is measured per transponder depending on the symbol rate it can provide. A satellite usually has several transponders over which the entire transport stream multiplex (consisting of programs & program specific information) is split.

Transponder total bitrate \rightarrow symbols/sec * bits/symbol = symbols/sec * 2

Transponder useful data bitrate \rightarrow symbols/sec * 2 * puncture rate * RS factor

Puncture rate = convolutional encoder's (input bits / output bits)

RS factor = (data bytes in RS block) / (data bytes in RS block + RS parity bytes)

Advantages of QPSK transmission,

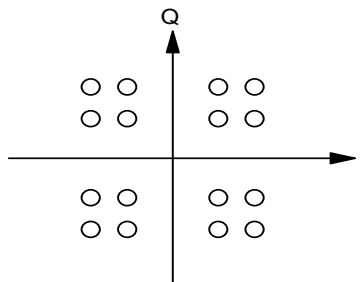
- In a satellite channel (point to point) the impulse response is zero (no echoes), so we do not need an equalizer for removing inter symbol interference (opening the eyes in the eye diagram) or channel ghosts.
- No specific power requirements.
- Large bandwidth available.
- Signal to noise ratio is quite good.

Where used,

- For satellite downstream.
- For cable upstream return channel.

QAM (Quadrature Amplitude Modulation)

A typical 16 level QAM constellation is shown below. QAM can take any constellation from 16, 32, 64, 128, 256 levels (where a level = $2^{\text{bits/symbol}}$). The bigger the level of the QAM constellation used the more bandwidth (bit rate per channel) it can provide over cable channels.



A typical QAM modulating scheme is shown below,

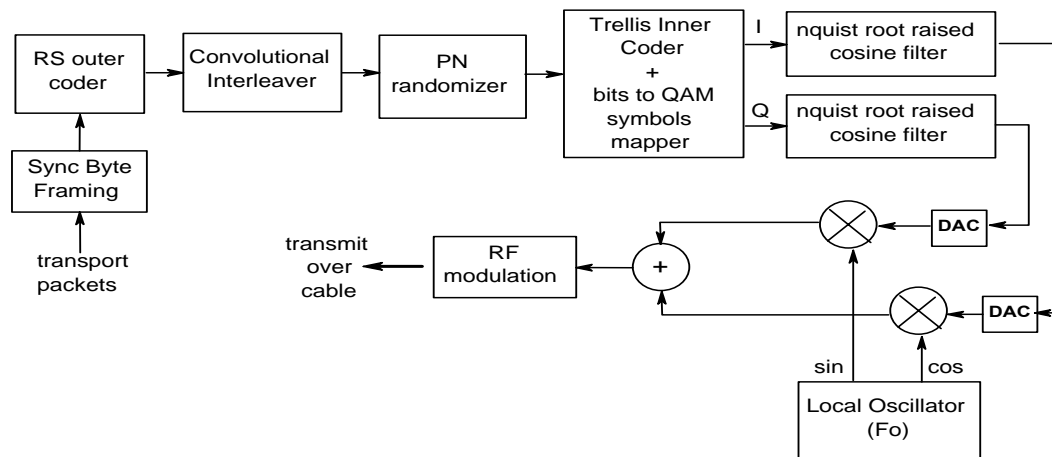
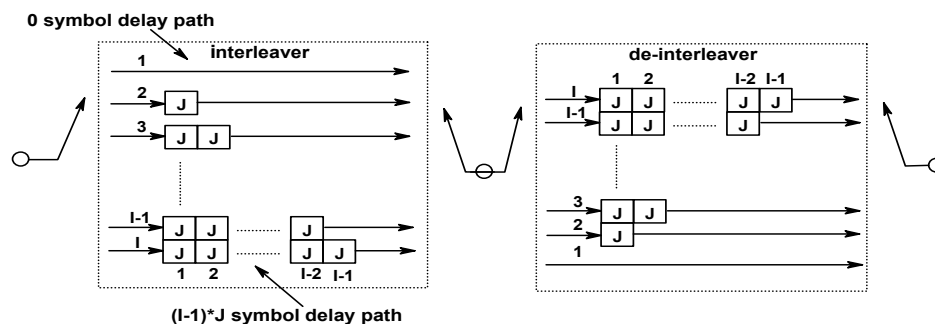
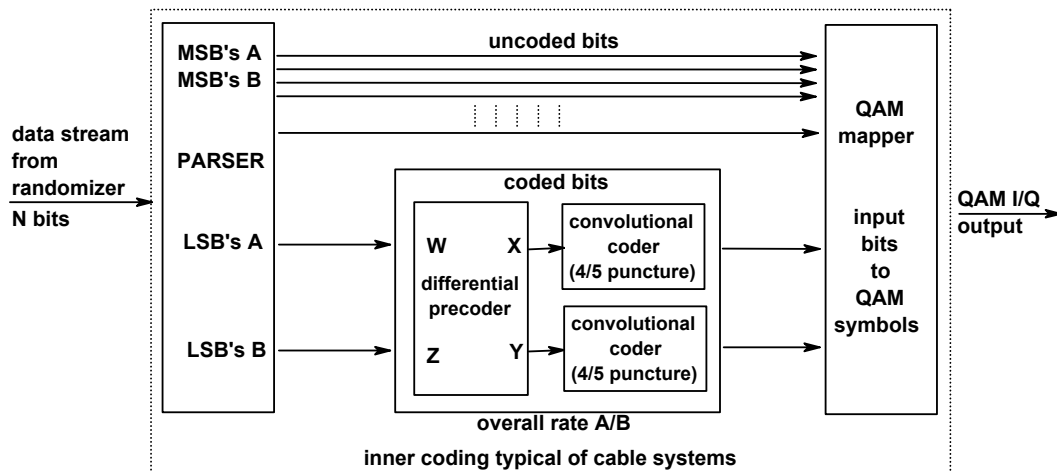


Figure 46. QAM modulation scheme for transmission of MPEG2 packetized transport data over a coaxial cable

An MPEG2 transport stream (DVB standard, 188 bytes packets) is taken and a sync framing algorithm (parity checksum using a linear feedback register structure) is applied on the 187 bytes minus the sync byte. The resulting byte is inserted in TS instead of the 0x47 sync byte. This allows for simultaneous packet and error detection at the decoder. The Reed Solomon encoder (outer block coding) now inputs and encodes this data stream, using a block code of the type (block bytes, block bytes-parity bytes) over a Galois Field ($2^{\text{bits_in_byte}}$). The convolutional interleaver used in QAM basically shifts the RS coded data symbols sequentially into a bank of 128 registers (size in bits of each register equals 1 RS coded data symbol) where every successive register has J symbols more storage than the previous register, thus providing a (I,J) interleaving depth. The interleaved data bits are then randomized using a randomizing polynomial $f(x)$, usually implemented as a linear feedback shift register (standard dependant). The randomized data is then trellis encoded and mapped into QAM I/Q symbols. The trellis encoder employs differential encoding along with convolutional encoding for robust inner coding. Finally the QAM symbols are shaped using a nyquist root raised cosine filter, modulated using identical 90° degrees phase shifted carriers, summed and RF modulated in the GHz range. This signal is then sent over a typical coaxial cable.





A typical QAM de-modulating scheme is shown below,

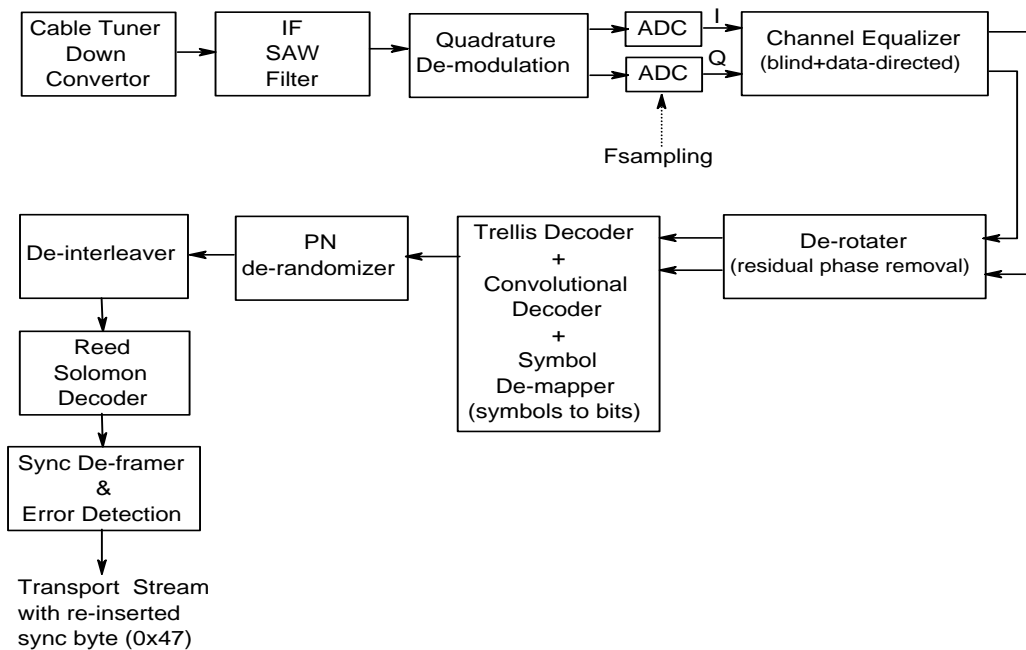
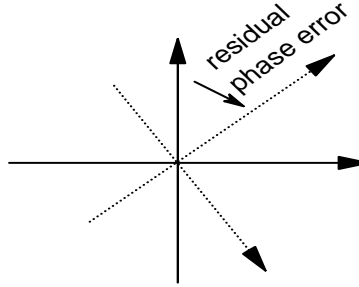


Figure 47. QAM de-modulation scheme for reception of MPEG2 packetized transport data over a coaxial cable

The QAM demodulator accepts a RF modulated signal and down-converts it to IF. An IF SAW filter (BW around 36 to 57 MHz) band passes the signal. It is then quadrature demodulated (analog domain), digitized and then sent to a complex (I/Q) channel equalizer. The channel equalizer initially uses a blind equalization scheme (no training reference signal provided in the transmission) to open the eyes of the I/Q eye diagram wide enough for slicing. The equalization is then changed to data-directed mode after enough symbol averaging. The sliced

symbols are then sent to the de-rotator which removes the residual phase error in I/Q symbols.



The de-rotated symbols then go to the trellis decoder and convolutional decoder combination which soft decodes the symbols (removes the soft redundancy introduced depending on the puncture rate) and then a de-mapper de-maps the QAM symbols to bits. The bitstream is then de-randomized using the identical randomizing polynomial used at the transmitter, de-interleaved (depending on the I, J programmed length) and sent to the reed solomon decoder which converts the bitstream into blocks and removes byte errors depending on the added RS parity bytes. For cable the RS code is usually RS(128,122). So a total of $6/2 = 3$ byte errors can be corrected in the 128 byte block. The resulting bitstream is then framed into 188 bytes, the parity checksum byte is removed, a parity check is run on the 187 bytes and compared with the parity checksum byte. After this error check (good or bad frame), the 0x47 sync byte is re-attached to the 187 bytes and the 188 byte transport stream is recreated.

Cable channel usable bitrate = QAM symbols/sec * bits/symbol * Puncture Rate * RS factor

5 Msymbols/sec, 64 QAM signal with PR of 14/15 and RS code (128,122) provides a effective transport bitrate of = $5 * 6 * 14/15 * 122/128 = \sim 26$ mbits/sec

Some of the subtle differences between QAM and QPSK are,

- Additional sync packet removal and parity checksum on transport packets
- Inner coding (differential + convolutional) is different than the pure convolutional coder used in QPSK. QAM systems employ a differential pre-coder which performs 90^{deg} rotationally invariant trellis coding before the convolutional encoder. This allows the demodulator to recover very quickly from carrier phase slips because the data information bits are carried by a change in phase.
- Channel equalization is quite complex. Most QPSK systems skip the channel equalizer because the eyes (eye diagram) are wide enough for correct slicing.
- Programmable interleaving depth depending on QAM size used.

Salient points of QAM transmission,

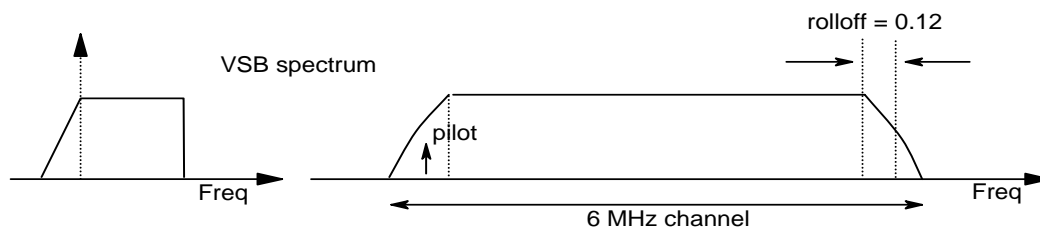
- Low bandwidth requirements due to cable channel (need to pack more bits per symbol – 256 QAM provides twice the bitrate than 64 QAM).
- SNR requirement is quite high.
- Channel has long impulse response (echoes or ghosting significant).

Where used,

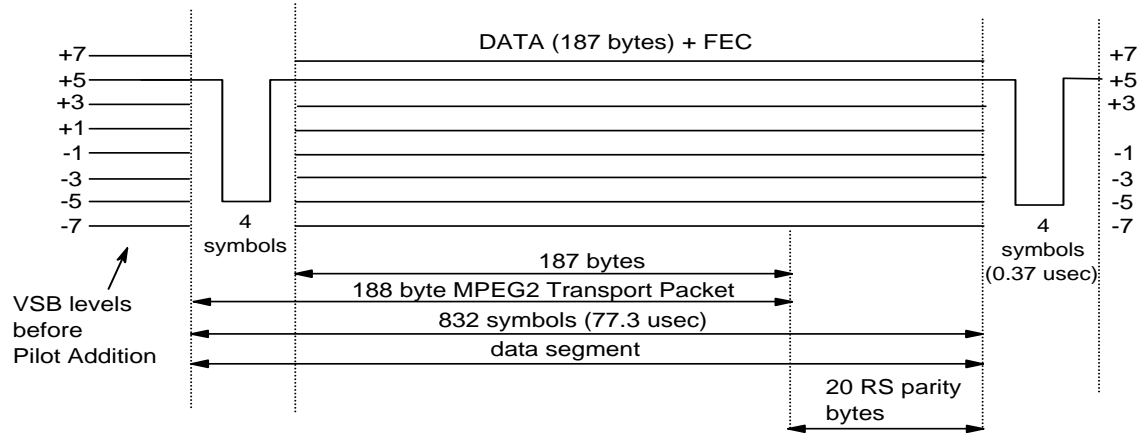
- Digital Cable TV downstream transmission all over the world – depending on the country and region, some of the transmission parameters like QAM size, interleaving depth, forward error correction technique may vary.
- Digital Cable modem downstream channels (DOCSIS in the US, typically 64 and 256 QAM).
- Digital Cable modem upstream return channels (DOCSIS in the US – typically 16 QAM or QPSK).

VSB (Vestigial Side Band Modulation)

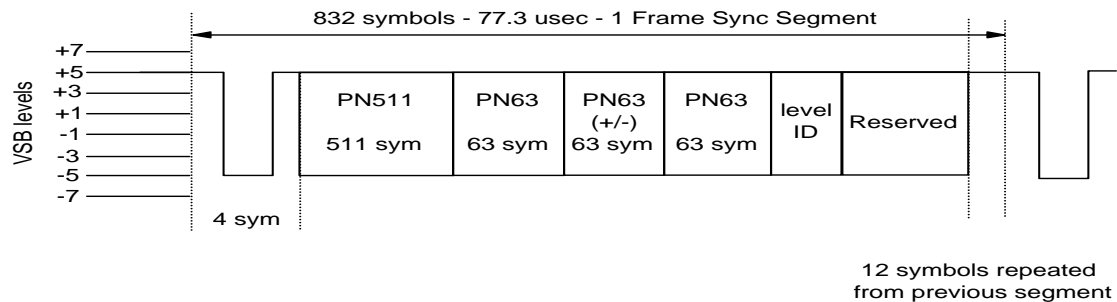
Vestigial side band modulation is only used in the ATSC transmission standard in the USA to transmit MPEG2 packetized standard definition and high definition signals in 6 MHz channels over the terrestrial media along with the analog NTSC's 6 MHz signal channels. Although there exists 2 modulation schemes, 8VSB for over the air (terrestrial) transmission and 16VSB for cable transmission, only 8VSB is being used.



The 8 VSB trellis coded digital signal is used for transmission and has 8 discrete data levels. The data signal is broken down into 832 symbol segments, including a 4 symbol, binary data segment sync signal. The symbol rate is 10.762 Msymbols/sec. The transmission segment's data part is the MPEG2 188 byte transport packet. A low level pilot is created by adding a DC value to the baseband signal. After modulation this DC value creates an in-phase pilot to be added to the total data spectrum. This becomes very useful in carrier recovery at the receiver independent of the data and provides reliable carrier recovery down to SNR of 0 dB. The repetitive 4 symbol data segment syncs not only delineate the data segment, but provides symbol clock recovery independent of the data. These replace the 0x47 MPEG syncs which are re-inserted into the data stream at the receiver.

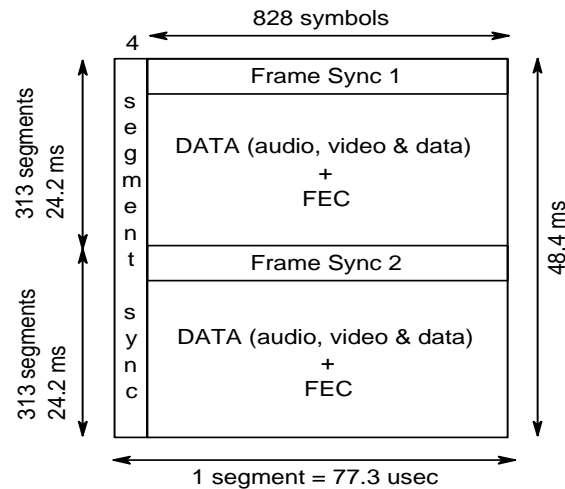


Trellis Coded 8VSB Baseband Data Segment Format



12 symbols repeated from previous segment

Framing (Frame) Sync Format (2 per frame) for 8 or 16 VSB



Transmission Format for (8 or 16 VSB) Frame

A frame sync is inserted every 313 segments, to facilitate training mode equalization at the receiver. It consists of 1 long 511 symbol PN sequence (mostly for terrestrial media – long ghosts) which is very useful for tracking precursor and long post-cursor ghosts. It also consists of 3 short 63 symbol PN sequences (2nd PN sequence in this group of 3 is inverted between frame sync 1

and frame sync 2) which are useful in cable media to cancel short ghosts which are characteristic of cable transmission systems.

A typical 8VSB transmission scheme is shown below,

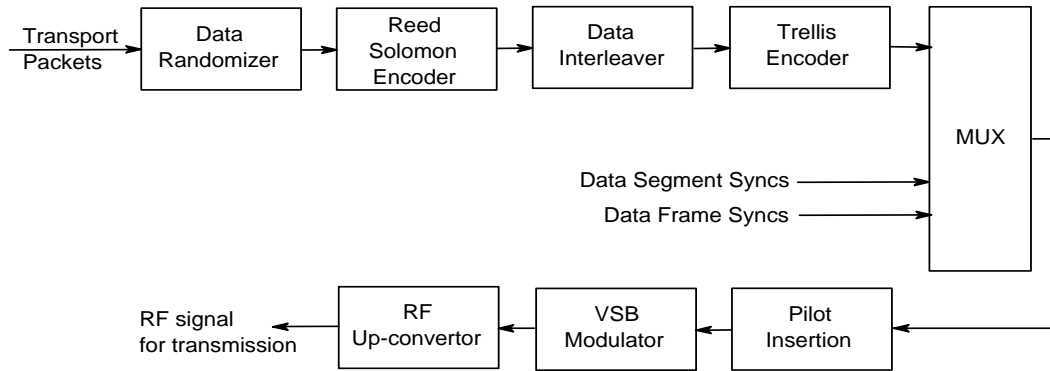


Figure 48. 8 VSB modulation scheme for transmission of MPEG2 packetized transport data over the terrestrial media

The transport packets are first randomized by a randomizing polynomial $f(x)$ which is again a LFSR implementation such that its spectrum is flat across all frequencies when sent over the terrestrial channel. The implementation of this randomizer becomes important because 8VSB today is a simulcast system (occupies spectrum with analog NTSC) and minimizing interference with NTSC becomes very important. The Reed Solomon encoder encodes the randomized data block by block by adding 20 parity (redundant error coding) bytes to 188 byte blocks using a Galois Field ($2^{\text{bits_in_byte}}$) and code RS (208,188). The resulting RS symbols are convolutionally interleaved over 52 data segments (or $1/6$ frame = 4msec) to provide for protection against channel burst errors. The segment and frame syncs are not-interleaved. This data stream is then trellis encoded (pre-coded) in 8VSB, adding additional redundancy to the data creating the multilevel 8VSB data symbols. A convolutional encoder adds a $2/3$ puncture rate to the data. The syncs (data segment and data frame) are multiplexed into these symbols and a DC offset is added (to create the in-phase pilot). A VSB modulator creates a root raised cosine shaped (filtered) IF signal (at 44 MHz) with most of one sideband removed (1 sideband + vestige of 2nd sideband). This IF signal is then RF up-converted for transmission over the terrestrial media.

A typical 8 VSB reception scheme is shown in figure 49,

The most important characteristics of a VSB transmission/reception system is that the received signal does not need to be sampled at 2 times the 8VSB symbol rate of 10.762 Msymbols/sec, because all the sync and the timing recovery is independent of the slicing of data. Basically the syncs (data segment and data frame syncs) can be recovered by correlation methods and hence all

the data processing in a VSB receiver is done at the sample rate (T), allowing for a simpler digital signal processing chain (ADC's, equalizers, etc.).

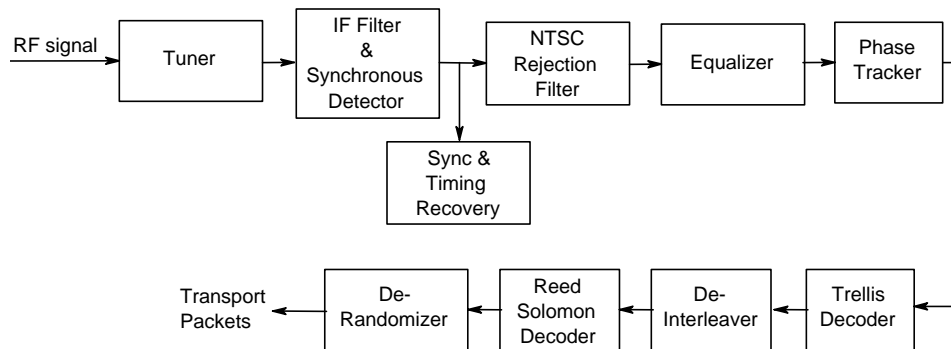
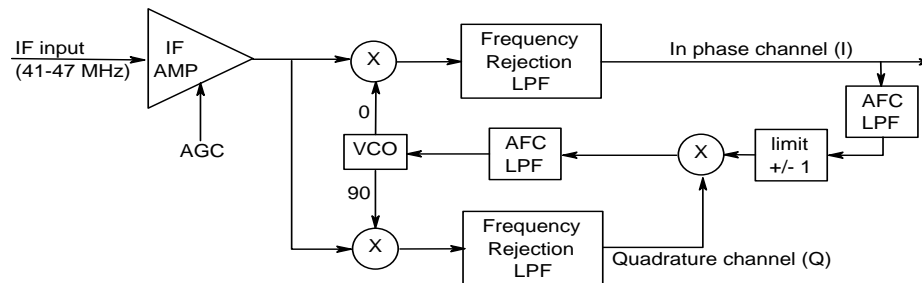
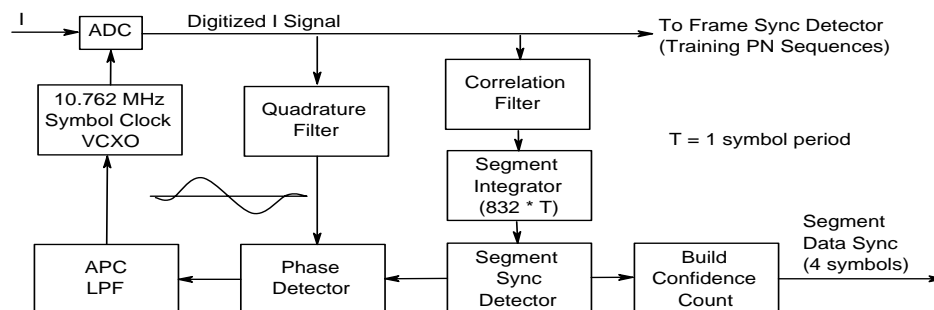


Figure 49. 8 VSB de-modulation scheme for reception of MPEG2 packetized transport data over the terrestrial media

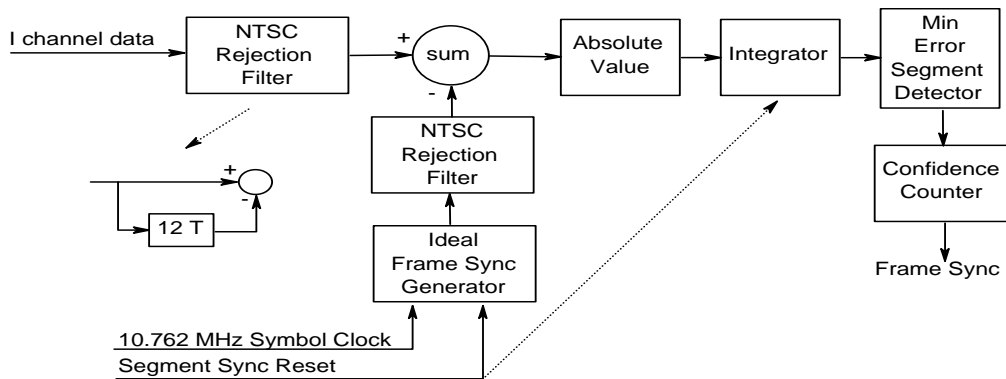
The RF signal is demodulated in the tuner and sent to an IF filter and synchronous detector which is basically a Frequency and Phase Locked Loop (FPLL) for wideband frequency acquisition and narrowband phase tracking. The presence of a pilot carrier allows the VSB receiver to recover the carrier in the IF signal by frequency locking to the incoming signal immediately.



Using both the I & Q channels the FPLL automatic frequency control low pass filter (AFC LPF) acts on the beat signal resulting from the frequency difference between the local oscillator frequency and the incoming pilot carrier frequency and limits it to within ± 1 . This ± 1 DC signal is used to control the VCO to reduce the frequency difference till frequency and phase lock occurs. The in-phase I channel is then sent to the segment and clock recovery loop.

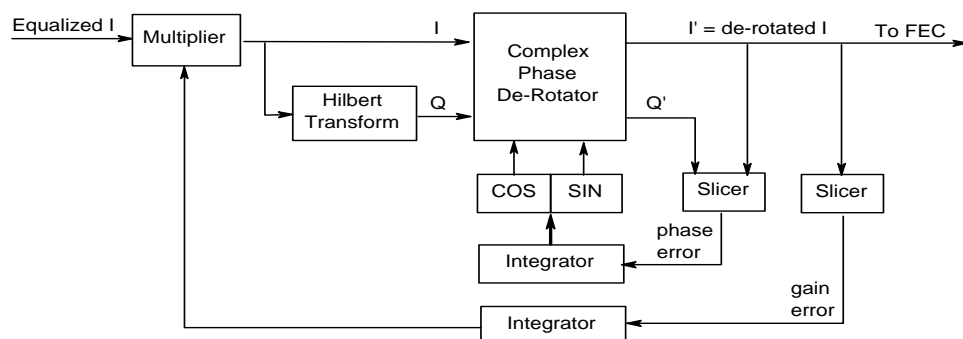


By using a correlation filter and a narrowband PLL tracking filter, the segment sync (4 symbol period) is found and the symbol clock is locked to this sync. A VCXO running at the symbol clock period (T) digitizes the I signal and sends it to a correlation filter (4 symbol correlator) which looks for binary syncs at the segment repetition rate (12.937 KHz). The output of this correlator is integrated over several segments and a confidence count of sync presence is built. Also a 4 tap quadrature filter converts the segment syncs into a “discriminator S curve” which is sampled during sync time by a phase detector to provide an error voltage proportional to the phase difference between the receiver’s sampling clock and zero crossing of the quadrature filter output to adjust the VCXO. Below we have shown the frame sync detector,

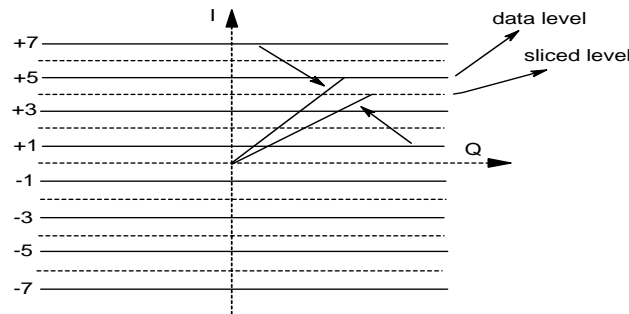


Segment frame sync detection is accomplished by first filtering the I channel using the NTSC rejection filter which removes the 3 main NTSC components (Aural, Visual & chrominance) and then subtracting the data segments from the ideal frame sync. The absolute value is then integrated symbol by symbol for the entire length of each segment. After one frame (313 segments) the data segment with the lowest error is taken as the received frame sync. Also the polarity difference in the middle PN63 sequence is used to determine between frame sync 1 and frame sync 2.

Once both the data segment sync and data frame syncs are found the received I signal is sent to the channel equalizer which uses the PN511 (inserted in the data frame syncs) to detect and remove any ghosting or ISI present in the 8VSB signal. This is shown in figure 43. The last step before the FEC is the phase tracking or de-rotation circuit as shown below,



The phase tracker is a data decision 1st order feedback loop which removes any phase noise in the I signal from the equalizer which may be left over from the carrier recovery process. Due to phase noise the entire VSB constellation could rotate around the origin. This phase tracker or de-rotator corrects this. The output of the equalizer (real I signal) is passed into a Hilbert Transformer which gives us the 90^{deg} phase shifted Q component. Both the I & Q signals are sent to a de-rotator which rotates the constellation by integrating (or accumulating) the phase error till it reaches 0.



This de-rotated signal is then passed into the trellis decoder which soft decodes the I symbols and maps them into a data bit stream. Then the de-interleaver de-interleaves the data segments and sends them to the reed solomon decoder which decodes the RS symbols and removes the additional parity bytes. Finally the signal is de-randomized using an identical polynomial used at the transmitter.

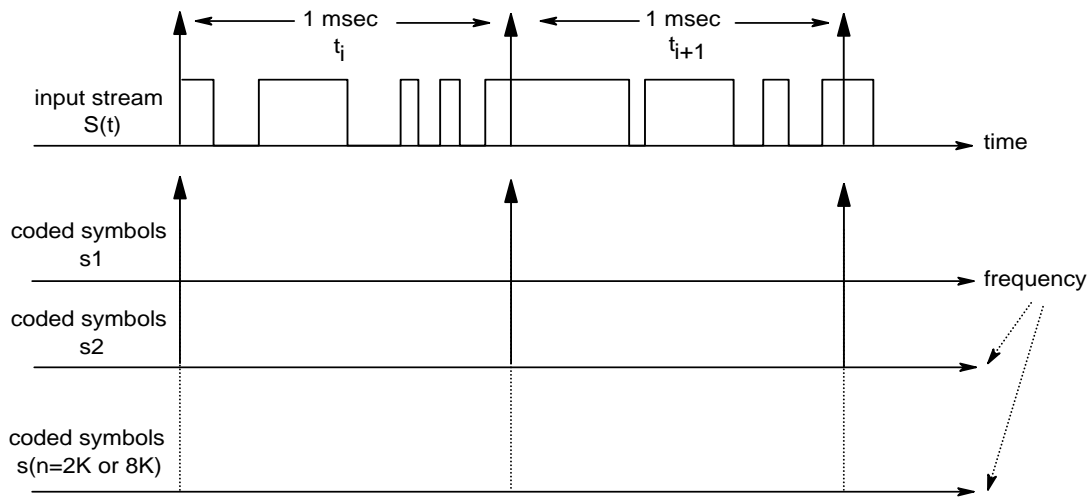
COFDM (Coded Orthogonal Frequency Division Multiplexing)

The Coded Orthogonal Frequency Division Multiplexing (COFDM) modulation scheme was adopted by the Digital Video Terrestrial Broadcasting (DVB-T) Group to transmit packetized MPEG2 standard and high definition A/V signals over the terrestrial medium in Europe. This transmission method is also widely used in Large Single Frequency Networks (SFN) and mobile transmission networks. Typically COFDM transmission systems have strong immunity to inter-symbol interference, long delay static and dynamic multi-path distortion prevalent in the terrestrial medium.

The ATSC 8-VSB system is more robust in an Additive White Gaussian Noise (AWGN) channel, has better spectrum efficiency, a lower peak-to-average power ratio, and is more robust to impulse and phase noise. It also has comparable performance to DVB-T on low level ghosts and analog TV interference into digital TV. Therefore, the ATSC 8-VSB system could be more advantageous for Multi-Frequency Network (MFN) implementation and for providing HDTV service within a 6 MHz channel.

In a COFDM transmission scheme the signal is divided and coded into symbols which last a maximum of 1ms and, are used to modulate multiple QAM carriers.

The multiple QAM carriers could be anything between 2000 (2K) to 8000 (8K) with a guard interval between them.



$$\text{Signal, } S(t) = \text{Summation}^{(2k \text{ or } 8k)} [a_i * s[\sin(w_i + Q_i)]]$$

This transformation is a time to frequency domain conversion of the signal (similar to the FFT).

If 64 QAM is used, then each carrier is modulated with a 6 bit symbol. So for each interval of 1msec, the 2K carriers can carry a total of $6 * 2K \text{ bits} = 12K\text{bits}$ and the 8K carriers can carry a total of $6 * 8K \text{ bits} = 48K\text{bits}$. Thus effective bitrate without the guard interval can be $12K\text{bits/sec} * 1\text{sec}/1\text{msec} = 12\text{Mbits/sec}$ for 2K carriers, and 48Mbits/sec for 8K carriers.

A typical COFDM transmission & reception scheme is shown in figure 50,

A strong inner code and a good channel estimation system are mandatory for a COFDM system to withstand 0 dB echoes. Turbo codes are used instead of convolutional codes at the receiver to improve performance.

In a COFDM modulator, the input transport stream is forward error corrected (using inner turbo coding instead of convolutional encoding) and randomized, then coded into symbols (with a 1ms length) and are used to modulate QAM carriers (either 2K for mobile transmission or 8K for terrestrial transmission). Then an IFFT with a programmable window size is taken over these modulated carriers with guard interval insertion. A synchronization frame is added and finally the signal is up-converted to RF for transmission.

In a COFDM de-modulator, the transport stream is RF down-converted to IF frequency, digitized and quadrature demodulated. The symbol clock is recovered by de-correlating the signal with the reference sync frame. This signal is then de-rotated to remove the residual phase error. An FFT algorithm performs the demodulation of the coded symbols as an array of N matched filters to the signal sent on each sub-carrier. A channel estimator (using blind equalization + data-directed equalization) removes ghosts/echoes, if present in the signal. Finally the

equalized signal is sent into the FEC which is a combination of X soft inner decoders (X depending on the turbo code used at encoder), a de-interleaver, reed solomon decoder and a data de-randomizer to output the transport stream.

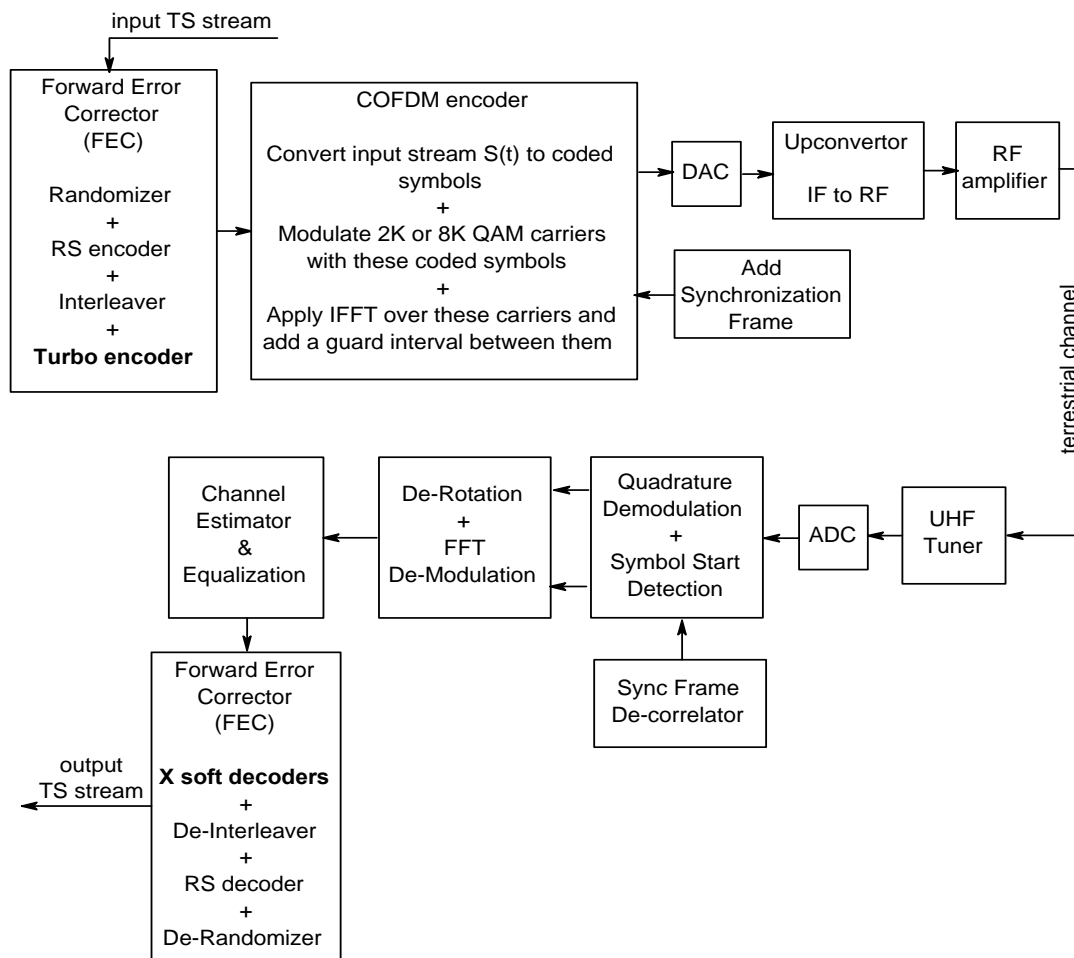


Figure 50. COFDM modulation/de-modulation scheme for transmission and reception of MPEG2 packetized transport data over the terrestrial media

COFDM is gaining popularity as the transmission system of choice over the terrestrial medium (other than the US) for the following reasons,

- Since up to 2K or 8K carriers (each carrier being a pure sine wave) are modulated with coded symbols with a guard interval insertion, the RF signal is less prone to noise and echoes
- Echo delays (or ghosts) less than the guard band interval between symbols do not affect the signal
- Doppler effect which is responsible for modifying the frequency of carriers has little effect because of the distance between each carrier.
- Single frequency networks (8K carriers) or multiple frequency networks (mobile networks - 2K carriers) are possible with COFDM

- Flexible in meeting various design requirements like bandwidth, power and spectrum shaping
- Requires no adaptation to instantaneous channel response.
- Less sensitive to shifts in sampling time
- Bandwidth efficiency approaches nyquist rate as FFT size increases
- A properly coded and interleaved COFDM system will exceed the BER performance of many practical systems, especially in wide band mobile reception and in the presence of strong and dynamic ghosts.

Two of the disadvantages of COFDM are higher transmitter power requirements and sensitivity to carrier frequency offsets and single tone interference.

Forward Error Correction (FEC) with Concatenated Codes

Forward Error Coding and decoding is employed in most transmission systems as shown previously for reducing the error rate to tolerable limits, since the probability of errors in any transmission channel (satellite, cable, terrestrial) is quite high. So a combination technique of inner coding, interleaving and outer coding is used to reduce the final error rate down to 10^{-4} to 10^{-11} which equates to less than 1 error per X hours during a typical transmission.

Inner coding could employ any of the following methods,

- **Convolutional encoder** – at the coding side, N output bits are transmitted from M (M being the constraint length) successive input bits. In QPSK, A & B patterns are computed at every instance (n) from each input sequence (x) of M bits. If $M = 7$ then,

$A(n) = x(n-6) + x(n-5) + x(n-4) + x(n-3) + x(n)...$, A modulates the I component

$B(n) = x(n-6) + x(n-3) + x(n-2) + x(n-1) + x(n)...$, B modulates the Q component

This gives us 1/2 puncture rate, basically 2 bits are outputted for 1 input bit. If the puncture rate is 7/8, then 8 bits are outputted for every 7 input bits. This involves dropping some of the computed A and B bits.

At the decoder side all possible AB sequences (corresponding to all possible sequences of input bits) are compared to the received bits and the closest match is chosen as per the **Viterbi** algorithm. Also since the puncture rate is unknown, all the possible puncture rates are tried till a reasonable error threshold is reached or a synchronization word is found.

- **Trellis pre-coder + Convolutional encoder** - in this type of coding the input bits are formed into trellis groups and a differential pre-coder performs a 90° rotationally invariant trellis coding on these groups. The differential encoder allows the information to be carried by the change in phase rather than absolute phase. This allows the receiver to recover very

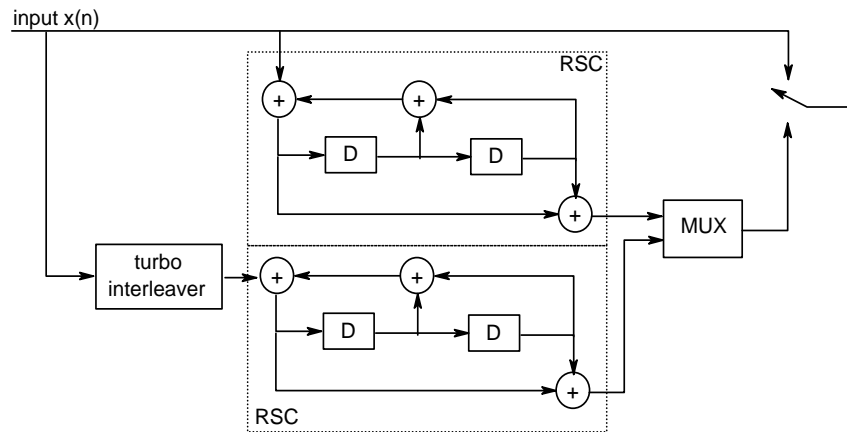
fast from carrier phase slips without re-synchronization of the FEC when carrier phase tracking changes quadrants, thus preventing error bursts at the output of the FEC.

$$A(n) = x(n) + A(n-1) + y(n) * [A(n-1) + B(n-1)]$$

$$B(n) = y(n) + x(n) + B(n-1) + y(n) * [A(n-1) + B(n-1)]$$

These differentially encoded A's & B's are then fed into a (constraint length M) convolutional encoder which introduces the redundancy ratio (or puncture rate) into the trellis groups.

- **Turbo coder** – in the turbo coder, the convolutional encoder is replaced with two or x (shown here as 2) recursive systematic identical convolutional (RSC) encoders that are fed the same input data in parallel, but in a different order. A RSC encoder can be derived from a conventional convolutional encoder by feeding back one of its outputs. The interleaver separating two RSC encoders changes the order of the input going to the next RSC encoder and is called the turbo interleaver.



The turbo decoder in the receiver uses the concept of iterative decoding. It basically consists of x (or 2) soft input decoders depending on x (or 2) RSC encoders used. Decoding begins with the 1st soft decoder deriving probability estimates based on the received symbols. These estimates, typically in the form of log-likelihood ratios (LLR), are then passed to the 2nd decoder which uses it as priori information. The 2nd decoder then uses this side information in conjunction with its received symbols to produce its own log-likelihood ratios. The LLR from the 2nd decoder is passed to the 1st decoder to be used as side information during the next iteration and this process follows iteration by iteration. The algorithm used to calculate the probability ratios (or least cost path) could be the viterbi algorithm.

Turbo codes typically improve the bit-error-rate (BER) of most transmission systems.

Interleaving between the inner and outer coding is used to improve the burst performance of the concatenated coding scheme. When the inner coding makes

a wrong decision, it generally produces a burst of errors. Rather than having that error burst in the same reed solomon (RS) block, a interleaver is used to spread them across several RS blocks. The principle is to write a rectangular table horizontally and then read it out vertically, either on a block by block basis (block interleaver) or in continuous time (convolutive interleaver).

Outer coders employ block coders like **Reed Solomon Codes**, which add redundancy words (bytes) to a block of information words (bytes). RS codes typically are of the type RS(M, M-2t), where M is the total message block length in bytes and 2t are the added redundancy bytes, which can correct up to t errors in the received message block M to generate the M-2t information words.

A message of M words having (n) bits per word is defined as a polynomial of degree M-1 on Galois Field, GF(2ⁿ).

$$a(x) = a_{M-1}x^{M-1} + a_{M-2}x^{M-2} + \dots + a_2x^2 + a_1x + a_0$$

A code generator polynomial g(x) is defined on Galois Field, GF(2ⁿ) such that,

- **g(x) is of degree 2t**
- **g(x) has 2t roots which are successive powers of alpha**
- **g(x) is normalized, g^{2t} = 1**
- **g(x) is of the form = (x-alpha^b)(x-alpha^{b+1})(.....)(x-alpha^{b+2t-1})**

The message polynomial a(x), is an RS code if,

- **a(x) is of degree M=2ⁿ-1**
- **a(x) is a multiple of g(x)**

Then $a(x) = g(x) * v(x)$

So now,

- **transmitted message is a(x)**
- **error message is e(x)**
- **received message is v(x) = a(x) + e(x)**
- **syndrome_i = v(beta_i), where beta_i are 2t roots of code polynomial g(x)**

Knowing each syndrome (total 2t values), and assuming < t errors, we solve a system of 2t equations of degree M with 2t unknowns to find,

- **t error values**
- **t error positions**

Thus RS codes can correct up to **t errors** with **2t redundancy bytes**.

Notes:

1. **New modulation schemes like 8PSK combined with Turbo Coding are being used to improve the bit-rates offered by current modulation schemes. Some of them also offer the same C/N ratios at improved bit-rates.**

AUDIO COMPRESSION TECHNIQUES

Audio compression is the technique of digitizing and compressing (or reducing the raw linear PCM data size) of 2 or more separate audio channels from analog audio sources (microphone, concert halls, recording studios, etc.) for distribution over any media like a hard disk, CD, DVD, digital TV, etc. There are 2 **DCT** (discrete cosine transform) based compression formats widely used,

- **MPEG audio** has 4 layers with multi-channel (MC) extensions. Without MC extensions, MPEG audio has 2 left and right stereo channels. With MC extensions MPEG audio can have up to 6 channels (2 left and right channels, 2 left and right surround channels, 1 center channel and 1 low frequency channel).
- **Dolby AC3 audio** has 6 or 5.1 channels (5 – left, right, left surround, right surround, center and a subwoofer/low-frequency channel).

MPEG Audio

MPEG audio can be split as follows,

- **MPEG1 audio (ISO/IEC 11172-3-audio standard)** having layers 1, 2 and 3 which support 2 channel stereo audio with or without a MPEG2 multi-channel extension in its ancillary data field.
- **MPEG2 audio (ISO/IEC 13818-3-audio standard)** which supports multi-channel extensions for all layers (1, 2, 3) with up to 5 channels and 1 LFE channel.
- **MPEG2 Layer 4 audio (ISO/IEC 13818-7-audio standard)** which supports multi-channel audio (5.1 full bandwidth audio channels) at bit-rates up to 320kbts/sec. This standard is also called **Advanced Audio Coding (AAC)**, and is the encoding method used by MPEG4 audio when encoding linear PCM audio channels between 24 to 64 kbts/sec.

The MPEG standard can be used to carry or store two or more high quality digital audio channels across any transmission or storage media. Some of the overall features of MPEG audio layers are,

- **Sampling rates are 32KHz, 44.1KHz & 48KHz**
- **Half sampling rates are 16KHz, 22.05KHz & 24KHz for low (~64kbts/sec) bit rates**
- **Quantization can be up to 24 bits**
- **Overall bit rate can be as high as 1.002Mbts/sec @48KHz**

Some of the features of the MPEG audio encoder/decoder are,

Coding Modes,

- Audio channels could be 3/2, 3/1, 3/0, 2/2, 2/1, 2/0, 1/0, where 3/2 is (L, R, C / Ls, Rs)
- A second stereo program
- Up to 7 additional multi-lingual or commentary channels

Subband Filter Transforms,

- Number of subbands are 32
- Sampling Frequency is $F_s/32$
- Bandwidth of subbands is $F_s/64$

Additional Decomposition by MDCT (modified DCT) for layer 3 only,

- Frequency resolution can be 6 or 18 components per subband

LFE Channel Filter Transform,

- Number of LFE channels are 1
- Sampling Frequency is $F_s/96$
- Bandwidth of LFE channel is 125 Hz

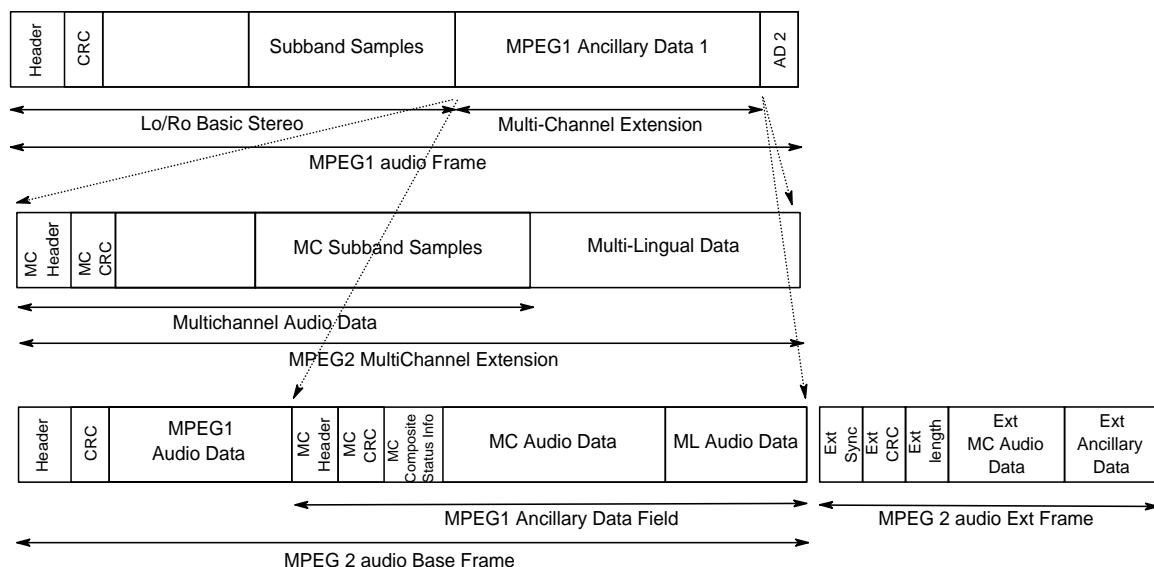
Dynamic Range,

- Greater than 20 bits

Due to the widespread use of MPEG1 two channel stereo audio, if MPEG2 multi-channel audio extensions are transmitted, the 2 down-mixed Lo/Ro stereo channels are also transmitted, such that any MPEG1 only audio decoder can decode only these channels and be backward compatible with the transmission.

$$L_o = L + \frac{1}{2} \sqrt{2} * C + \frac{1}{2} \sqrt{2} * L_s$$

$$R_o = R + \frac{1}{2} \sqrt{2} * C + \frac{1}{2} \sqrt{2} * R_s$$



A typical **audio encoding structure** (which is a **psychoacoustic algorithm**) is shown in figure 51. PCM audio samples are input into the encoder. The mapping creates a filtered and subsampled representation of the input audio stream. The mapped samples can be subband samples (layers 1 and 2) or transformed subband samples (layer 3). A psychoacoustic model creates a set of data to control the quantizer and coding. The quantization and coding block creates a set of coding symbols from the mapped input samples. The frame packing block assembles the actual bitstream from the output data of the other blocks and then adds other information (e.g. error correction) as necessary.

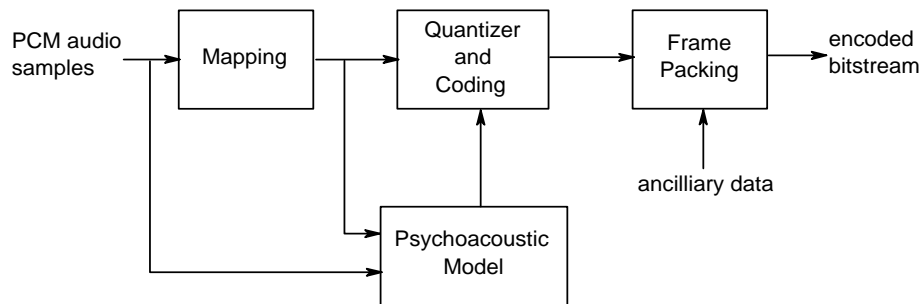


Figure 51. Typical MPEG audio encoder

Depending on the application, different **layers** of the audio coding system with increasing encoder complexity and performance can be used.

- **Layer 1**, contains the basic mapping of the digital audio input into 32 subbands, fixed segmentation to format the data into blocks, a psychoacoustic model to determine the adaptive bit allocation, and quantization using block companding and formatting.
- **Layer 2**, provides additional coding of bit allocation, scalefactors and samples. Different framing is used.
- **Layer 3**, provides increased frequency resolution based on a hybrid filterbank. It adds a different (non-uniform) quantizer, adaptive segmentation and entropy coding of the quantized values.

Layer 3 has the maximum **encoding delay**. Let us describe the 4 main blocks of the mpeg audio encoding system in some detail,

- The **filterbank** does a time to frequency mapping by splitting the audio signal with frequency F_s into several subbands. There are 2 filterbanks in the MPEG audio algorithm, a polyphase filterbank and a hybrid polyphase (MCDT) filterbank. Each provides a specific mapping in time and frequency. These filterbanks are critically sampled (i.e. there are as many samples in the analyzed domain as there are in the time domain). In layers 1 and 2, a filterbank with 32 subbands is used. In each subband, 12

or 36 samples are grouped for processing. In layer 3, the filterbank has a signal dependant resolution, where there are either 6x32 or 18x32 frequency bands.

- The **psychoacoustic model** calculates just a noticeable noise-level for each band in the filterbank. This noise level is used in the bit or noise allocation to determine the actual quantizers and quantizer levels. The psychoacoustic model most commonly used for all 3 layers is model 2 described in the MPEG ISO/IEC spec. The final output of this model is a signal-to-mask ratio (SMR) for each subband (layers 1 and 2) or group of subbands (layer 3).
- This **bit or noise allocator** looks at both the output samples from the subband filterbank and the SMR's from the psychoacoustic model, and adjusts the bit allocation (for layers 1 and 2) or noise allocation (for layer 3) in order to simultaneously meet both the bitrate and masking requirements. At low bitrates, these methods attempt to allocate bits such that it is psycho-acoustically inoffensive when they cannot meet the psychoacoustic demand at the required bitrate. For layers 1 and 2, this method is a bit-allocation process, i.e. a number of bits are assigned to each sample (or group of samples) in each subband. For layer 3, this method is a noise-allocation loop, where the quantizers are varied in an organized fashion, and the variable to be controlled is the injected noise. In both cases the output of this allocator is a set of quantization parameters and quantized output samples that are then sent to the bitstream formatter.
- The **bitstream formatter** varies from layer to layer. In layers 1 and 2, a fixed PCM code is used for each subband sample, with the exception that in layer 2 quantized samples may be grouped. In layer 3, Huffman codes are used to represent the quantized frequency samples. These Huffman codes are variable length codes that allow for more efficient bitstream representation of the quantized samples at the expense of added complexity.

The encoding algorithms provide a frequency response down to DC. When this is not required a **high pass filter** (cut off at 2-10 Hz) is applied to the PCM samples to avoid unnecessary high bit-rate requirements at the lowest sub-band. This generally improves overall audio quality.

MPEG 1 (all layers 1/2/3) audio frames without MC extensions contain 1152 (or 2 granules, corresponding to L/R channels) samples.

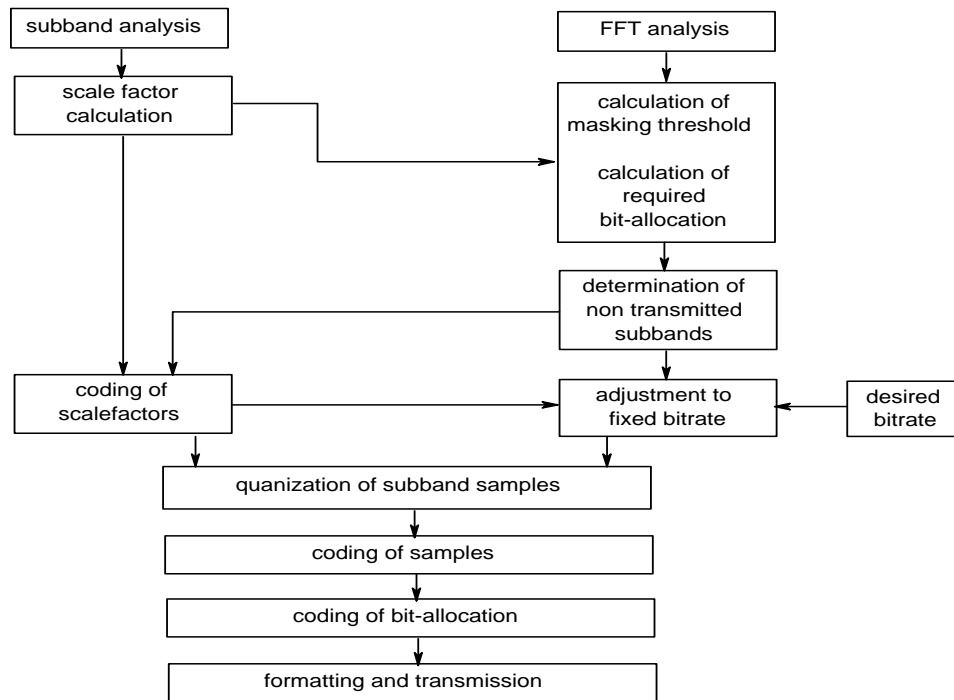


Figure 52. Some details of a MPEG audio encoder

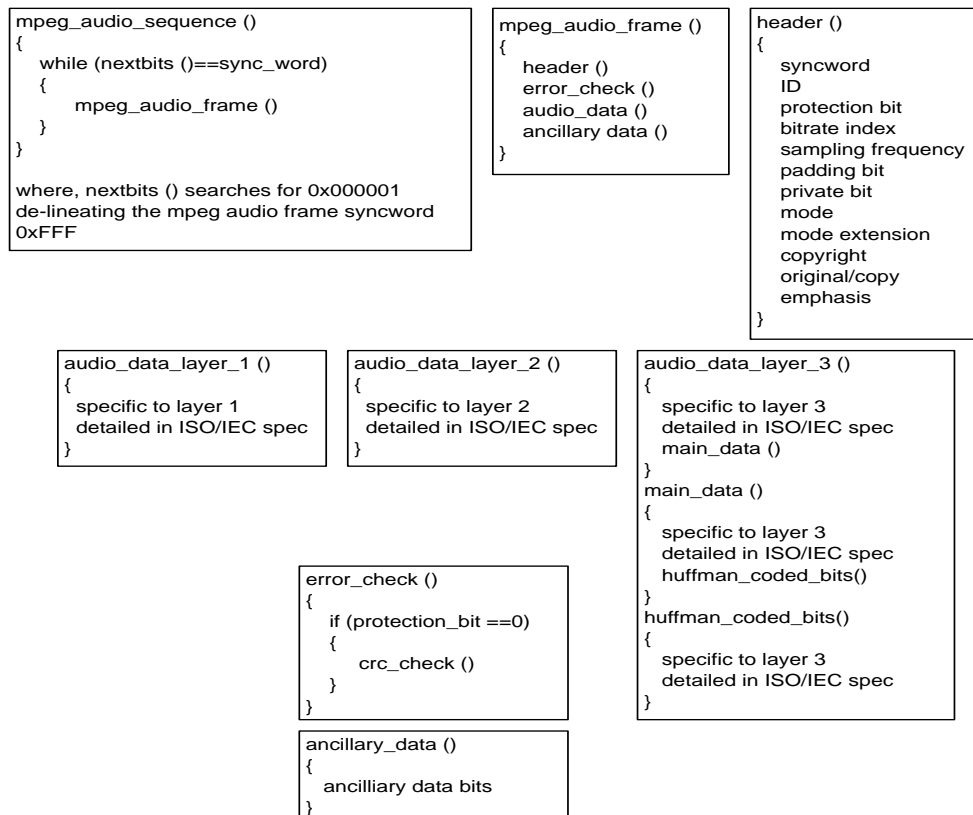
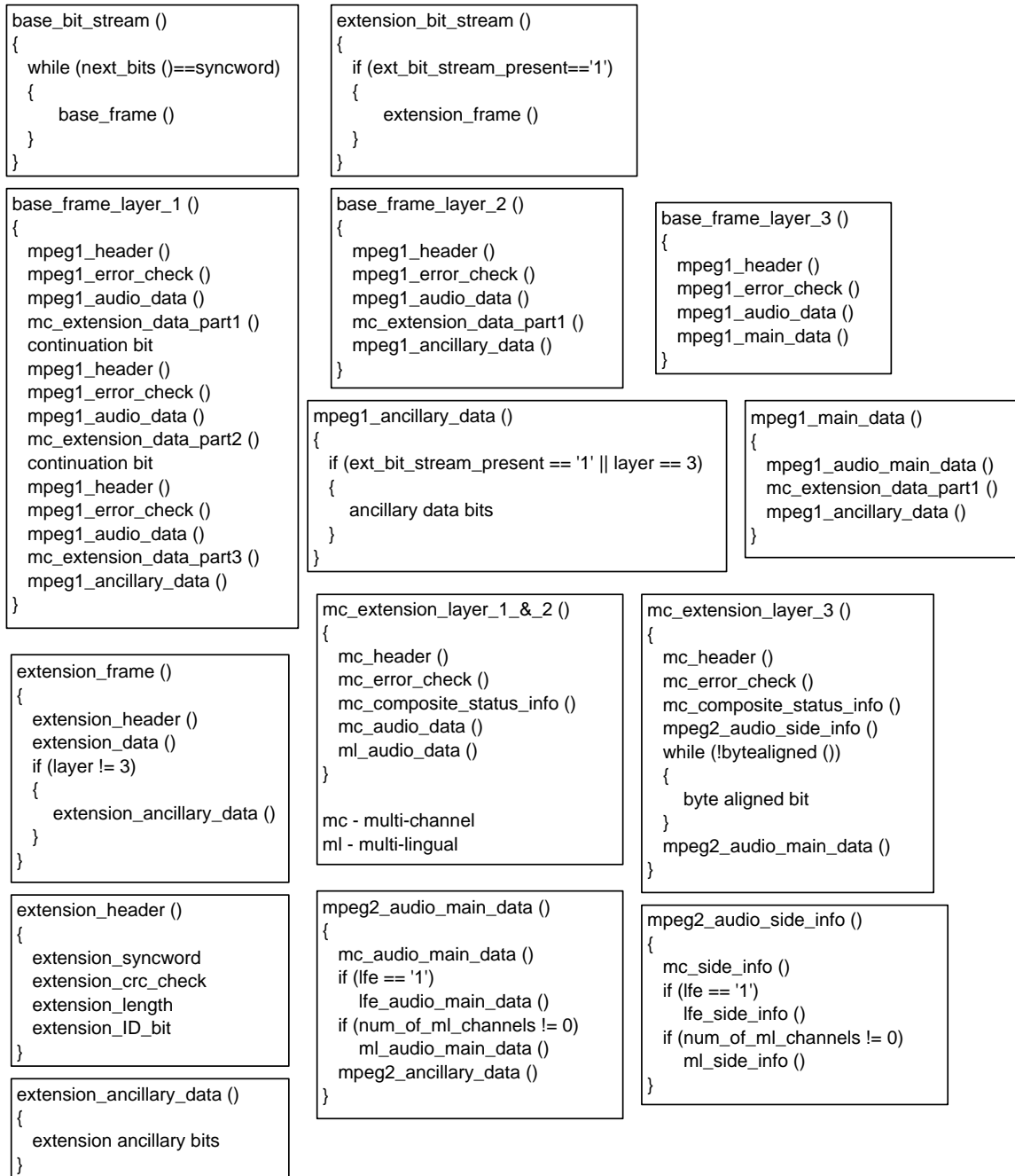
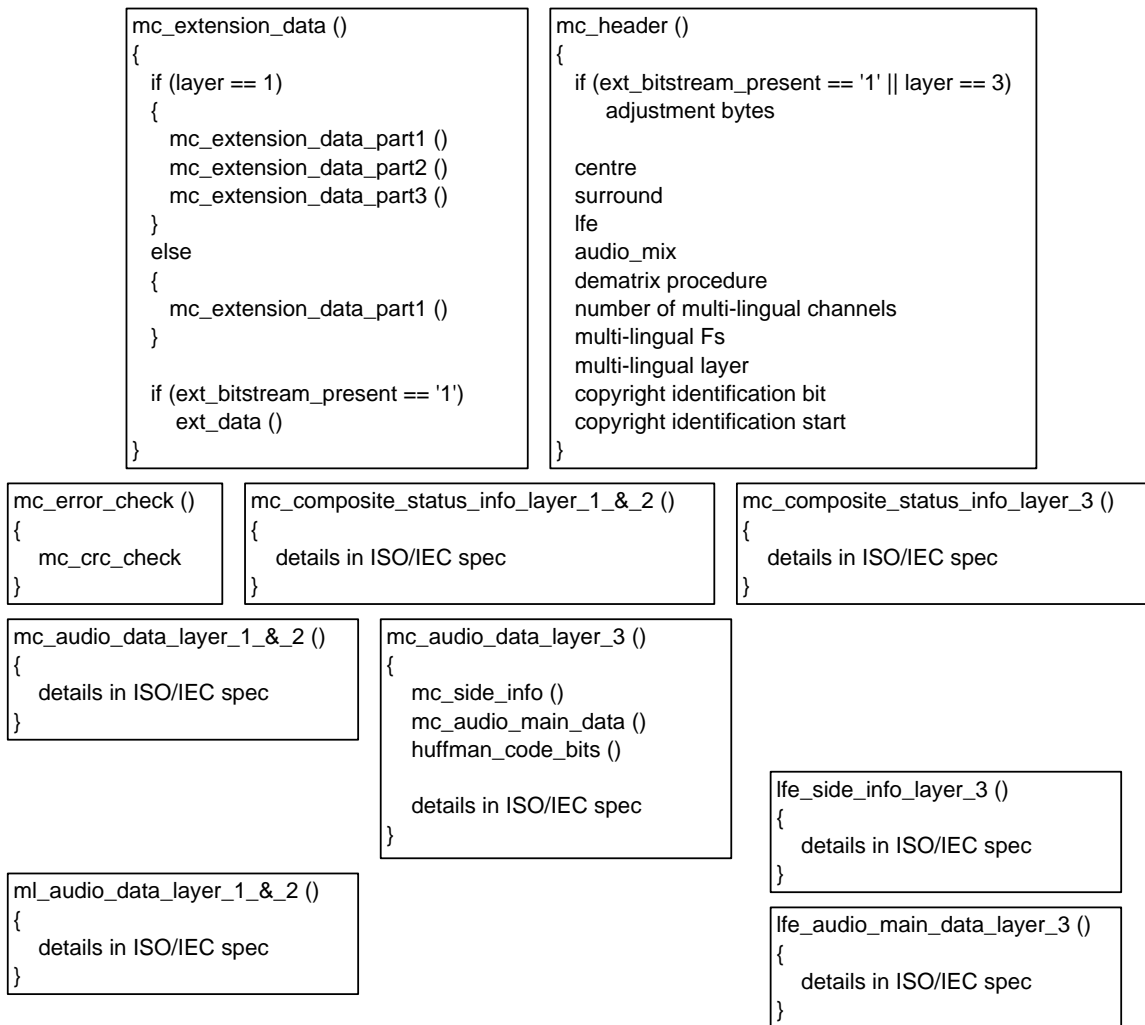


Figure 53. MPEG1 audio frame *without Multi-Channel* extensions



In layer 1, the contents of the mc_extension () are subdivided between mc_extension_data_part1 (), mc_extension_data_part2 () and mc_extension_data_part3 (), optionally followed by ext_data (), which is transmitted in the corresponding extension frame.

In layers 2 and 3, the contents of mc_extension () are subdivided between mc_extension_data_part1 (), optionally followed by ext_data () transmitted in the corresponding extension frame.



MPEG 2 audio frames with MC extensions contain a *base frame* (compatibility with MPEG1 only audio decoders) and an *extension frame* as shown.

MPEG 2 audio frames with MC extensions contain 1152 samples for each coded L/R channel, 12 samples for the LFE channel and either 1152 or 576 samples for each multilingual (ML) channel.

MPEG 2 (layer 3) audio frames with half sampling frequencies contain 576 (1 granule) samples.

Figure 54. MPEG 2 audio frame with *Multi-Channel* extensions

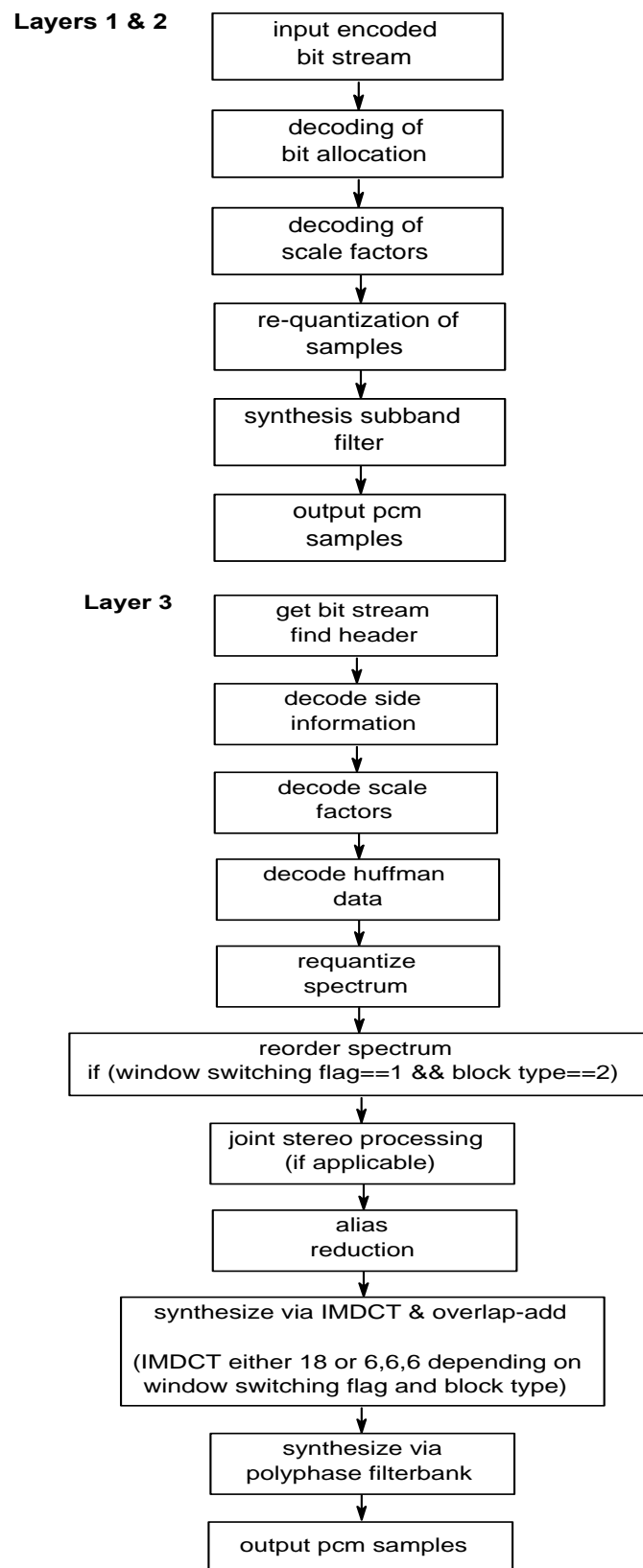


Figure 55. MPEG 1 audio decoding flow diagrams for layers 1/2 and 3

and scale-factors for a given block, bit-rate and output from the perceptual model is usually done by **two nested inner/outer iteration loops** in an analysis-by-synthesis method.

- a) **INNER Iteration Loop (rate loop)** - the Huffman code tables assign shorter code words to (more frequent) smaller quantized values. If the number of bits resulting from the coding operation exceeds the number of bits available to code a given block of data, this can be corrected by adjusting the global gain to result in a larger quantization step size, leading to smaller quantized values. This operation is repeated with different quantization step sizes until the resulting bit demand for Huffman coding is small enough. The loop is called rate loop because it modifies the overall coder **bit-rate** until it is small enough.
- b) **OUTER Iteration Loop (noise control or distortion loop)** – for shaping the quantization noise according to the masking threshold, scale-factors are applied to each scale-factor band. The system starts with a default factor of 1.0 for each band. If the quantization noise in a given band is found to exceed the masking threshold (allowed noise) as supplied by the perceptual model, the scale-factor for this band is adjusted to reduce the quantization noise. Since achieving a smaller quantization noise requires a larger number of quantization steps, and thus a higher bitrate, the rate adjustment loop has to be repeated every time new scale-factors are used. In other words, the **rate loop is nested within the noise control loop**. The outer (noise control) loop is executed until the actual noise (computed from the difference of the original spectral values minus the quantized spectral values) is below the masking threshold for every scale-factor band (i.e. critical band).

MPEG2 layer 4 or Advanced Audio Coding (AAC)

The AAC encoding method was invented by the MPEG committee to obtain high quality multi-channel audio having full bandwidth (~20 KHz) at data rates smaller (~320 Kbits/sec) than MPEG1 (with MC extensions) or MPEG2 multi-channel audio. This method is also widely used in MPEG4 audio coding when PCM channels having bitrates ranging from 24 to 64 Kbits/sec are to be compressed.

AAC has a total of 3 profiles which may use one or more of the **optional decoding tools** shown in the below table. Also these profiles may or may not be inter-operable with each other. AAC can encode PCM audio channels sampled from 8 KHz to 96 KHz.

Decoding Tool Name	Required/Optional
Bitstream formatter	Required
Noiseless Decoding	Required
Inverse Quantization	Required
Scale factors	Required
M/S decisions	Optional
Prediction	Optional
Intensity/Coupling	Optional
TNS (Temporal Noise Shaping)	Optional
Filterbank	Required
Gain Control	Optional

- **Main profile** – this is used when the memory (storage RAM) requirements are not stringent and when the processing power on the decoder implementation is substantially high. With the exception of the gain control tool, all parts of the tools may be used in order to provide the best compression ratios.
- **Low complexity profile** – this is used when there are significant limitations on storage RAM and processing power. In this profile prediction and gain control are not permitted and the TNS filter order is limited.
- **Scalable sampling rate profile** – in this profile the gain control loop is required. Prediction and coupling channels are not permitted. Gain control is not used in the lowest of the 4 PQF (polyphase quadrature filters) subbands. Also the TNS filter order and bandwidth is limited. This profile scales according to the audio bandwidth requirements.

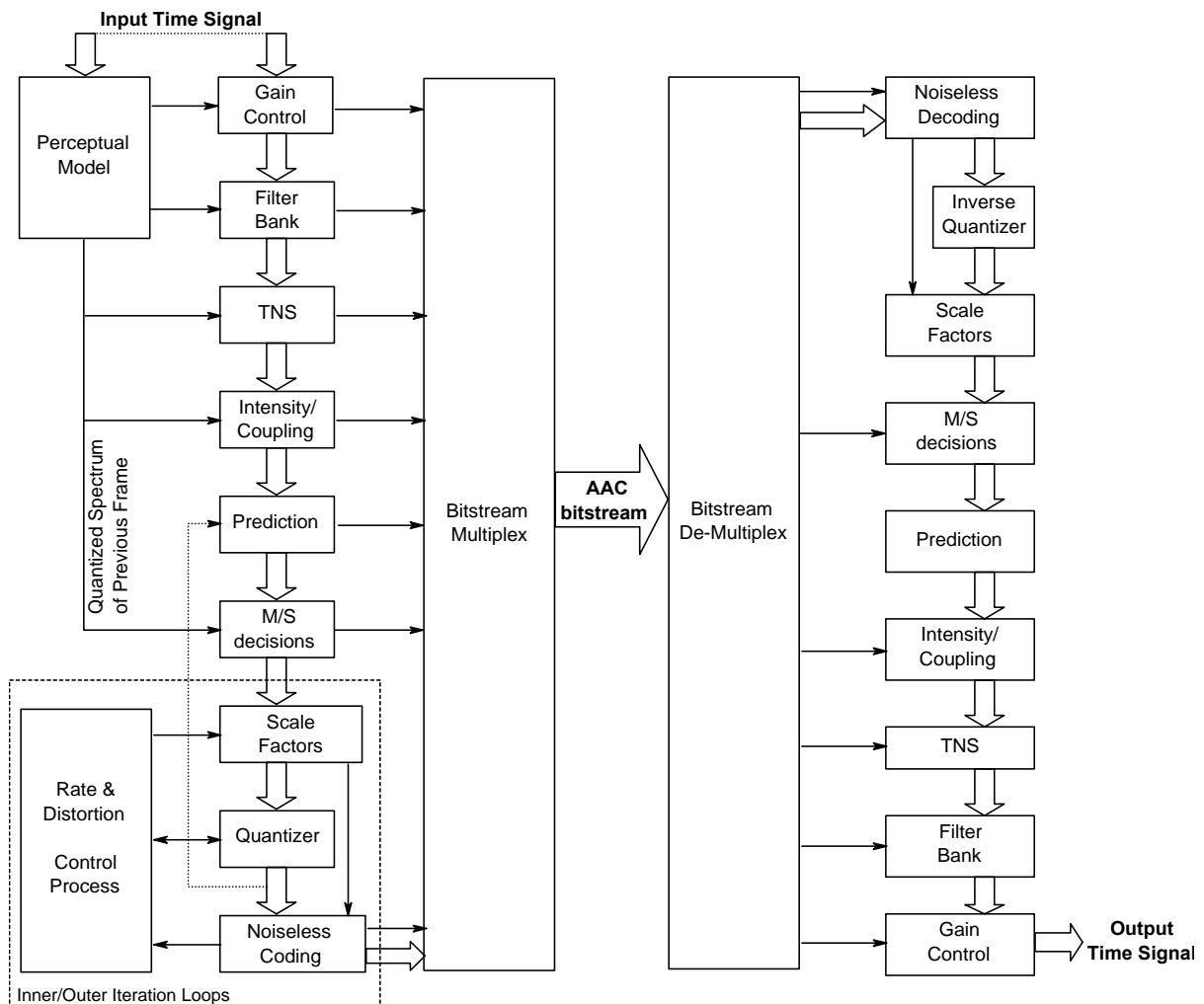


Figure 56. AAC audio encoding/decoding flow diagram

The data flow in figure 56 is from **left to right** and **top to bottom**. The function of the AAC decoder is to find the description of the quantized audio spectrum in the bitstream (by parsing the headers), decode the quantized values and reconstruction information, reconstruct the quantized spectrum, process the reconstructed spectrum through whatever tools are active in the bitstream to produce the actual signal spectrum as described by the input bitstream and finally convert the frequency domain spectrum into the time domain signal with or without a gain control tool. Following the initial reconstruction and scaling of the spectrum, there are many optional tools as shown in the table that can be used to modify one or more of the spectrum components in order to provide more efficient coding. For each of the optional tools a “pass through” option is provided in the encoding and decoding paths such that the spectrum can pass unchanged thus complying with the various ACC profiles.

Let us briefly discuss each stage in the **decoding path**,

1. The input to the **bitstream de-multiplexer** is the AAC bitstream. The de-multiplexer parses the AAC bitstream into parts and provides each tool with specific bitstream information. The outputs from the bitstream de-multiplexer are,
 - Sectioning information for the noiselessly coded spectrum
 - The noiselessly encoded spectrum
 - The M/S decision information (optional)
 - The predictor state information (optional)
 - The intensity stereo control information and coupling channel control information (optional)
 - The temporal noise shaping (TNS) information (optional)
 - The filterbank control information
 - The gain control information (optional)
2. The **noiseless decoding tool** takes information from the bit-stream demux, parses that information, decodes the Huffman coded data, and reconstructs the quantized spectrum and the Huffman and DPCM coded scale-factors. The inputs to this tool are,
 - The sectioning information for the noiselessly coded spectrum components
 - The noiselessly coded spectrum componentsIts outputs are,
 - The decoded integer representation of the scalefactors
 - The quantized values for the spectrum components
3. The **inverse quantizer tool** takes the quantized values for the spectra, and converts the integer values to the non-scaled reconstructed spectra. The quantizer is a non-uniform quantizer.
4. The **scale-factor tool** converts the integer representation of the scale-factors to the actual values, and multiplies the un-scaled inversely quantized spectra by the relevant scale-factors. Its output is the scaled inversely quantized spectra.
5. The **M/S tool** converts spectra pairs from Mid/Side channels to Left/Right channels under control of the M/S decision information in order to improve coding efficiency. The scaled

inversely quantized spectral components of individually coded channels are not processed by this block, instead these spectra are passed untouched through this block. If the M/S block is not active, then all spectral components (including channel pairs) are passed untouched through this block. The inputs to this block are,

- The M/S decision information
- The scaled inversely quantized spectral components related to channel pairs

Its output is, the scaled inversely quantized spectra related to channel pairs after M/S decoding

6. The **prediction tool** reverses the prediction process carried out at the encoder. This prediction process re-inserts the redundancy that was extracted by the prediction tool at the encoder, under control of the predictor state information. This tool is implemented as a second order backward adaptive predictor. If this block is not active then the scaled, inversely quantized spectra are passed through this block unmodified. Its inputs are,

- The predictor state information
- The scaled inversely quantized spectral components

Its output is, the scaled inversely quantized spectra after prediction is applied

7. The **intensity stereo / coupling tool** implements stereo decoding on pairs of spectra. In addition it adds the relevant data from a dependant switched coupling channel to the spectrum components as directed by the coupling control information. If this block is not active its inputs are passed un-modified through it. Also the intensity stereo tool and the M/S tool is arranged such that their operation is mutually exclusive on any given scale-factor band and a group of one spectral pair. Its inputs are,

- The inversely quantized spectrum components
- The intensity stereo control information

Its output is, the inversely quantized spectra after intensity and coupling channel decoding

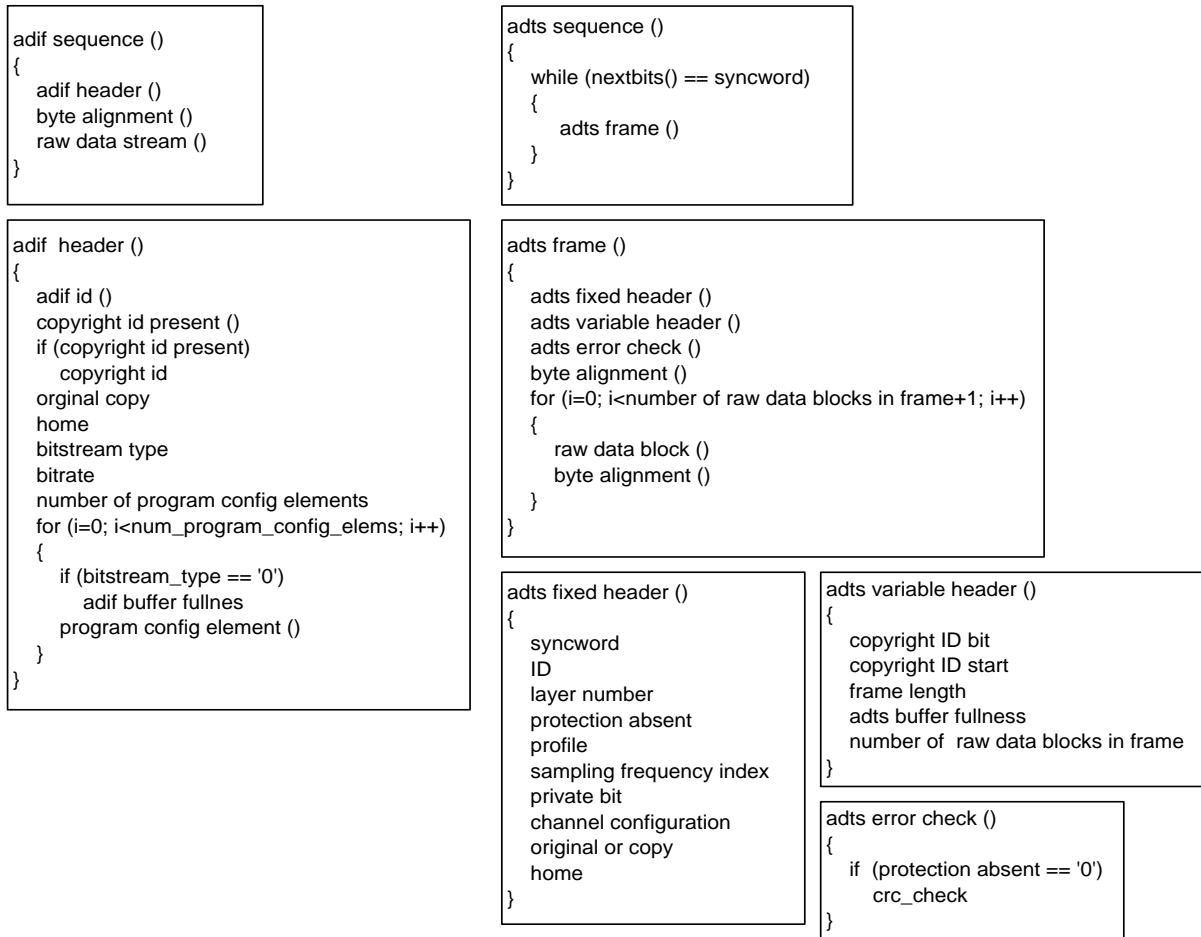
8. The **temporal noise shaping (TNS) tool** implements fine control in time over the coding noise. In the encoder the TNS process has flattened (to provide noise immunity using some form of nyquist filtering) the temporal envelope of the signal. In the decoder, the inverse process is to restore the actual temporal envelope under control of the TNS information. This is done by filtering (using a Nth order filter) some parts of the spectra. If this block is not active then the input to this block is passed unmodified through it.

9. The **filterbank tool** applies the inverse of the frequency mapping that was carried out in the encoder. An inverse modified discrete cosine transform (IMDCT) is used in the filterbank tool. If the gain control tool is not used, the IMDCT input consists of either 1024 or 128 spectral coefficients depending on the value of the window sequence. If the gain control tool is used the IMDCT is configured to use four sets of either 256 or 32 coefficients, depending on the window sequence. This tool outputs the **time domain audio signal**.

10. The **gain control tool** when active applies a separate time domain gain control to each of the 4 frequency bands that have been created by the gain control PQF (polyphase quadrature mirror filters) filterbank in the encoder.

The **AAC frame** can be arranged into 2 formats,

- **Audio Data Interchange Format (ADIF)**
- **Audio Data Transport Stream Frame (ADTS)** – this is also known as MPEG2 layer 4 audio.



The raw data stream can be decoded directly or put into the ADTS using a variable bitrate header and a buffer fullness measure, with all the data of one frame contained within 2 occurrences of the header

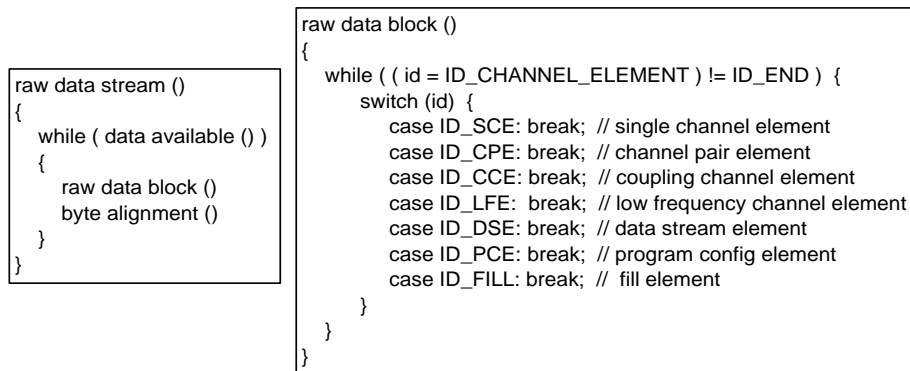


Figure 57. ADIF and ADTS encapsulated AAC frame

Dolby Digital (AC3) Audio

This multi-channel audio compression method (**ATSC Standard A/53**) is widely used today in almost all high-fidelity systems such as DVD (digital video or versatile disk), ATSC (audio channels accompanying high definition video) and in high end satellite/cable transmission/reception systems.

The AC-3 digital compression algorithm can encode a total of **5.1 PCM audio channels** at data rates ranging from **32 to 640 kbits/sec** at **32, 44.1 & 48 KHz** sampling frequencies. The 0.1 channel refers to a fractional bandwidth channel intended to convey only very low frequency (subwoofer or bass) signals.

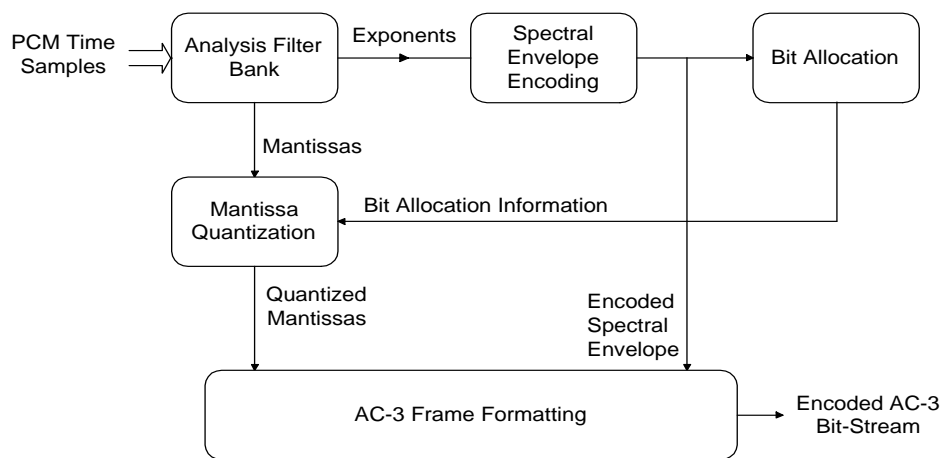
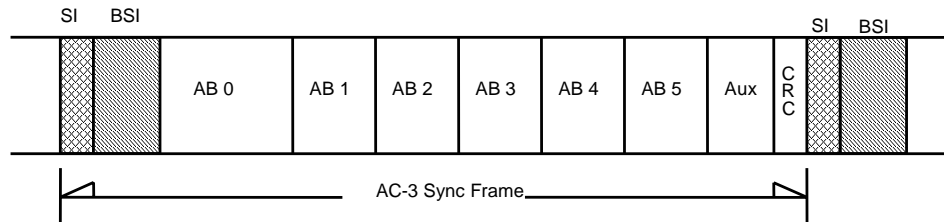


Figure 58. AC-3 audio encoding functional diagram

1. The first step in the AC-3 encoding process is to transform the representation of audio from a sequence of PCM time samples into a sequence of blocks of frequency coefficients. This is done in the analysis filter bank.
2. Overlapping blocks of 512 time samples are multiplied by a time window and transformed into the frequency domain. Due to the overlapping blocks, each PCM input sample is represented in two sequential transformed blocks.
3. The frequency domain representation is then be decimated by a factor of 2 so that each block contains 256 frequency coefficients.
4. The individual frequency coefficients are represented in binary exponential notation as a binary exponent and a mantissa.
5. The set of exponents is encoded into a coarse representation of the signal spectrum which is referred to as the spectral envelope.

6. This spectral envelope is used by the core bit allocation routine which determines how many bits to use to encode each individual mantissa.
7. The spectral envelope and the coarsely quantized mantissas for 6 audio blocks (1536 audio samples) are formatted into an AC-3 frame. The AC-3 bit stream is a sequence of AC-3 frames.



8. A frame header is attached which contains information (bit-rate, sample rate, number of encoded channels, etc.) required to synchronize to and decode the encoded bit stream.
9. Error detection codes are inserted in order to allow the decoder to verify that a received frame of data is error free.
10. The analysis filter-bank spectral resolution may be dynamically altered to better match the time/frequency characteristic of each audio block.
11. The spectral envelope can be encoded with variable time/frequency resolution.
12. A more complex bit allocation can be performed, and parameters of the core bit allocation routine modified so as to produce a more optimum bit allocation.
13. The channels may be coupled together at high frequencies in order to achieve higher coding gain for operation at lower bit-rates.
14. In the two-channel (Left/Right only) audio mode a re-matrix process may be selectively performed in order to provide additional coding gain, and to allow improved results to be obtained in the event that the two-channel audio signal is decoded with a matrix surround decoder.

An AC-3 coded audio bit stream is made up of a sequence of synchronization frames. Each synchronization frame contains 6 coded audio blocks (AB), each of which representing 256 new audio samples. A synchronization information (SI) header at the beginning of each frame contains information needed to acquire and maintain synchronization. A bit stream information (BSI) header follows SI, and contains parameters describing the coded audio service. The coded audio blocks may be followed by an auxiliary data (Aux) field. At the end of each frame

is an error check field that includes a CRC word for error detection. An additional optional CRC word is located in the SI header.

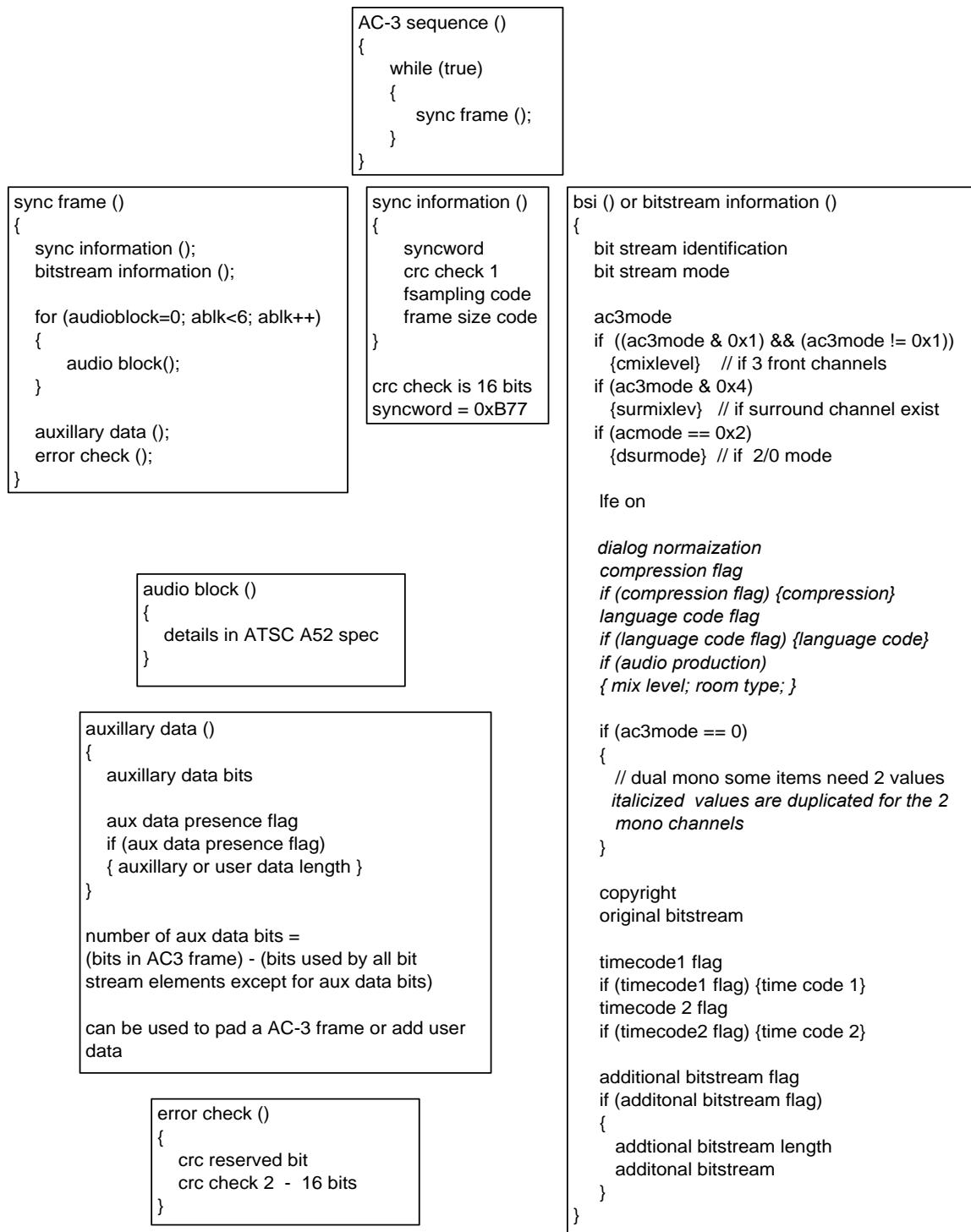


Figure 59. AC-3 audio frame format

A typical **AC-3 audio decoder** functional diagram and basic decoding steps are outlined below.

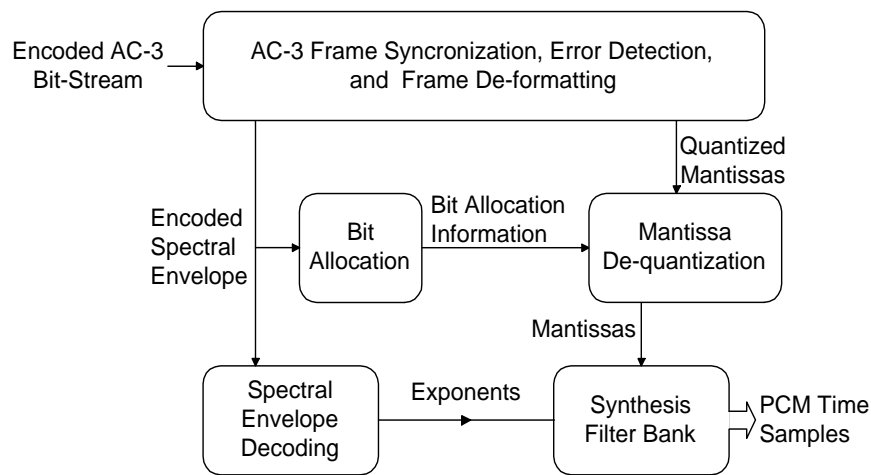


Figure 60. AC-3 audio decoding functional diagram

1. The decoder must first synchronize to the encoded bit stream, check for errors, and de-format the various types of data like the encoded spectral envelope and the quantized mantissas.
2. The bit allocation routine is run and the results used to unpack and de-quantize the mantissas.
3. The spectral envelope is decoded to produce the exponents.
4. The exponents and mantissas are transformed back into the time domain from the frequency domain to produce the decoded PCM time samples.
5. Error concealment or muting may be applied in case data errors are detected.
6. Channels which have had their high-frequency content coupled together must be de-coupled.
7. De-matrixing must be applied (in the 2-channel L/R mode) whenever the channels have been re-matrixed.
8. The synthesis filter bank resolution must be dynamically altered in the same manner as the encoder analysis filter bank had been during the encoding process.

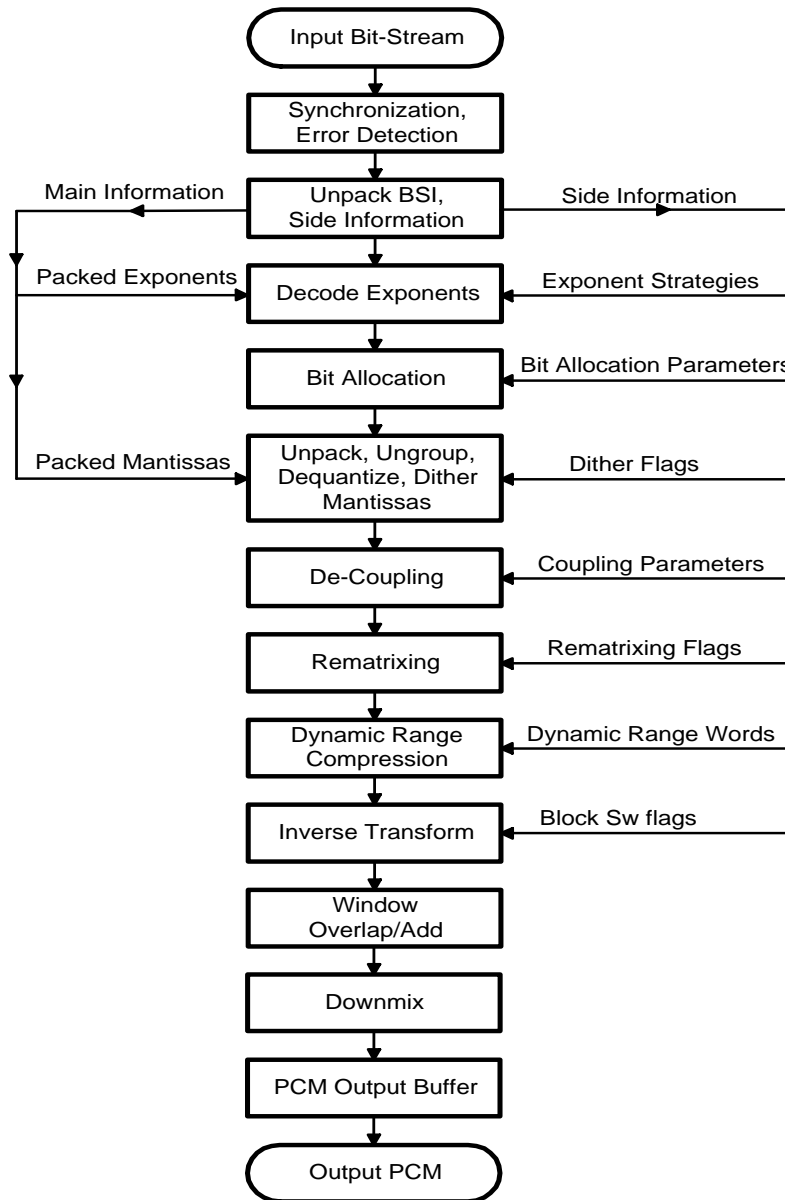


Figure 61. Details and flow diagram of a typical AC-3 decoder

1. The input bit stream typically comes from a transmission (satellite, cable, terrestrial) or storage (DVD, hard-disk) media. The input bit-stream interface to a AC-3 decoder affects the decoder implementation details.
2. The encoded AC-3 data may be input to the decoder as a continuous data stream at the average bit-rate, or chunks of data may be burst into the decoder. For burst mode operation, either the data source or the decoder must be the master controlling the burst timing. The AC-3 decoder input buffer may be smaller in size if the decoder can request bursts of data depending on its internal decode engine.

3. The sync-frame is an integral number of 16-bit words. The decoder may receive data as a continuous serial stream of bits which it has to sync to or, the data may be input to the decoder with either byte or word alignment. Alignment helps sync detection and verification.
4. When a sync pattern is detected the decoder is confirmed to be in sync and one of the CRC words (crc1 or crc2) can be checked. Since crc1 comes first and covers the first 5/8 of the frame, the result of a crc1 check may be available after only 5/8 of the frame has been received, or, the entire frame size can be received and crc2 checked. If either CRC checks, the decoder is presumed to be in sync and the decoding process can proceed.
5. Before the decoding process the unpacking of the various types of information included in the bit stream (BSI) must be done. Some of these items may be copied from the input buffer into the AC-3 decoders dedicated registers, some may be copied to specific working memory location, and some of the items may simply be located in the input buffer with pointers to them saved to another location for use when the information is required.
6. The exponents are delivered in the bit stream in an encoded form. In order to unpack and decode the exponents two types of side information are required. First, the number of exponents must be known and second, the exponent strategy in use by each channel must be known.
7. The bit allocation computation reveals how many bits are used for each mantissa. The inputs to the bit allocation computation are the decoded exponents, and the bit allocation side information. The outputs of the bit allocation computation are a set of bit allocation pointers one for each coded mantissa. A bit allocation pointer indicates the quantizer used for the mantissa, and how many bits in the bit stream were used for each mantissa.
8. The coarsely quantized mantissas make up the bulk of the AC-3 data stream. Each mantissa has been quantized to a level of precision indicated by the corresponding bit allocation pointer and the mantissa data can be packed by grouping some mantissas together into a single transmitted value. The mantissa data is unpacked by peeling off groups of bits as indicated by the bit allocation pointers. Grouped mantissas are ungrouped. The individual coded mantissa values are converted into de-quantized values. Mantissas which have zero bits are reproduced as zero, or are given a random dither value.
9. When coupling is used, the AC-3 audio channels which are coupled must be decoupled. Decoupling involves reconstructing the high frequency section (exponents and mantissas) of each coupled channel, from the common coupling channel and the coupling coordinates for the individual channel. Within each coupling band, the coupling channel coefficients (exponent and mantissa) are multiplied by the individual channel coupling coordinates.
10. If in the 2/0 audio coding mode rematrixing is employed for a band, the coefficients encoded in the bit stream are the sum and difference values instead of left and right values.
11. For each audio block, a dynamic range control value may be included in the bit stream. The decoder uses this value to alter the magnitude of the coefficient (exponent and mantissa).
12. The decoding steps described above will result in a set of frequency coefficients for each encoded channel. The inverse transform converts the blocks of frequency coefficients into blocks of samples in the time domain.
13. The individual blocks of time samples must be windowed, and adjacent blocks must be overlapped and added together in order to reconstruct the final continuous time domain PCM audio signal.

14. If the number of channels required at the decoder output is smaller than the number of channels which are encoded in the bit stream, then down-mixing is required. Below is a down-mix equation to obtain Lo/Ro stereo channels from the Lo, Ro, C, Ls and Rs channels in the time domain.

$$Lo = 1.0 * L + clef * C + slef * Ls$$

$$Ro = 1.0 * R + clef * C + slev * Rs$$

Where **clef** and **slef** are determined by **cmixlev** and **surmixlev** bit fields in the **BSI**.

15. PCM data is outputted at the PCM sampling rate. Since blocks of samples are processed by the decoding stages, a smoothing buffer is needed at the output. Also the PCM samples are sent out using different serial formats (16, 24, 32 data bits slot size, MSB, LSB first) for compatibility with various DACs.

Notes:

1. **3D audio** is a method of projecting and/or generating the surround information using a combination of just 2 left/right speakers. *In the absence of surround information (channels) in the encoded audio it generates and projects this information. In the presence of surround information in the encoded audio, it just projects this information.* This method uses sophisticated digital signal processing algorithms, such that a user perceives a complete 3-dimensional sound depth from just 2 speakers. This applies both to MPEG1/2 and AC-3 audio.
2. **AC-3 audio** is formatted and modulated (SPDIF) for transmission to an external decoder/receiver using two methods, IEC1937 (formats decoded linear PCM samples or frames) or IEC958 (formats encoded AC-3 synchronization frames). SPDIF digital output (clock modulated with data) is via a RCA or an optical jack. IEC1937 also applies to linear PCM samples or frames decoded from MPEG audio.
3. **MPEG or AC-3 frames** are called elementary bit-streams.
4. **MPEG4 audio** uses CELP (Code Excited Linear Prediction) for encoding PCM audio channels between 6 to 24 Kbits/sec. AAC encoding is used for all the bitrates exceeding 24 Kbits/sec.
5. **Maximum bit-buffering requirements** for all the audio standards is around 16 Kbits. Most Practical audio decoders use a bit-buffer to store atleast 2 full encoded audio frames to smoothen out the jitter in the delivery mechanisms.
6. **Analysis filterbanks** transform PCM signals into spectral coefficients. **Synthesis filterbanks** do the reverse. Polyphase filterbanks are a set of equal bandwidth (usually quadrature mirror) subband filters. Hybrid filterbank is a serial combination of a subband filterbank and MDCT (modified DCT).
7. **M/S stereo** is method of removing stereo imaging artifacts as well as exploiting stereo irrelevances (redundancies) in stereophonic audio programmes.
8. **Psychoacoustic model** is a mathematical model of the masking behaviour of the human auditory system.

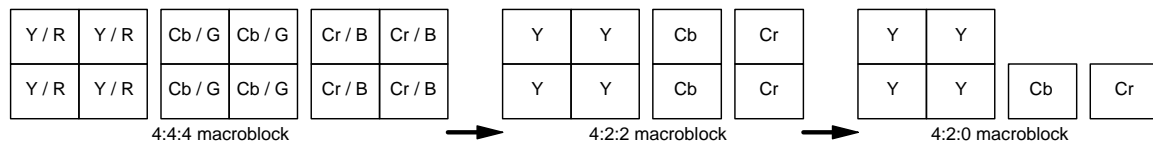
VIDEO COMPRESSION TECHNIQUES

The most widely used transform based (using the Discrete Cosine Transform) still image and video compression techniques are,

- **JPEG** (Joint Photographic Experts Group) for still image compression
- **MPEG** (Motion Pictures Experts Group) for video compression

JPEG Encoding/Decoding Algorithm

The **JPEG** compression algorithm is typically applied to 2 types of color spaces, **YUV (YCrCb) or RGB, 4:4:4, 24 bit raw digitized pixels** from any source (e.g. frame grabber, image editing packages, etc.). The JPEG algorithm is used to compress **each color component** to yield the final compressed picture.



A **JPEG compressed image** has the following format,

SOI (0xD8) Start of Image	APP0 (0xE0) JPEG Application Segment 0	APPn (0xE1 - 0xEF) JPEG Application Segment n	DQT (0xDB) Quantization Table	SOF0 (0xC0) Start of Frame	DHT (0xC4) Huffman Table	SOS (0xDA) Start of Scan	EOI (0xD9) End of Image
---------------------------------	--	---	-------------------------------------	----------------------------------	--------------------------------	--------------------------------	-------------------------------

A **JPEG image frame** consists of the following parts,

- A Start of Image (SOI)
- APP0 Marker
 1. APP0 length
 2. Identifier
 3. Version
 4. Units for X & Y densities
 5. X density
 6. Y density
 7. Thumbnail horizontal pixels
 8. Thumbnail vertical pixels
 9. Thumbnail RGB bitmap
- Optional (APPn) Markers where, n can be from 1 to 15
 1. APPn length
 2. Application Specific Information
- One or more quantization tables (DQT)
 1. Quantization table length
 2. Quantization table number
 3. Quantization table

- Start of Frame (SOF0)
 1. Start of Frame length
 2. Precision (Bits per pixel per color component, 8 bits in our example)
 3. Image height
 4. Image width
 5. Number of color components
 6. For each component
 1. An ID
 2. A vertical sample factor
 3. A horizontal sample factor
 4. A quantization table number
- One or more Huffman tables (DHT)
 1. Huffman table length
 2. Type, AC or DC
 3. Index
 4. A Bits table
 5. A Value table
- Start of Scan (SOS)
 1. Start of Scan length
 2. Number of color components
 3. For **each COLOR COMPONENT**
 1. An ID
 2. An AC table number
 3. An DC table number
 4. **COMPRESSED IMAGE DATA** scans (Not included in Start of Scan length)
- End of Image (EOI)

JPEG encoding algorithm applied to **one color component** is shown below,

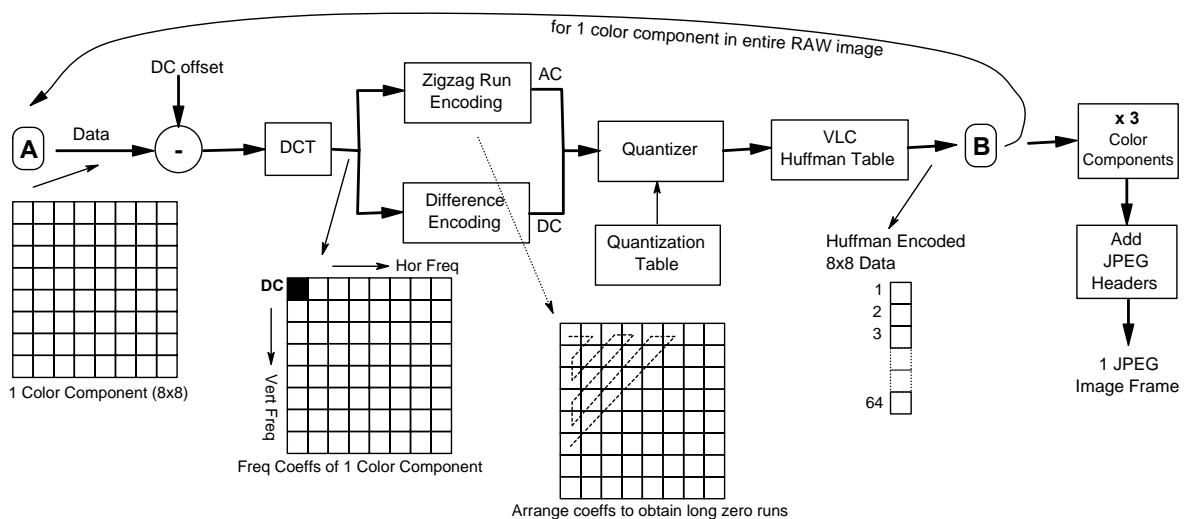


Figure 62. JPEG Encoding Algorithm for Single Color Component

1. The JPEG encoder splits the RAW raster image (either YUV or RGB) into its 3 color components, and scales each component to 8 bits.
2. Then each color component is converted into an 8x8 array and a DC offset if present (similar to normalization) is subtracted from it.
3. A 2D DCT is then applied to this 8x8 array. The DCT converts the raw pixels of the color component into frequency components (or coefficients) with the zero-frequency component (DC term) at the left hand corner. The other terms are the AC components with increasing frequency (horizontal & vertical) away from the DC component.
4. The DCT components are then quantized. The quantization step size varies with frequency and component. The dependence on frequency reflects the fact that the high frequency coefficients subjectively matter less than the low frequency ones and can be quantized with a larger step size (more coarsely). In addition an individual component may have its own quantization table. The JPEG committee allows up to 4 quantization tables for a maximum of 4 color components (e.g. the CMYK color table).
5. After quantization, the coefficients are re-ordered into a 1-dimensional array by reading out the entries of the 2D array in a zigzag fashion. In this way the quantized coefficients are “approximately” arranged in order of ascending frequency.
6. The zigzag coded AC coefficients are first run-length encoded. This process reduces each 8x8 block of DCT coefficients to a number of events. Each event represents a non-zero coefficient and the number of preceding zero coefficients. Since the high frequency coefficients are more likely to be zero, Huffman coding these events makes it possible to achieve high compression ratios.
7. Next the DC coefficients and AC events are coded losslessly using both Huffman-style (reduction in bits needed to represent a data set) encoding but keyed with different parameters. The DC coefficients are differentially encoded such that the DC coefficient of the previous 8x8 block of the same component is used to predict the DC coefficient of the current 8x8 block and the difference between these 2 DC terms is encoded. The Huffman code table for the DC term is based on the difference values. The JPEG algorithm allows up to 2 huffman tables each for encoding the AC and DC coefficients.

The JPEG algorithm is a **symmetric algorithm** meaning the encoding algorithm is simply run in reverse to decode the encoded image data set.

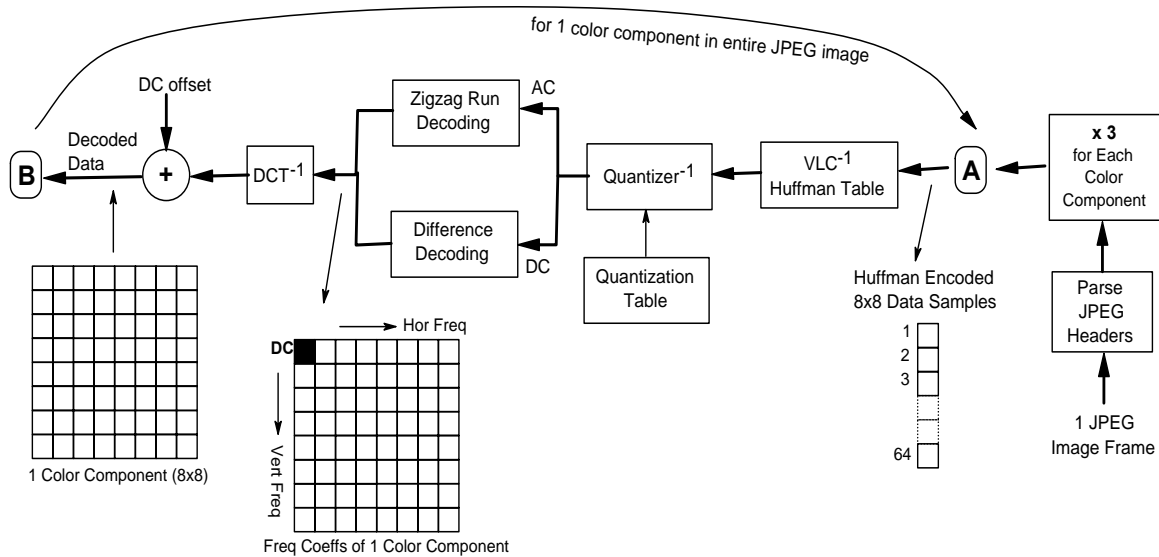


Figure 63. JPEG Decoding Algorithm for Single Color Component

In JPEG decoding, the decoder starts parsing the jpeg headers in the frame storing the variables extracted, till it reaches the Start of Scan (SOS) header. It then extracts the DC and AC Huffman tables for each color component and starts scanning (extracting) the VLC encoded color components passing them into **A**. The decoding chain shown builds up the complete original image by decoding 8x8 blocks.

MJPEG or motion JPEG is a method which basically takes a sequence of uncompressed raw images and applies the JPEG compression algorithm in series to each one of these images. No motion prediction is applied between each image.

MPEG Video Encoding and Decoding Methods

MPEG video encoding/decoding can be split up into the following subtypes,

- **MPEG1 (ISO/IEC 11172-2-video standard)** – supports image resolutions up to **352x240 or 352x288**. The input is 4:2:0 progressively scanned macroblocks (at 24, 25 or 30 Hz frame rates) to which the compression is applied. This contains I – intra coded, P – forward predicted and B – forward and backward predicted pictures.
- **MPEG2 (ISO/IEC 13818-2-video standard)** – this can be split into of 3 main types using profiles and levels.

1. **MP@LL (main profile @ low level) or MPEG1**, which supports image resolutions up to 352x240 or 352x288. The input is mainly 4:2:0 progressively scanned macroblocks (at 24, 25, or 30Hz frame rates). This contains I/P/B picture types.
 2. **MP@ML (main profile @ main level)** which supports image resolutions up to 720x480 or 720x576. The input is mainly 4:2:0 interlaced or progressively scanned macroblocks (at 24, 25, or 30Hz frame rates). This contains I/P/B picture types and encompasses MP@LL. This is mainly used in SDTV (standard definition television).
 3. **MP@HL (main profile @ high level)** which supports image resolutions up to 1920x1152. The input can be 4:2:0 or 4:2:2 interlaced or progressively scanned macroblocks (at almost any frame rate supporting TV and PC monitors up to 60Hz). This contains I/P/B picture types and encompasses MP@LL & MP@ML. The higher resolutions are mainly used in HDTV (high definition television).
- **MPEG4 (ISO/IEC 14496-visual standard, H.264 video standard)** – the visual standard allows the hybrid coding of natural (pixel based) images and video together with synthetic (computer generated) scenes. Also the Visual standard comprises of tools and algorithms supporting the coding of natural still images and video sequences as well as tools to support the compression of synthetic 2-D and 3-D graphic geometric patterns.
 1. **Bitrates ranging from 5 kbits/sec to 4 Mbits/sec**
 2. **Interlaced and Progressive Formats**
 3. **Resolutions ranging from QCIF (170x144) to SVGA (800x600)**
 4. **Robust Error Resiliency methods**

Some details of the **MPEG2 video encoding** process,

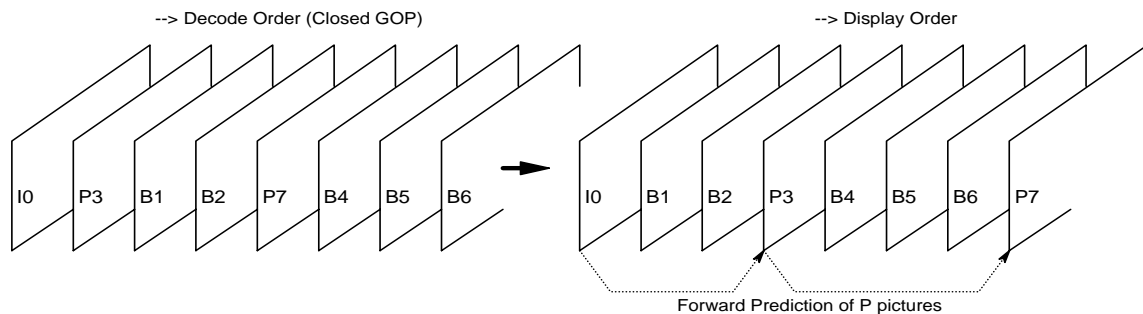
LEVEL	Spatial Resolution Layer	Profile / Level Maximum Parameters	PROFILE
			MAIN
HIGH	Enhancement	Samples/Line Lines/Frame Frames/sec Macroblock VBV Buffer Size - Mbits Bitrate - Mbits	1920 1152 60 4:2:0 9.781248 80
MAIN	Enhancement	Samples/Line Lines/Frame Frames/sec Macroblock VBV Buffer Size - Mbits Bitrate - Mbits	720 576 30 4:2:0 1.835008 15
LOW	Enhancement	Samples/Line Lines/Frame Frames/sec Macroblock VBV Buffer Size - Mbits Bitrate - Mbits	352 288 30 4:2:0 0.475136 1.5

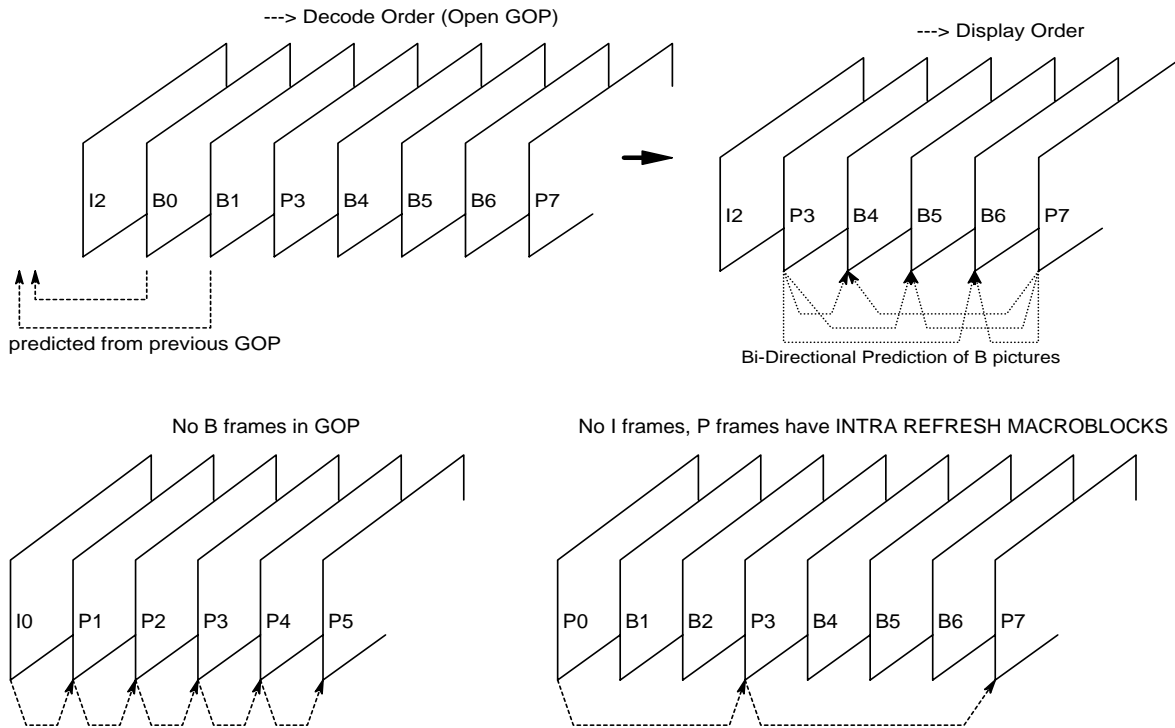
Profile : A Subset of the ISO/IEC 13818-2 specification
 Level : Set of constraints on the parameters in the bitstream
 Bitrate : Most MP@HL video bitrates for ATSC (HD) are ~18 Mbits/sec
 : Most MP@ML video bitrates for Satellite/Cable are ~6/8 Mbits/sec
 : Most MP@ML video bitrates for DVD are ~6/12 Mbits/sec
 : Most MP@LL video bitrates for MPEG1 are ~1.5 Mbits/sec

We will discuss MPEG2 video encoding/decoding (**without SNR or Spatial scalable modes**) using **4:2:0 macroblocks only**, since this is the most widely used video compression standard (for standard & high definition video and also encompassing MPEG1 video) in satellite, cable & terrestrial transmission systems and also in recording/playback devices like DVDs.

Non-Scalable Syntax – the coded representation of video defined in the non-scalable syntax achieves high compression ratios and good image quality. The algorithm used is lossy as the exact sample values are not preserved during coding. Obtaining good image quality at the required bit-rates cannot be done purely by intra-coding pictures, hence a mix of intra-coded, forward predicted and bi-directionally predicted pictures are generated in this type of encoding. This is done using block based motion compensation to reduce the **temporal redundancies** in the video. **Motion compensation** is used both for **causal** prediction of the current picture from a previous picture, and for **non-causal**, interpolative prediction from past and future pictures. **Motion vectors** are defined for each 16-sample by 16-line region of the picture. The prediction error, is further compressed using the Discrete Cosine Transform (DCT) to remove any **spatial** correlation before it is quantized by removing less important (high frequency coefficients) information. Finally the motion vectors are combined with the quantized DCT information and encoded using variable length (Huffman) codes.

Temporal Processing – Three (3) main picture types are defined. **Intra coded pictures (I-pictures)** are coded without reference to other pictures and are called reference pictures. These reference or I-pictures provide access points (anchor points) into the coded sequence where the decoding can begin. **Predictive coded pictures (P-pictures)** are coded more efficiently using motion compensated prediction from a past intra I-picture or a past predictive coded picture P-picture and are used as references for further prediction. **Bidirectionally-predictive coded pictures (B-pictures)** are predicted from both the past and future reference pictures using motion compensation and provided the highest degree of compression. B-pictures are never used as references for further prediction. These pictures are organized by the encoder in a typical **mpeg video sequence** depending on the **group of picture structure (GOP)**.



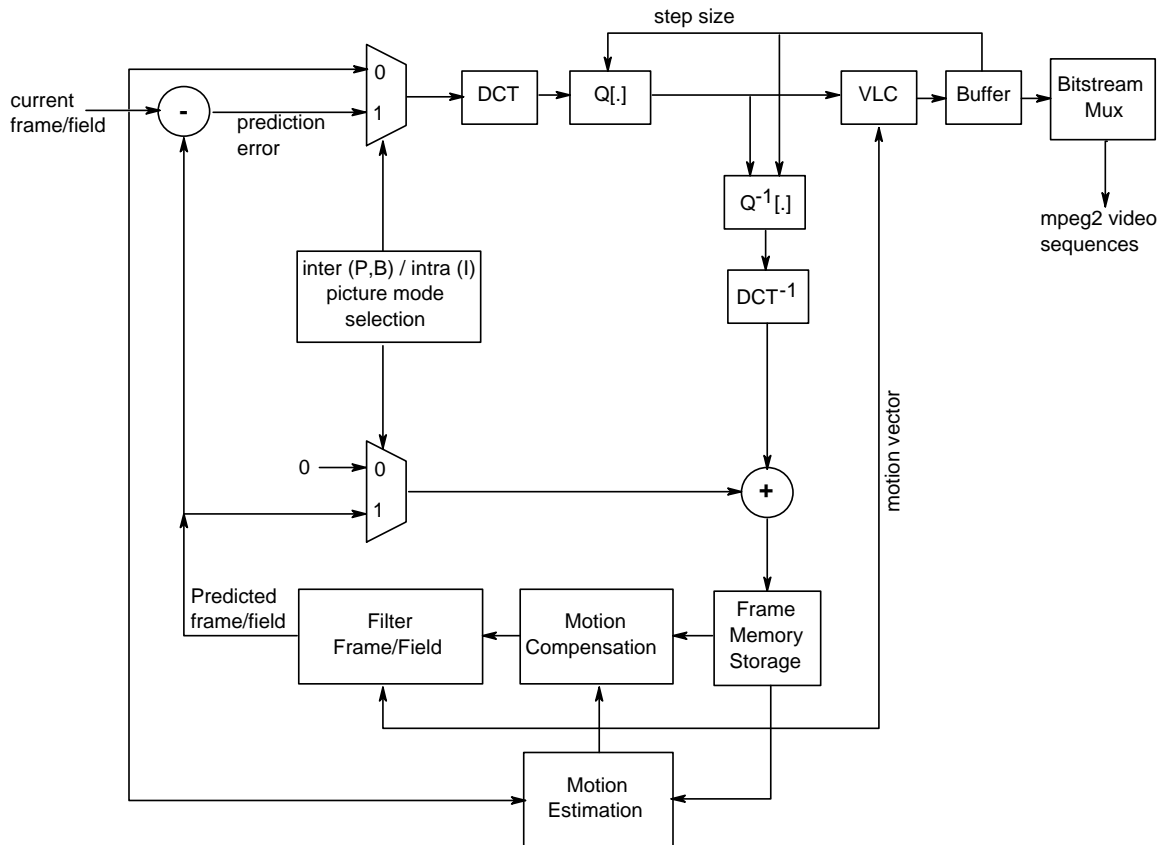


Coding Interlaced Video – Each frame of interlaced video (scanned by a source field by field) consists of 2 fields which are separated by 1 field period. The standard allows either the frame to be encoded as 1 picture or the 2 fields to be encoded as 2 pictures. Frame encoding or field encoding can be selected on a frame by frame basis. Frame encoding is typically preferred when the video scene contains significant detail but limited motion. Field encoding, where the 2nd field can be predicted from the first field works better when there is lot of motion in the interlaced video.

Motion Representation (using 16x16 Luminance Pixel Macroblocks) – 16x16 luminance (since changes in luminance is indicative of motion) pel macroblocks are used in the motion compensation unit. Each macroblock can be temporally predicted in a number of different ways. For example, in frame encoding, the prediction from the previous reference frame can itself be either frame-based or field-based. Depending on the type of the macroblock, motion vector and other side information is encoded along with the compressed prediction error in each macroblock. The motion vectors are encoded differentially w.r.t the last encoded motion vectors using variable length (Huffman) codes. The maximum length of the motion vectors can be adaptively programmed by the encoder on a picture by picture basis to meet the most demanding applications.

Spatial Redundancy Reduction – Both source pictures (I) and prediction errors (P/B) have a high spatial redundancy. The MPEG2 standard uses a block based DCT with visually weighted quantization and run-length encoding. After motion compensated prediction or interpolation, the resulting prediction error is split into

8x8 blocks. These are transformed by the DCT, then weighted and quantized. After quantization, since most of the DCT coefficients are zero a 2-D run length encoding along with variable length coding is used to encode the remaining DCT coefficients. The process is the same for source un-predicted I pictures.



Motion vector calculated is a 2D vector using luma information
 Motion vector for chroma blocks is a scaled luma motion vector by 2 in both H/V directions
 Motion vector search range is usually limited in the horizontal/vertical direction by a encoding parameter
 Motion vectors in MPEG2 have a resolution of 1/2 pixel (0.5 pel)

Figure 64. MPEG2 Encoding flow diagram

- It can be seen that depending on the picture type (Intra coded I or inter predicted P and B) the paths through the encoder are quite different.
- If the picture is an I picture (intra mode in the 2 multiplexers) then the current frame or field does not go through the motion estimation/prediction path. Its 4:2:0 macroblocks are split into 8x8 blocks, transformed by a DCT stage, quantized, run length coded, over which variable length encoding is applied and finally muxed into the MPEG2 bitstream.
- These intra coded pictures are also reconstructed using an inverse quantization and an inverse DCT stage into the encoders frame memory for predicting other frames/fields.
- If the encoder decides to encode the next frame/field as a P or a B, then the 2 multiplexers are switched into inter mode. In this mode to compensate for

motion each 16x16 luma macroblock in the current frame is matched with a search window in the frame memory. Then the motion vector that represents the offset between the current macroblock and a macroblock in the prior reconstructed image that forms the best match is chosen.

- This motion vector is further differentially encoded with the previous motion vector, that difference is huffman coded and sent to the bitstream multiplexer.
- The motion prediction block (using a combination of motion estimation and compensation) produces a predicted frame/field which is subtracted from the next frame to give a prediction error which is converted into 8x8 blocks, transformed, quantized, run length and VLC encoded and muxed into the bitstream.
- Also this predicted frame is summed with the prediction error (after inverse quantization and inverse DCT) and stored into frame memory. This will be the reference frame for the next prediction stage.

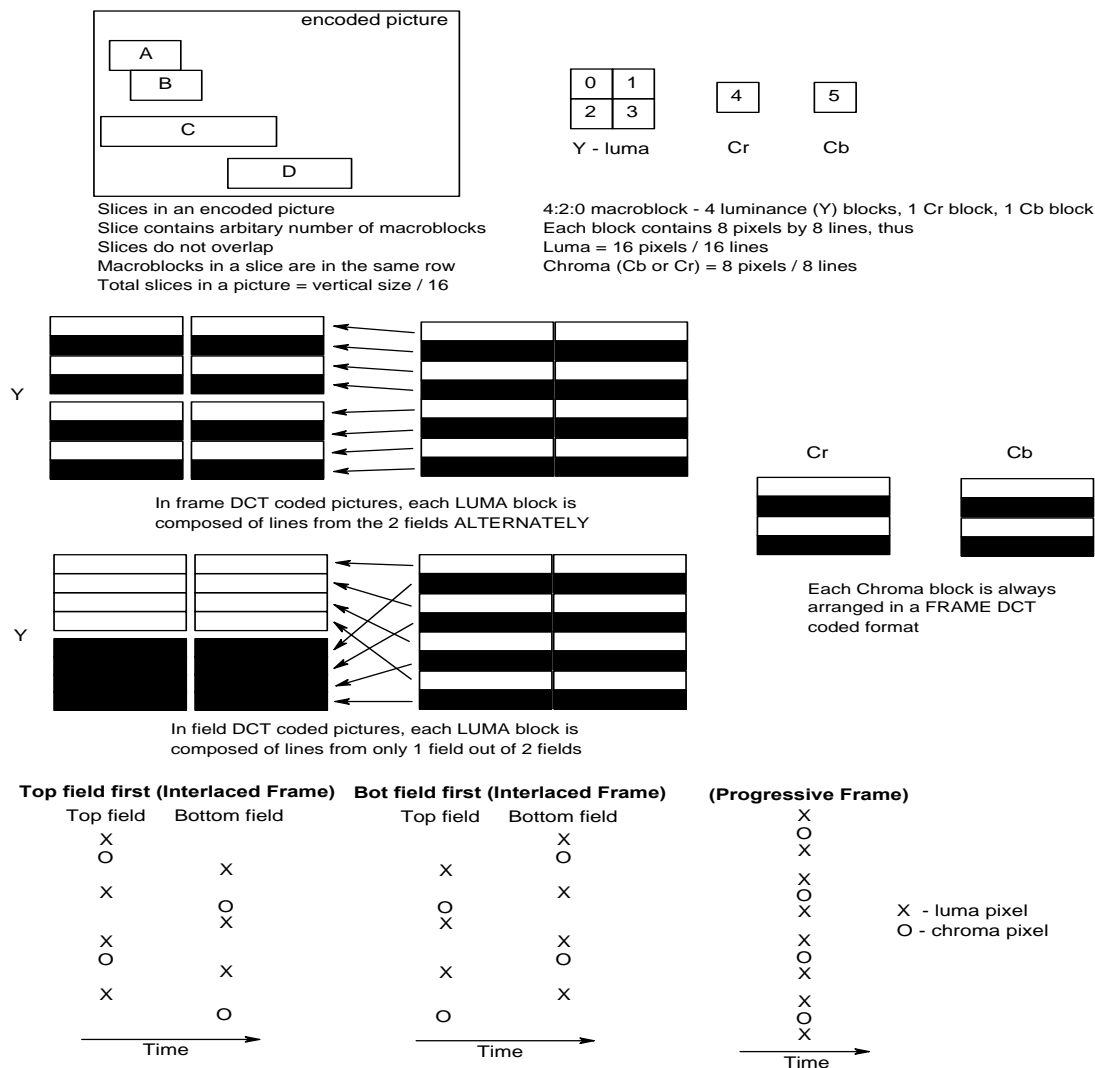
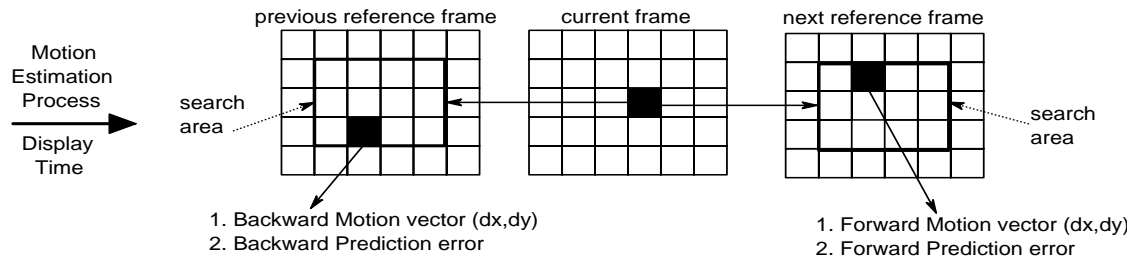
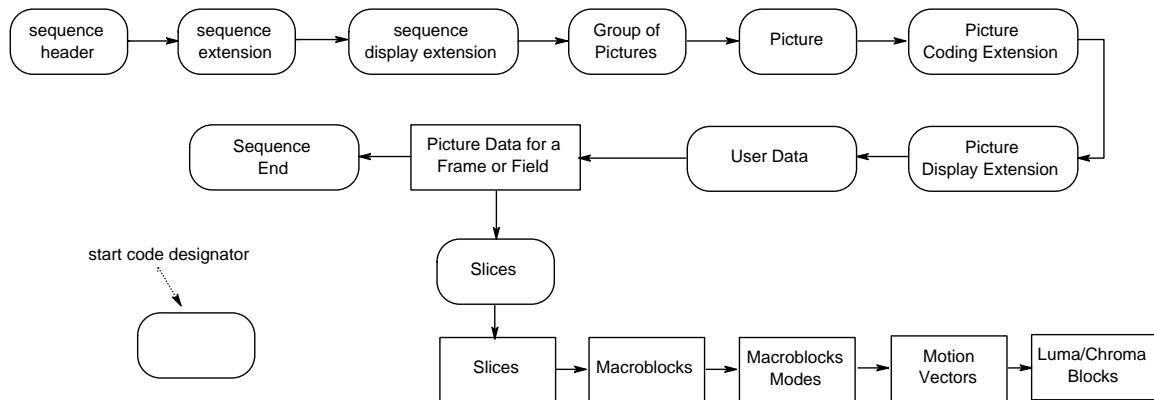


Figure 65. Slice and 4:2:0 Macroblock Organization

In figure 65 we have shown the slice and 4:2:0 macroblock organization in an encoded frame/field. We have also shown how the luminance and chrominance samples appear in time in a 4:2:0 interlaced or progressive frame.



The MPEG2 video stream format and its various headers are shown below. After each header start code relevant encoding information about that specific video bitstream is added. The encoded picture data (put together as slices → macroblocks → luma/chroma blocks) is added after the picture header and its coding and display extensions.



Typical start codes and their extensions found in the MPEG2 video stream are shown below,

Start Code Name	Start Code Value	Extension Start Code Identifier	4 Bit Value
sequence header start code	0x000001B3	Sequence Extension ID	0001
sequence error start code	0x000001B4	Sequence Display Extension ID	0010
Any header extension start code	0x000001B5	Quant Matrix Extension ID	0011
sequence end start code	0x000001B7	Copyright Extension ID	0100
group of pictures header start code	0x000001B8	Sequence Scalable Extension ID	0101
picture header start code	0x00000100	Picture Display Extension ID	0111
slice header start code	0x000001 (0x01 - 0xAF)	Picture Coding Extension ID	1000
user data header start code	0x000001B2	Picture Spatial Scalable Extension ID	1001
system start codes	0x000001 (0xB9 - 0xFF)	Picture Temporal Scalable Extension ID	1010

The format of the main header types found in a typical MPEG2 video sequence is shown ahead.

```

video sequence ()
{
    do {
        next start code ()
        sequence header ()
        if (nextbits() == any_extension_start_code + sequence extension ID)
        {
            sequence extension header ()
        }
        if (nextbits() == any_extension_start_code + sequence display extension ID)
        {
            sequence display extension header ()
        }
    }

    do {
        if (nextbits() == group of pictures start code)
        {
            group of pictures header ()
        }
        if (nextbits() == picture start code)
        {
            picture header ()
        }
        if (nextbits() == any_extension_start_code + picture coding extension ID)
        {
            picture coding extension ()
        }
        if (nextbits() == any_extension_start_code + quantization matrix extension ID)
        {
            quant matrix extension ()
        }
        if (nextbits() == any_extension_start_code + copyright extension ID)
        {
            copyright extension ()
        }
        if (nextbits() == any_extension_start_code + picture display extension ID)
        {
            picture display extension ()
        }
        if (nextbits() == user data header start code)
        {
            user data ()
        }
    }

    picture data ()

} while ( (nextbits() == picture start code) || (nextbits() == group of pictures start code))
} while (nextbits() != sequence end code)
}

```

MP@LL or MPEG1 does not have the
sequence or picture extensions

```

sequence header ()
{
    sequence header code
    horizontal size value
    aspect ratio information
    frame rate code
    bit rate code
    market bit
    vbv buffer size value
    constrained parameters flag
    load intra quantizer matrix flag
    if (load intra quantizer matrix flag == true)
        intra_quantizer_matrix [64]
    load non-intra quantizer matrix flag
    if (load non-intra quantizer matrix flag == true)
        non_intra_quantizer_matrix [64]

    next start code ()
}

```

```

sequence extension header ()
{
    extension start code
    extension start code identifier
    profile and level indication
    progressive sequence
    chroma format
    horizontal size extension
    vertical size extension
    bit rate extension
    marker bit
    vbv buffer size extension
    low delay
    frame rate extension_n
    frame rate extension_d

    next start code ()
}

```

```

sequence display extension header ()
{
    extension start code identifier
    video format
    color description flag
    if (color description flag)
    {
        color primaries
        transfer characteristics
        matrix coefficients
    }
    display horizontal size
    marker bit
    display vertical size

    next start code ()
}

```

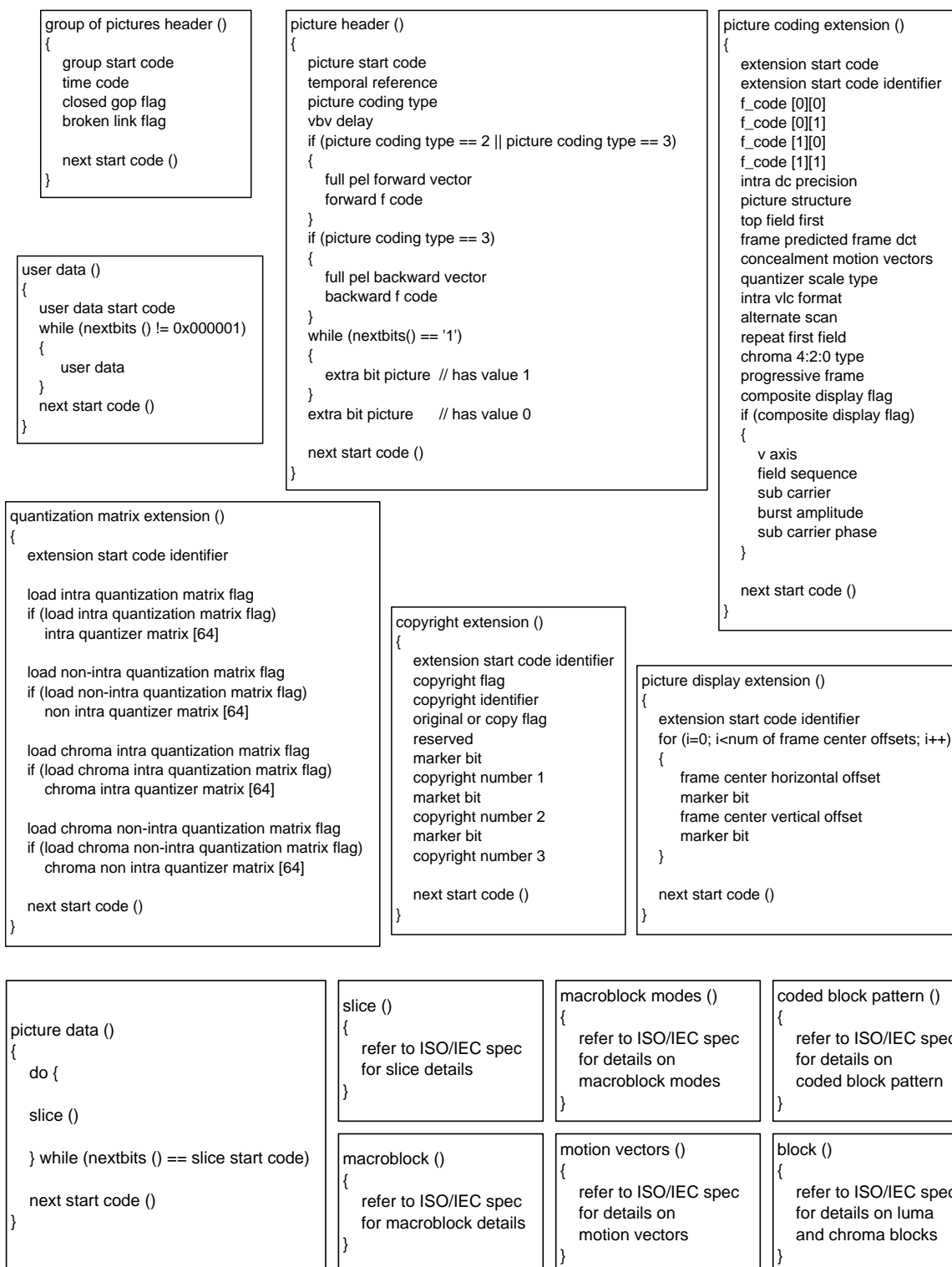


Figure 66. MPEG2 Video Sequence (Elementary Stream) Header Format

Some details of the MPEG2 video decoding process,

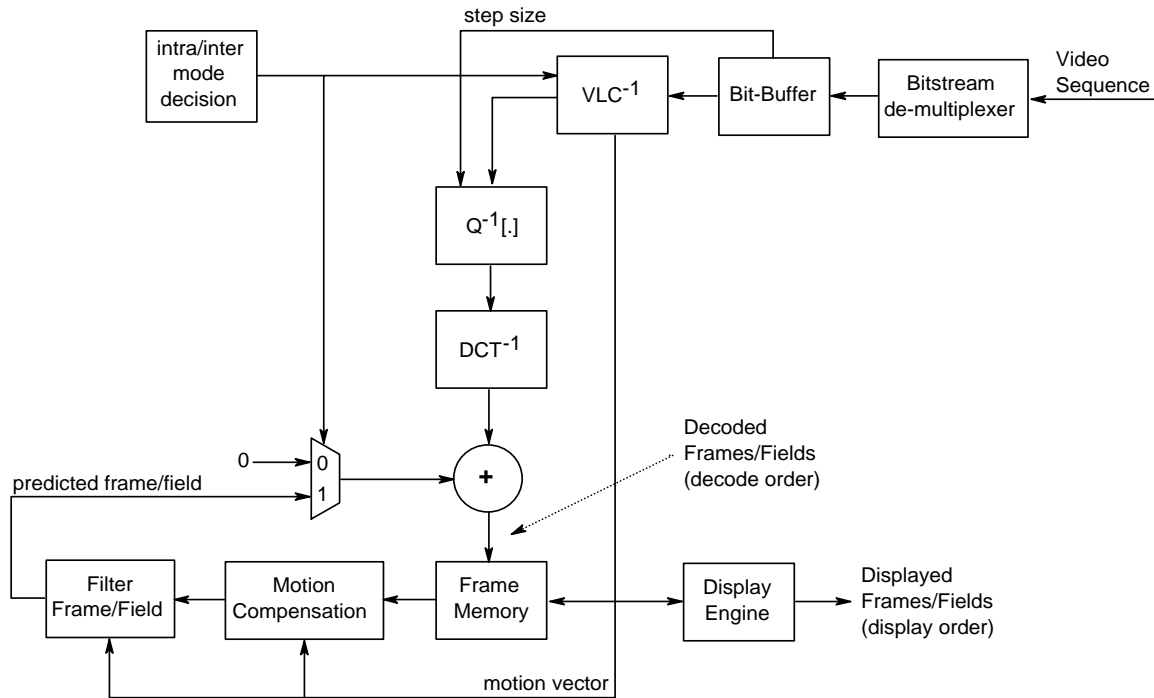


Figure 67. Typical MPEG2 Decoder

A mpeg 2 decoder as shown in figure 67, has the following components,

- Bit-stream de-multiplexer – this is responsible for parsing all the headers in the video sequence or a video elementary stream, extracting all the required information (fields) to decode that particular bitstream (this information can then be used to program registers in a hardware decoder or fill appropriate decoding/display structures in a software decoder). The header parsing process (a.k.a. start code detection and associated header parsing) is usually carried out in parallel with the video decoding process.
- Buffer – this is also known as a bit-buffer and can be broken up into the following parts,
 1. VBV buffer size (MP@LL, MP@ML and MP@HL VBV buffer sizes vary)
 2. Storage for $\frac{1}{2}$ the number of bits in the maximum picture size
 3. 4 ms typical MPEG jitter (4ms * bit-rate)

The video decoder has to examine this buffer and only if it is filled to a certain level (Threshold T, usually selected to $\frac{3}{4}$ picture size), remove the encoded data from this buffer into the video decoding pipeline. In low delay (low encoding delay) applications (no B frames and no re-ordering at the display), an encoded I or P picture may fill this buffer very slowly, so re-examination of the buffer may be needed, in order to feed (without starving) the video decoding pipeline at the correct rate.

- Variable length decoder – this decodes the VLC encoded data from the 3 encoded paths,
 1. Intra coded blocks from I frames
 2. Prediction error blocks from P and B frames
 3. Differentially encoded motion vectors
- Inverse Quantization – this de-quantizes the VLC decoded data using an adaptive mechanism which modifies the quantizer step size using 2 methods,
 1. A weighting matrix modifies the step size within a 8x8 block
 2. A scale factor is used to keep the modified quantizer step size within a few bits
- Inverse DCT – this takes the de-quantized coefficients for the 8x8 intra coded blocks (I frames) or the 8x8 predicted error blocks (P/B frames) and converts them back into the time domain.
- Motion Compensation unit – if the frame is an I frame, then it is stored in frame memory. The I frames along with the VLC decoded motion vectors are used to predict P frames. A P frame is also stored in memory. The P frames along with the I frames and the VLC decoded motion vectors are used to predict B frames. The VLC decoded motion vectors give the locations for the predicted blocks within the reference (I or P) frames.
- Display unit – this is responsible for re-ordering the decoded frames for display (e.g. closed GOP decode order is I0P3B1B2 → display I0B1B2P3 order). The responsibility of the display engine is to take the decoded frames or fields and send them to a display device (NTSC/PAL TV or a HD monitor when the streams are interlaced or progressive HD streams). Also the display unit has to accomplish typically 7 things,
 1. Convert the 4:2:0 decoded macroblocks to 4:2:2 or 4:4:4 macroblocks (by interpolating chroma in the V/H direction) which can be accepted by digital encoders or display devices with digital interfaces (e.g. DVI-digital video interface, digital YUV or RGB).
 2. Scale the decoded frame in the horizontal and vertical direction as needed by the user or display device.
 3. If decoded interlaced frames are to be displayed on a progressive display, then these frames have to be de-interlaced (using a motion detection method) with a possible frame rate up-conversion (x2 frame rate).
 4. If decoded progressive frames are to be displayed on an interlaced display, then the display engine has to output each frame one field at a time.
 5. Most HD video streams can be progressive/interlaced which have to be displayed on a combination of interlaced/progressive displays. Due to this reason the display engine on most decoders contain a vertical timing generator which can be programmed with almost any horizontal line or vertical frame rate. The display engine also contains a horizontal/vertical format convertor to scale the horizontal pixels and vertical lines to match any line/frame rate.

6. Change the origin of display of the frame/field using the pan (horizontal) and scan (vertical) vectors encoded in the bit-stream. This is to accommodate displaying 4/3 streams on 16/9 displays and vice versa.
7. When the repeat_first_field_flag is set to 1 for progressive frames, then that decoded frame is displayed as follows (Top_{field} Bottom_{field} Top_{field}). Many encoders are now using this mechanism to save bandwidth (user gets 3 fields instead of the usual 2).

MPEG4 video encoding/decoding of natural objects

The **MPEG4 visual standard** was developed to code and represent **visual objects** of **natural** or **synthetic origin**.

The visual part of the MPEG-4 standard provides tools and algorithms for achieving the following,

- Efficient compression of images and video
- Efficient compression of textures for texture mapping on 2D and 3D meshes
- Efficient compression of implicit 2D meshes
- Efficient compression of time-varying geometry streams that animate meshes
- Efficient random access to all types of visual objects
- Extended manipulation functionality for images and video sequences
- Content-based coding of images and video
- Spatial, temporal and quality scalability
- Error robustness and resilience in error prone environments

We will discuss the MPEG4 tools for coding and representing **natural video objects only**. The tools for representing natural video in the MPEG-4 visual standard aims to provide a standardized core technology allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools allow the decoding and representation of atomic units of image and video content, called “**Video Object Planes**” (**VOPs**).

The MPEG4 **Video Verification Model (VM)** consists of the following features,

- Standard YUV luminance and chrominance intensity representation of regularly sampled pixels in 4:2:0 macroblock format. The intensity of each Y, U or V pixel is quantized into 8 bits. The image size and shape depends on the application.
- Coding of multiple "Video Object Planes" (VOPs) as images of arbitrary shape to support many of the content-based functionalities. Thus the image sequence input for the MPEG-4 Video VM is in general considered to be of arbitrary shape - and the shape and location of a VOP within a reference window may vary over time. The coding of standard rectangular image input sequences is supported as a special case of the more general Video Object Plane approach.

- Coding of shape and transparency information of each VOP by coding binary or gray scale alpha plane image sequences using a particularly optimized Modified Modified Reed Code method (M⁴R).
- Support of Intra (I) coded VOPs as well as temporally predicted (P) and bi-directionally (B) predicted VOPs. Standard MPEG I, P and B frames are supported as a special case.
- Support of fixed and variable frame rates of the input VOP sequences of arbitrary or rectangular shape. The frame rate depends on the application.
- 8x8 pel block-based and 16x16 pel Macroblock-based motion estimation and compensation of the pixel values within VOPs, including provisions for block-overlapping motion compensation.
- Texture coding in I, P and B-VOPs using a 8x8 Discrete Cosine Transform or alternatively a Shape-Adaptive DCT (SADCT) adopted to regions of arbitrary shape, followed by MPEG-2 like quantization and run-length coding.
- Efficient prediction of DC and AC coefficients of the DCT in Intra coded VOPs.
- Quarter Pel Motion Compensation to enhance the precision of the motion compensation scheme, at the cost of only small syntactical and computational overhead. An accurate motion description leads to a smaller prediction error and thus provides better visual quality.
- Support for efficient static as well as dynamic SPRITE prediction of global motion from a VOP panoramic memory using 8 global motion parameters.
- Temporal and spatial scalability for arbitrarily shaped VOPs.
- Adaptive macroblock slices as well as improved bit stuffing and motion markers for resynchronization in error prone environments.
- Backward compatibility with standard MPEG1/2 coding algorithms if the input image sequences are coded in a single layer using a single rectangular VOP structure.

Description of the **MPEG4 Video Encoding/Decoding** Verification Model,

- **Provisions for Content Based Functionality - Decomposition into "Video Object Planes"**

The MPEG-4 Video coding algorithm will eventually support all functionalities already provided by MPEG-1 and MPEG-2, including the provision to efficiently compress standard rectangular sized image sequences at varying levels of input formats, frame rates and bit rates as well as provisions for interlaced input sources.

Furthermore, at the heart of the so-called "content"-based MPEG-4 Video functionalities, is the support for the separate encoding and decoding of content (i.e. physical objects in a scene). Within the context of MPEG-4 this functionality - the ability to identify and selectively decode and reconstruct video content of interest - is referred to as "Content-Based Scalability". This MPEG-4 feature provides the most elementary mechanism for interactivity and manipulation

with/of content of images or video in the compressed domain without the need for further segmentation or trans-coding at the receiver.

To enable the content based interactive functionalities envisioned, the MPEG-4 Video Verification Model introduces the concept of **Video Object Planes (VOPs)**. It is assumed that each frame of an input video sequence is segmented into a number of arbitrarily shaped image regions (video object planes) - each of the regions may possibly cover particular image or video content of interest (i.e. describing physical objects or content within scenes). In contrast to the video source format used for the MPEG-1 and MPEG-2 standards, the video input to be coded by the MPEG-4 Verification Model is no longer a rectangular region.

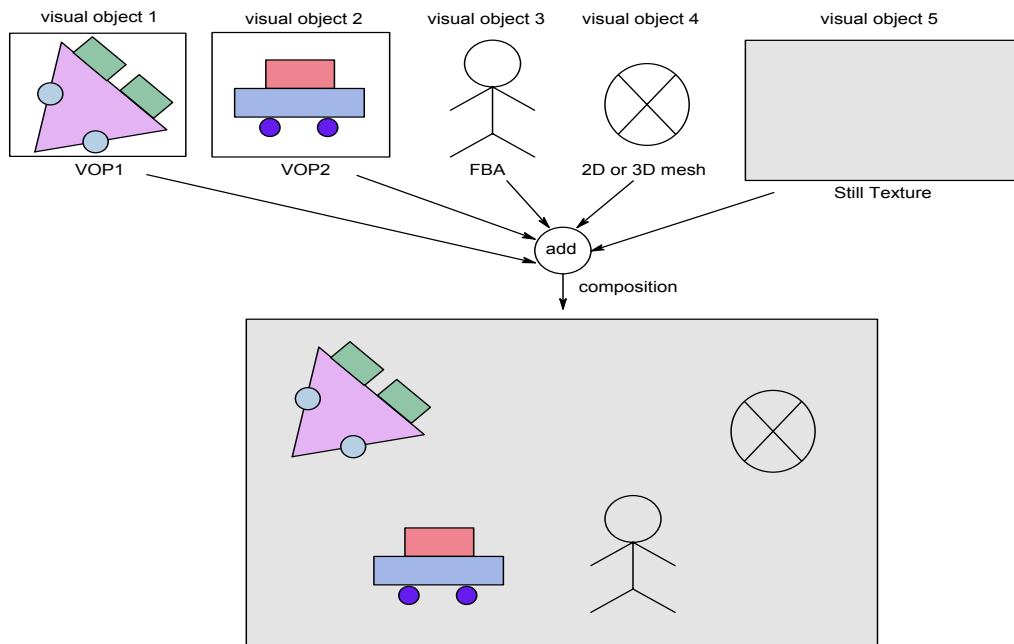
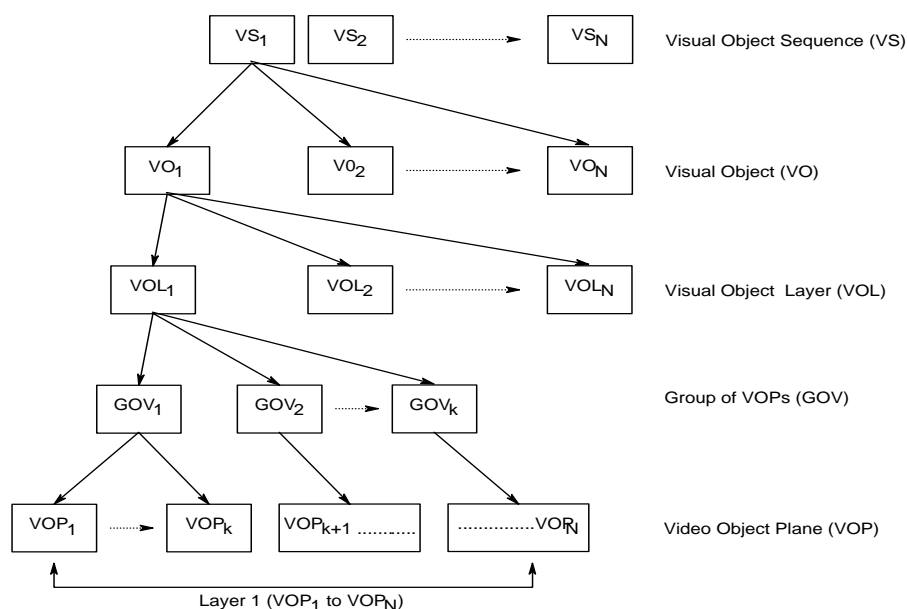


Figure 68. Decomposition/Composition of a MPEG4 visual scene

A MPEG4 visual scene could contain the above **5 visual objects** which includes **2 VOPs (we will NOT discuss how visual objects 3, 4, 5 are encoded)**. Each VOP specifies particular image sequence content and is coded into a separate VOL-layer (by coding shape/contour, motion and texture information). Decoding of all VOP-layers reconstructs the original image sequence. Thus content can be reconstructed by separately decoding a single or a set of VOL-layers (enabling content-based scalability/access in the compressed domain at the decoder).

The input to be coded can be a VOP image region of arbitrary shape and the shape and location of the region can vary from frame to frame. Successive VOPs belonging to the **same physical object** in a scene are referred to as Video Objects (VOs) - a sequence of VOPs of possibly arbitrary shape and position. The shape, motion and texture information of the VOPs belonging to the same VO is encoded and transmitted, or coded into a separate VOL (Video Object

Layer). In addition, relevant information needed to identify each of the VOLs and how the various VOLs are to be composed at the receiver for reconstructing the entire original sequence is also included in the bitstream. This allows the separate decoding of each VOP and the required flexible manipulation of the video sequence.



Visual Object Sequence (VS) - Ordered collection of Visual Objects (Video, Still, Mesh and FBA objects)

Visual Object (VO) - A VO could be a Video Object or a Mesh Object, etc.

Visual Object Layer (VOL) - If the MPEG4 video contains **multiple resolutions or bit-rates** then it can have **one VOL for each resolution or bit-rate**.

Group of VOPs (GOV) - In the context of video coding, a GOV is analogous to a GOP in MPEG1/2, a group of pictures. Generally 8 to 10 pictures for a group. The purpose of GOV is to facilitate random access in the video.

Video Object Plane (VOP) - Again, in the context of video coding, a VOP is analogous to a Frame in MPEG1/2. One difference between a VOP and a frame, is a VOP can contain both data (YUV data) and shapes (alpha plane), while a frame contains just data.

MPEG 4 can contain several LAYERS for each visual object – we have shown just 1 layer in the above tree. Below is a typical MPEG4 encoded bit-stream for the 2 VOPs corresponding to Video object 1 and Video object 2.

Visual Object Sequence 1 Header	Visual Object 1 Header	Visual Object 1 Visual Object Layer 1 Header	Elementary Stream - bitrate 1 Video Object 1 shape, motion, texture
same	same	Visual Object Layer 2	bitrate 2 - shape, motion, texture
same	same	Visual Object Layer 3	bitrate 3 - shape, motion, texture
Visual Object Sequence 1 Header	Visual Object 2 Header	Visual Object 2 Visual Object Layer 1 Header	Elementary Stream - resolution 1 Video Object 2 shape, motion, texture
same	same	Visual Object Layer 2	resolution 2 - shape, motion, texture
same	same	Visual Object Layer 3	resolution 3 - shape, motion, texture

If the original input image sequences are not decomposed into several VOLs of arbitrary shape, the coding structure simply degenerates into a single layer representation which supports conventional image sequences of rectangular shape. The MPEG-4 content-based approach can thus be seen as a logical extension of the conventional MPEG-1/2 coding approach towards image input sequences of arbitrary shape.

- **Coding of Shape, Motion and Texture Information for each VOP**

The information related to the **shape, motion and texture information for each VO is coded into a separate VOL-layer** in order to support separate decoding of Video Objects. The MPEG-4 Video VM uses an **identical algorithm to code the shape, motion and texture information** in each of the layers. The shape information is, however, not transmitted if the input image sequence to be coded contains only standard images of rectangular size. In this case the MPEG-4 Video coding algorithm has a structure similar to the successful MPEG2 coding algorithm - suitable for applications which require high coding efficiency without the need for extended content based functionalities.

The MPEG-4 VM compression algorithm employed for coding each VOP image sequence is based on the **successful block-based hybrid DPCM/Transform coding technique already employed in the MPEG coding standards**. The MPEG-4 coding algorithm encodes the first VOP in Intra-Frame VOP coding mode (I-VOP). Each subsequent frame can be coded using either forward VOP prediction (P-VOP) or B-directional VOP prediction (B-VOP) similar to MPEG1/2.

Similar to the MPEG baseline coders the MPEG-4 Verification Model algorithm processes the successive images of a VOP sequence block-based (the VOP is split into 4:2:0 macroblocks) and sends them into the encoder.

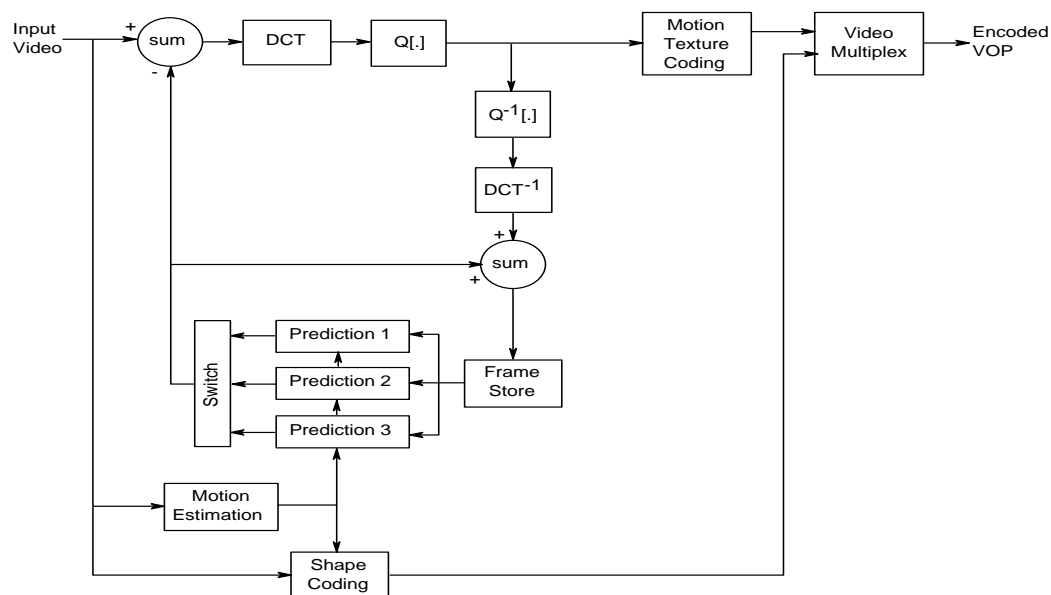
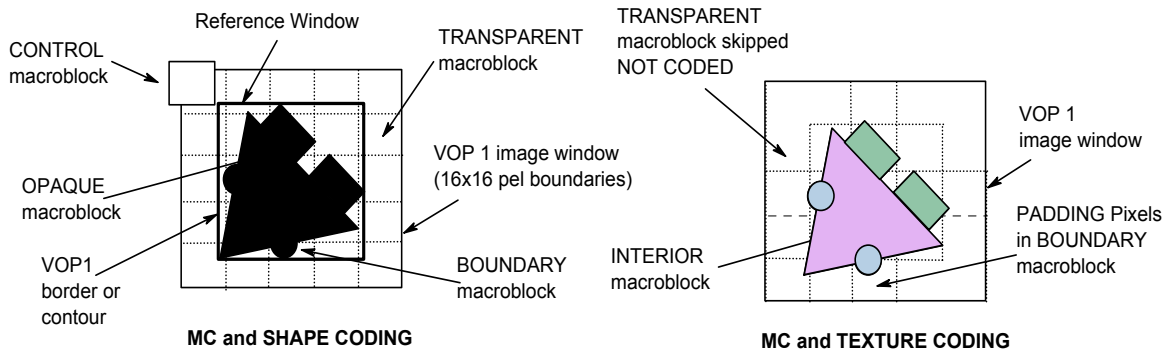


Figure 69. MPEG4 VOP encoder

The input images to be coded in each VOP layer are of arbitrary shape, and the shape and location of the images vary over time with respect to a reference window. **For coding shape, motion and texture information in arbitrarily shaped VOPs**, the MPEG-4 Video Verification Model introduces the concept of a

VOP image window together with a **shape-adaptive Macroblock grid**. All VOL layers to be coded for a given input video sequence are defined with reference to the reference window of constant size. An example of a VOP image window within a reference window and an example of a Macroblock grid for a particular VOP image is depicted below,



This **Macroblock grid** is used for **alpha plane coding, motion estimation and compensation** as well as for block based **DCT-based texture coding**. A VOP image window with a size of multiples of 16 pels (pixels) in each image direction surrounds the foreground VOP of arbitrary shape and specifies the location of the Macroblocks, each of size 16x16 pels. This window is adjusted to collocate with the most top and most left border of the VOP. A shift parameter (control macroblock) is coded to indicate the location of the VOP image window with respect to the borders of a reference window (or original image border).

The shape information of a VOP is coded prior to coding motion vectors based on the VOP image window macroblock grid, and is available to both encoder and decoder. In subsequent processing steps only the motion and texture information for the Macroblocks belonging to the VOP image are coded (which includes the standard Macroblocks as well as the contour Macroblocks indicated in the figure above).

Shape Coding - Essentially two coding methods are supported by the MPEG-4 Video Verification Model for binary and gray scale shape information. The shape information is referred to as "alpha planes" in the context of the MPEG-4 VM. The techniques to be adopted for the standard will provide lossless coding of alpha-planes as well as the provision for lossy coding of shapes and transparency information, allowing the trade off between bit rate and the accuracy of shape representation. Furthermore it is foreseen to support both Intra shape coding as well as Inter shape coding functionalities employing motion compensated shape prediction - to allow both efficient random access operations as well as an efficient compression of shape and transparency information for diverse applications.

Motion Estimation and Compensation - The MPEG-4 VM employs block-based motion estimation and compensation techniques to efficiently explore

temporal redundancies of the video content in the separate VOP layers. In general, the motion estimation and compensation techniques used is an extension of the standard MPEG block matching techniques towards image sequences of arbitrary shape.

To perform block based motion estimation and compensation between VOPs of varying location, size and shape, the shape-adaptive Macroblock (MB) grid approach for each VOP image is employed. A block-matching procedure is used for standard Macroblocks. The prediction error is coded together with the Macroblock motion vectors used for prediction. An advanced motion compensation mode is defined which supports block-overlapping motion compensation as with the MPEG2 standard as well as the coding of motion vectors for 8x8 blocks. The motion estimation and compensation techniques are modified at the borders of a VOP. An image padding technique is used for the reference VOP frame N-1, which is available to both encoder and decoder, to perform motion estimation and compensation. The aim of the padding procedure is to allow separate decoding and reconstruction of VOPs by extrapolating texture inside the VOP to regions outside the VOP. This allows block-based DCT coding of texture across a VOP border as in INTRA VOPs well. Furthermore the block based motion vector range for search and motion compensation in a VOP in frame N can be specified covering regions outside the VOP in frame N-1. The VOP padding method can be seen as an extrapolation of pels outside of the VOP based on pels inside of the VOP. After padding the reference VOP in frame N-1, a "polygon" matching technique is employed for motion estimation and compensation. A polygon defines the part of the contour Macroblock which belongs to the active area inside of the VOP frame N to be coded and excludes the pels outside of this area. Thus, the pels not belonging to the active area in the VOP to be coded are essentially excluded from the motion estimation process. MPEG 4 has introduced 3 new MC modes (than MPEG2),

- The motion vector can have a resolution of $\frac{1}{4}$ PEL to improve the quality of the predicted pictures
- Global motion compensation MVs (4) for all MBs to describe some types of motion like panning, zooming or rotation for a entire VOP
- Direct mode in bi-directional prediction which calculates 2 new MVs from the original forward and backward MVs

Texture Coding - The Intra VOP's as well as the residual errors after motion compensated prediction are coded using a DCT on 8x8 blocks similar to the MPEG standard. Again, the adaptive VOP window Macroblock grid is employed for this purpose. For each Macroblock a maximum of four 8x8 Luminance blocks and two 8x8 Chrominance blocks are coded. Particular adaptation is required for the 8x8 blocks straddling the VOP borders. The image padding technique in the figure shown is used to fill the Macroblock content outside of a VOP prior to applying the DCT in Intra-VOPs. For the coding of motion compensated prediction error P or B-VOPs the content of the pels outside of the active VOP

area are set to 128. Alternatively a low complexity shape-adaptive DCT (SADCT) technique can be used to only encode the pixels belonging to the VOP. This results in higher quality at same bit rate at a slightly increased implementation complexity. Scanning of the DCT coefficients followed by quantization and run-length coding of the coefficients is performed using techniques and VLC tables defined with the MPEG-1/2 standards, including the provision for quantization matrices. An efficient prediction of the DC- and AC-coefficients of the DCT is performed for Intra coded VOP's.

Multiplexing of Shape, Motion and Texture Information - Basically all "tools" (DCT, motion estimation and compensation, etc.) defined in the MPEG2 Main Profile are supported by the MPEG-4 Video Verification Model. The compressed alpha plane, motion vector and DCT bit words are multiplexed into a VOL layer bitstream by coding the shape information first, followed by motion and texture coding.

- **Coding Efficiency and Compatability**

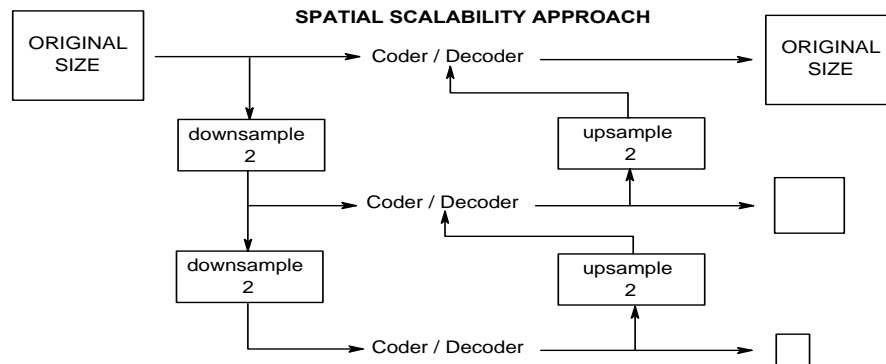
Besides the provision for new content-based functionalities and error resilience and robustness, the coding of video with very high coding efficiency over a range of bit rates continues to be supported for the MPEG-4 standard. As indicated above, the MPEG-4 Video Verification Model allows the single object-layer (single VOP) coding approach as a special case. In this coding mode the single VOP input image sequence format may not be segmented into several VOPs, and the MPEG-4 Video Verification Model coding algorithm can be made almost compatible to the ISO/IEC MPEG-1/2 standards. Most of the coding techniques used by the MPEG-2 standard at Main Profile are also supported.

- **Spatial and Temporal Scalability**

An important goal of scalable coding of video is to support receivers with different bandwidth or display capabilities or display requests to allow video database browsing and multi-resolution playback of video content in multimedia environments. Another important purpose of scalable coding is to provide a layered video bit stream which is suitable for prioritized transmission. The techniques adopted for the MPEG-4 Video Verification Model allow the "content-based" access or transmission of arbitrarily-shaped VOPs at various temporal or spatial resolutions, in contrast to the frame-based scalability approaches introduced for MPEG2. Receivers either not capable or willing to reconstruct the full resolution arbitrarily shaped VOPs can decode subsets of the layered bit stream to display the arbitrarily shaped VOPs content/objects at lower spatial or temporal resolution or with lower quality.

Spatial Scalability - The next figure depicts the MPEG-4 general philosophy of a content-based VOP multi-scale video coding scheme. Here **three layers** are provided, each layer supporting a VOP at different spatial resolution scales, i.e. a

multi-resolution representation can be achieved by downscaling the input video signal into a lower resolution video (downsampling spatially in our example). The downscaled version is encoded into a base layer bit stream with reduced bit rate. The upscaled reconstructed base layer video (upsampled spatially in our example) is used as a prediction for the coding of the original input video signal. The prediction error is encoded into an enhancement layer bit stream. If a receiver is either not capable or willing to display the full quality VOPs, downscaled VOP signals can be reconstructed by only decoding the lower layer bit streams. It is important to notice, however, that the display of the VOP at highest resolution with reduced quality is also possible by only decoding the lower bit rate base layer(s). Thus scalable coding can be used to encode content-based video with a suitable bit rate allocated to each layer in order to meet specific bandwidth requirements of transmission channels or storage media. Browsing through video data bases and transmission of video over heterogeneous networks are applications expected to benefit from this functionality.



Temporal Scalability - This technique was developed with a goal similar to spatial scalability. Different frame rates can be supported with a layered bit stream. Layering is achieved by providing a temporal prediction for the enhancement layer based on coded video from the lower layers. Using the MPEG-4 "content-based" VOP temporal scalability approach it is possible to provide different display rates for different VOLs within the same video sequence (i.e. a foreground person of interest may be displayed with a higher frame rate compared to the remaining background or other objects).

- **Resilience to Errors**

MPEG-4 error-resiliency tools were designed to transport compressed video over error-prone networks (mobile channels) with acceptable quality. They are,

- Flexible re-synchronization markers in the bitstream to detect and localize errors within the encoded stream
- Data partitioning to separate shape, texture, motion and header information, thus organizing a stream to make the decoding process more robust to errors

- Header protection to provide additional redundancy for the most important header fields
- Reversible variable length coding (RVLC) for texture information. Since these codes are decodable in the forward and reverse directions, burst errors at the start of a packet does not render the remaining encoded data in that packet un-usable. Reverse variable length decoding can be used to recover these data coefficients.
- Long predictive sequences (P/B VOPs) cause the propagation and persistence of errors to increase from VOP to VOP. Forced intra-frame refresh can lessen this effect. Thus intra-coded macro-blocks are inserted within predicted VOPs, to reduce the persistence of errors.

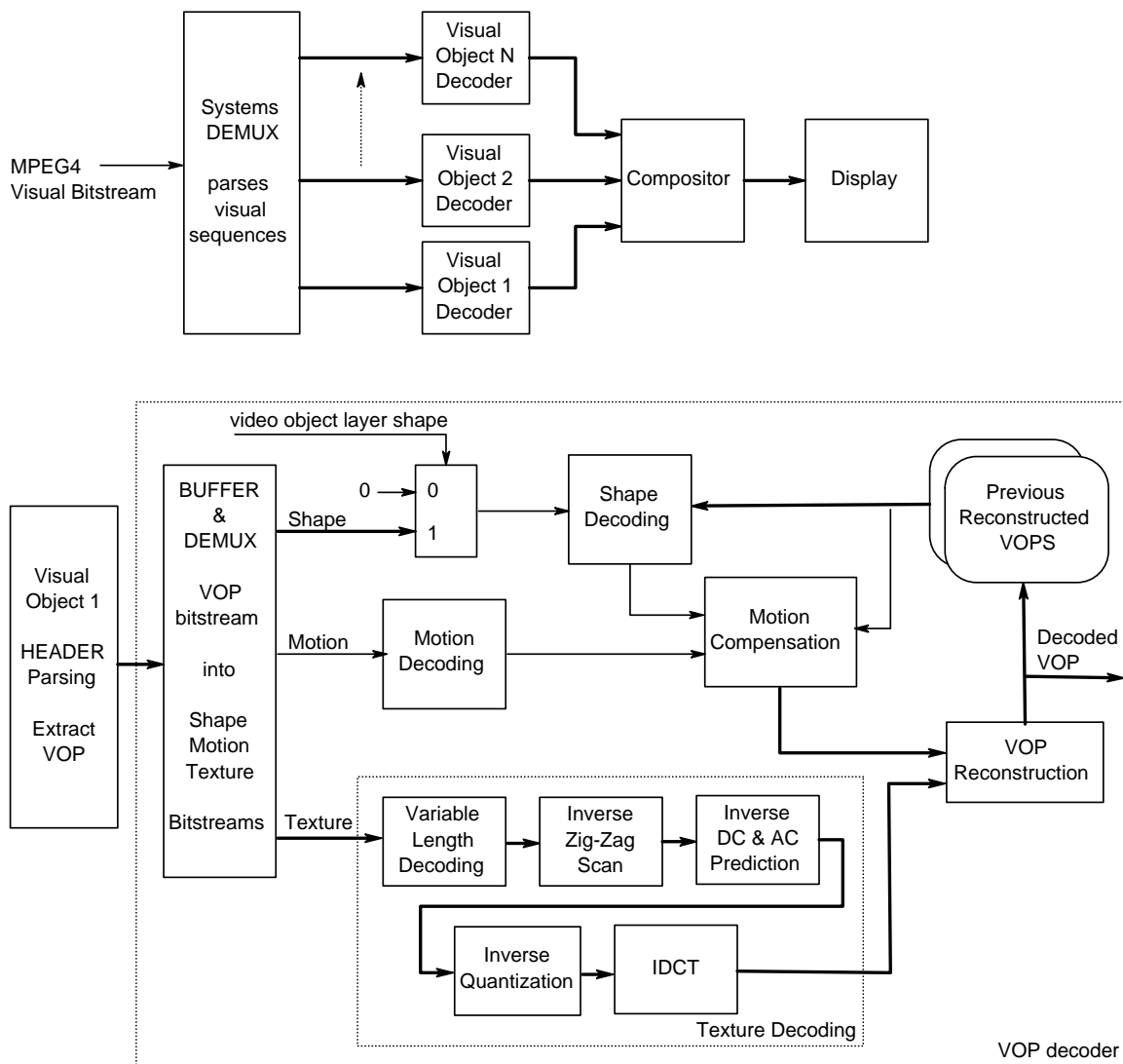


Figure 70. MPEG4 VOP decoder

The MPEG4 VOP decoder does the following,

- The MPEG4 visual bit-stream consisting of a sequence of visual objects (videos, meshes, FBA-fixed body animations and still textures) is parsed by the visual systems de-multiplexer. It basically parses the visual object sequence header (VS header) and de-multiplexes the visual bit-stream into individual visual objects and sends them to their respective visual object decoders.
- The visual object decoder parses the visual object header (VO header) and the visual object layer header (VOL header). Each visual object could have encoded a number of layers depending on different bitrates and resolutions for that visual object and all these layers have to be separated. The group of video planes header (GOV) is then parsed and the video object planes (VOPs) are extracted and sent to the **VOP decoder**.
- In the VOP decoder the incoming VOPs are buffered (similar to the bit-buffering mechanism found in MPEG1/2), de-multiplexed into their 3 parts (shape, motion and texture bitstreams) and then sent to the shape, motion and texture decoding blocks.

Notes:

1. Motion estimation in the video encoder generates (estimates) 2D motion vectors which provide offsets (macroblock resolution) from the coordinate position in the current frame or field to the coordinates in a reference frame or field.
2. Motion compensation in the video decoder uses these same motion vectors to provide offsets into the past and/or future reference frames or fields containing previously decoded macroblocks which are then used to calculate the prediction error (used in the reconstruction of P/B pictures by the decoder).
3. Motion vectors for 4:2:0 chroma blocks are derived by scaling motion vectors for luma blocks by a factor of 2 in horizontal and vertical direction.
4. I pictures (used as an anchor frame/field) are coded with information only within itself, thus no motion compensation is needed to reconstruct this frame or field. P pictures (can be used as an anchor frame) are coded using motion compensated prediction from the past reference frame or field. B pictures (never used as anchor frames) are coded using motion compensated prediction from past and future reference frames or fields.
5. In MPEG4 the motion vector is 1st calculated with a resolution of $\frac{1}{2}$ pel using a 8 tap FIR filter and further scaled to $\frac{1}{4}$ pel resolution using bilinear interpolation. In MPEG2 bilinear interpolation is used to calculate the $\frac{1}{4}$ pel motion vectors.
6. MPEG4 has introduced 2 additional scan matrices (apart from the standard zigzag scan matrix) to convert (re-arrange) the 2D matrix of coefficients (after 8x8 block DCT) into a 1D matrix of coefficients (with long zero runs) before quantization.

7. In MPEG2 it is possible to have frame/field sequences, thus you could have a sequence with progressive pictures followed by a sequence with interlaced pictures. Also the encoder could choose to send a separate set of picture headers (with coding and display extensions) before every field of picture data (in field mode). The decoder has to extract these picture headers twice (1 for each field in a frame) before decoding/displaying each field.
8. Still image object coding in MPEG4 uses the wavelet transform.
9. DIVX and Windows Media Player 9 both support MPEG4 encoding and decoding.
10. A combination of run-length encoding & variable length coding (Huffman codes) is used to encode and deliver internet web pages (HTML). These web pages are then decoded and displayed at the user's terminal/computer giving improved web browsing rates.

PACKETIZATION of AUDIO, VIDEO & DATA for DELIVERY over NETWORKS

Packetization of a compressed/un-compressed baseband bit-stream involves the encapsulation of different parts within that bitstream (using de-lineating headers) for achieving software compatibility with different network/transport layers in a transmission/reception or storage environment. Packetization is used in encapsulating (and formatting) these bitstreams before transmission over media or storage channels which have standardized software interfaces (software stacks) for seamlessly interfacing transmitters and receivers. The process of packetization is also needed to synchronize transmitters and receivers by adding synchronization headers (time-stamps) into the transmitted baseband audio/video/data samples, which can then be extracted by a receiver or a group of receivers to sample/parse the baseband information correctly (especially if the transmission channels are noise or error prone or are time variant - jitter).

Below, we have shown the slot in which packetization as a whole fits into the general gamut of transmission/storage of baseband audio, video and data.

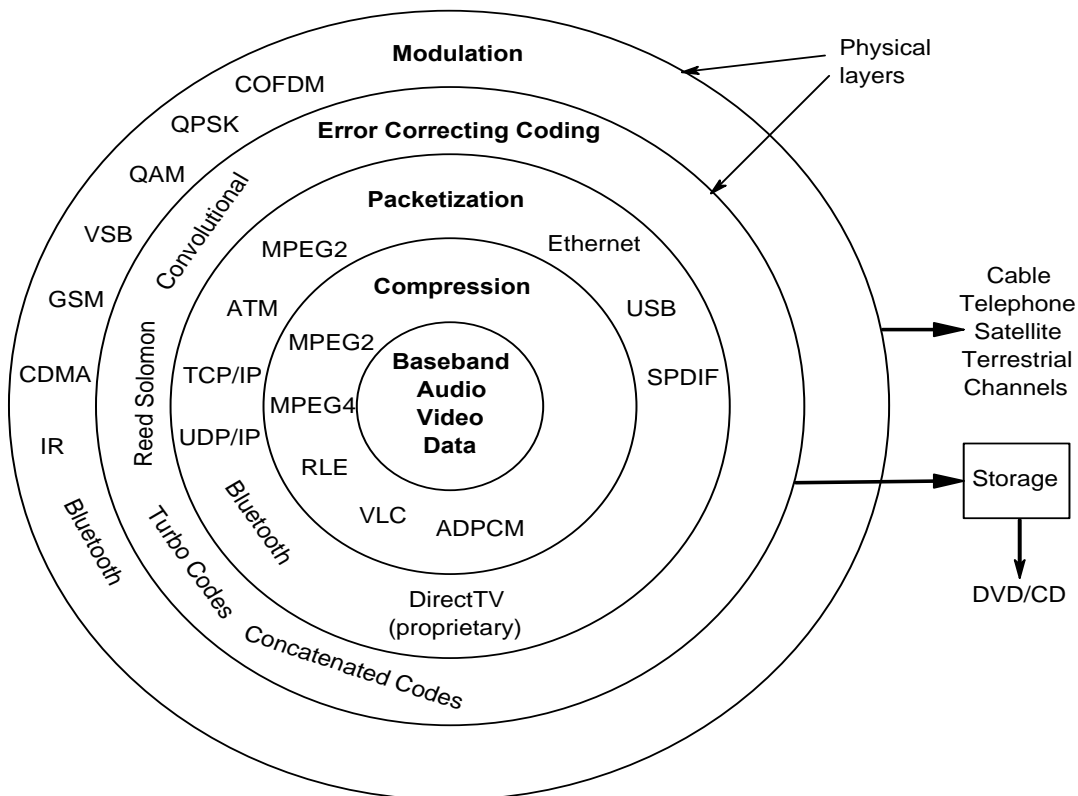


Figure 71. Typical Packetization Slot

The typical **Packetization Schemes** discussed are listed below,

- **MPEG2 Packetization as applicable in DVB and DVD**
- **MPEG2 Packetized Elementary Streams (PES)**
- **MPEG2 Transport Streams (Satellite/Cable/Terrestrial SETTOP box)**
- **ATM Encapsulation of MPEG2 Transport (IP, IPX, Ethernet Networks)**
- **Electronic Program Guide - DVB tables**
- **MPEG2 program streams (DVD)**
- **Navigation Guide (DVD)**
- **MPEG4 Access Unit, FlexMux and TransMux Layers**
- **DOCSIS Downstream and Upstream Channel MAC Packetization**
- **IEEE 802.3 (Ethernet) Packetization**
- **Internet Protocol (IP) Packetization**
- **USB Packetization**

MPEG2 Packetization (ISO/IEC 13818 systems & ETSI ETS 300468)

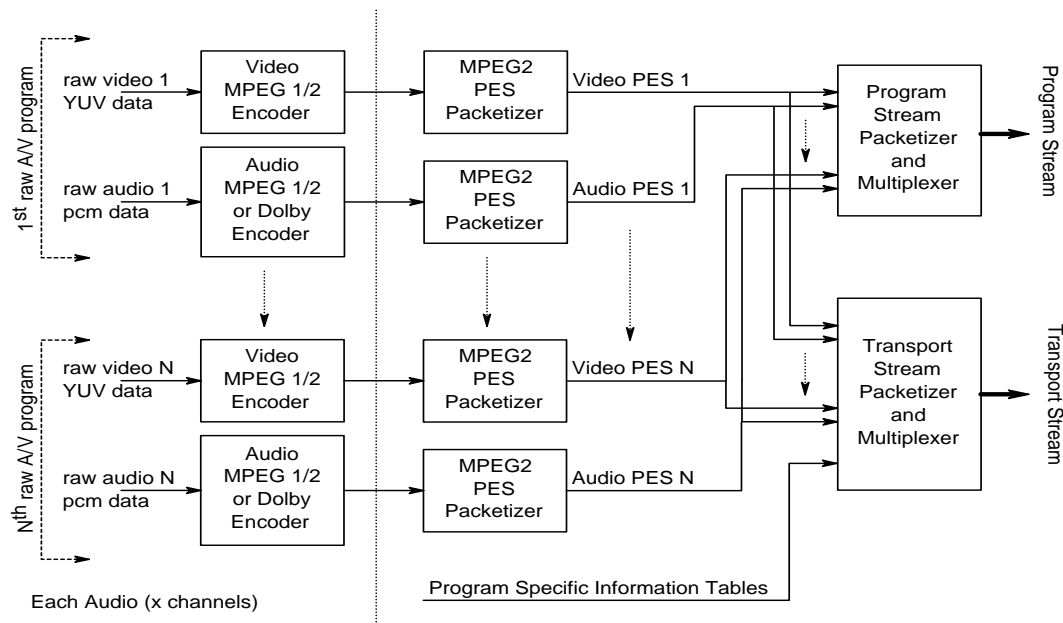


Figure 72. MPEG2 Packetization (system layer)

- **MPEG2 packetized elementary streams (PES)**

As shown in Figure 72, PES packetization (taking the encoded A/V data and forming PES packets) of encoded A/V streams is done before multiplexing and packetizing them into transport or program streams. Since PES packets are the intermediate data exchange format between program and transport

streams, conversion between these 2 layers (TS and PS) is greatly simplified. **A PES stream is a continuous sequence of PES packets of one elementary stream (one and only one encoded audio or video stream) having the same stream ID.** In order to decode PES streams independently (outside their program or transport layer), an elementary stream clock reference (ESCR) field and an elementary stream rate (ES Rate) field is added into the PES packets. Also PES packets can be of fixed and variable lengths.

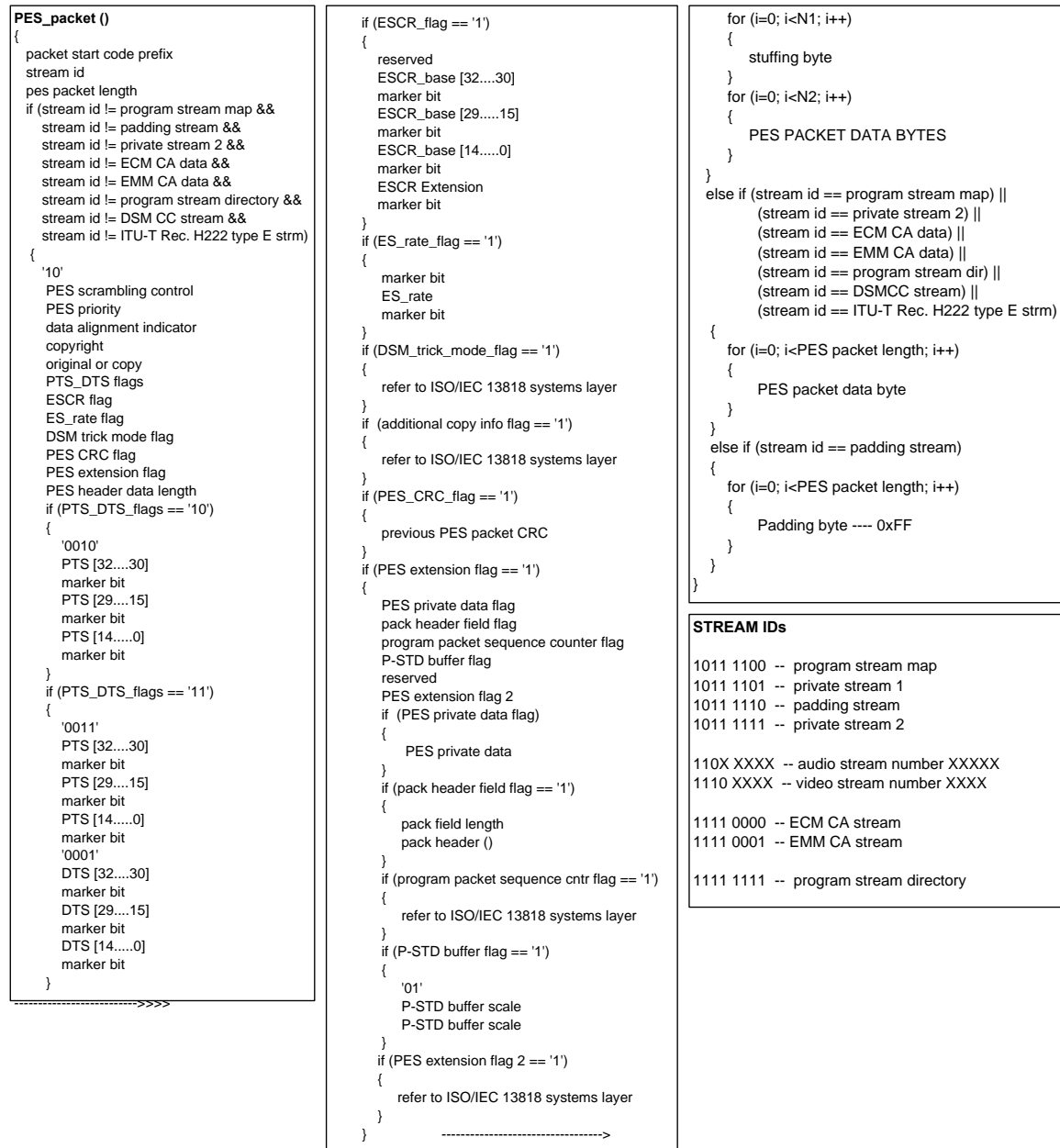
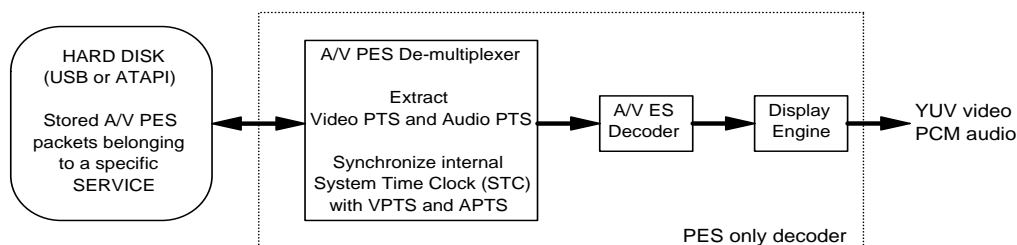


Figure 73. PES packet

The important fields which PES packets carry,

1. **Packet Start Code Prefix** - is a 24 bit code 0x000001. Together with the stream id, this packet start code identifies the beginning of a PES packet.
2. **8 bit Stream ID** - in program streams the stream id specifies the type and number of the elementary stream as defined by the stream id table in figure 73. In a transport stream the stream id is set to any valid value which correctly describes the stream type (since in a transport stream the PID supercedes the stream id).
3. **PES Scrambling bits (Scrambling of PES payload – ES)** – these 2 bits indicate the scrambling mode of the PES payload (ES stream) when scrambling is performed at the PES level (00 – not scrambled, 01-11 – user defined).
4. **PTS_DTS flags** – when set to '10' only the PTS is present in the PES header. When set to '11' both the PTS and the DTS are present in the PES header. When set to '00' the PTS and DTS are not present in the PES header.
5. **PTS (Presentation Time Stamp)** – is a 33 bit number coded in units of 90 KHz (system clock / 300). It indicates the time of presentation, $T_{p_n}(k)$, of the access unit k of elementary stream n at the decoder. $PTS(k) = ((\text{system clock frequency} \times T_{p_n}(k))/300)$, where $T_{p_n}(k)$ is the presentation time of presentation unit $P_n(k)$. In the case of audio, if a PTS is present in the PES header, $P_n(k)$ refers to the 1st audio access unit (1st decoded audio frame) in the PES packet. In the case of video, if a PTS is present in the PES header, it refers to the 1st video access unit (frame or field) containing the 1st picture start code in the PES packet.
6. **DTS (Decoding Time Stamp)** – is a 33 bit number coded in units of 90 KHz. It indicates the decoding time, $T_{d_n}(k)$, of an access unit k of the elementary stream n at the decoder. $DTS(k) = ((\text{system clock frequency} \times T_{d_n}(k))/300)$. Basically it shows when an A/V access unit present in that PES packet is to be decoded.
7. **ESCR (Elementary Stream Clock Reference)** – is a 42 bit field found in 2 fields (ESCR base and ESCR extension). The ESCR base is a 33 bit field and the ESCR extension is a 9 bit field. The ESCR base is clocked using the 27MHz system clock and the ESCR extension is clocked using the system clock/300. Thus $ESCR = ESCR \text{ base} \times 300 + ESCR \text{ extension}$. This field indicates the intended time of arrival of the byte containing the last bit of the ESCR base at the input of the PES stream decoder.
8. **ES Stream Rate (Elementary Stream Rate)** – is a 22 bit unsigned integer which specifies the rate at which the PES decoder receives bytes of the PES packet. The value of the ES rate is measured in units of 50 bytes/sec and defines the time of arrival of bytes in the PES packet at the input of the PES decoder.
9. **Pack header field flag** – this indicates the presence of a **program stream Pack Header** in the PES packet header.
10. **Program stream buffering (P_STD buffer size)** – is a 13 bit integer which is defined only if the PES packet is contained in a program stream. It defines the size of the input buffering, BS_n , in the program stream decoder for decoding audio or video PES packets contained within program streams.
11. **Presence of ECM/EMM Conditional Access data** – if the stream id is either the EMM or the ECM stream id, then the entire PES header is ignored and the PES packet takes the form, **0x000001 + stream id + PES packet length + EMM or ECM data bytes**.



In a decoder which decodes PES streams belonging to a service the goal is to keep the video and audio decode in sync (maintain lip-sync for the user) during decode. Thus the 1st operation is to extract the PTS from the 1st received video PES and write it into an internal 32 bit counter in the decoder, which is then incremented by a 90 KHz free running clock. Then each subsequent extracted PTS from the received video or an audio PES packets is used to keep track of the synchronization between audio and video, by doing an **(STC-VPTS)** and **(STC-APTS)** and skipping or repeating audio frames or video fields/frames if the differences cross a threshold (usually half a audio/video frame).

- **MPEG2 Transport Streams (TS)**

The transport stream combines one or more programs with one or more independent time bases into a single stream. Basically you could have several A/V programs (A/V PES packets belonging to separate programs), in a transport stream with each A/V program encoded on a different time base. This is typical of the transport multiplex where several hundred programs (a program could be a combination of video, audio and data channels) are multiplexed together along with their program specific information (information about the all multiplexed programs – a.k.a. information to form a electronic program guide or EPG). A MPEG2 transport stream is typically used in error prone transmission environments and each of its packets is 188 bytes. The transport stream can be used to span a range of layers in a networked environment and is ideally suited for high bandwidth applications.

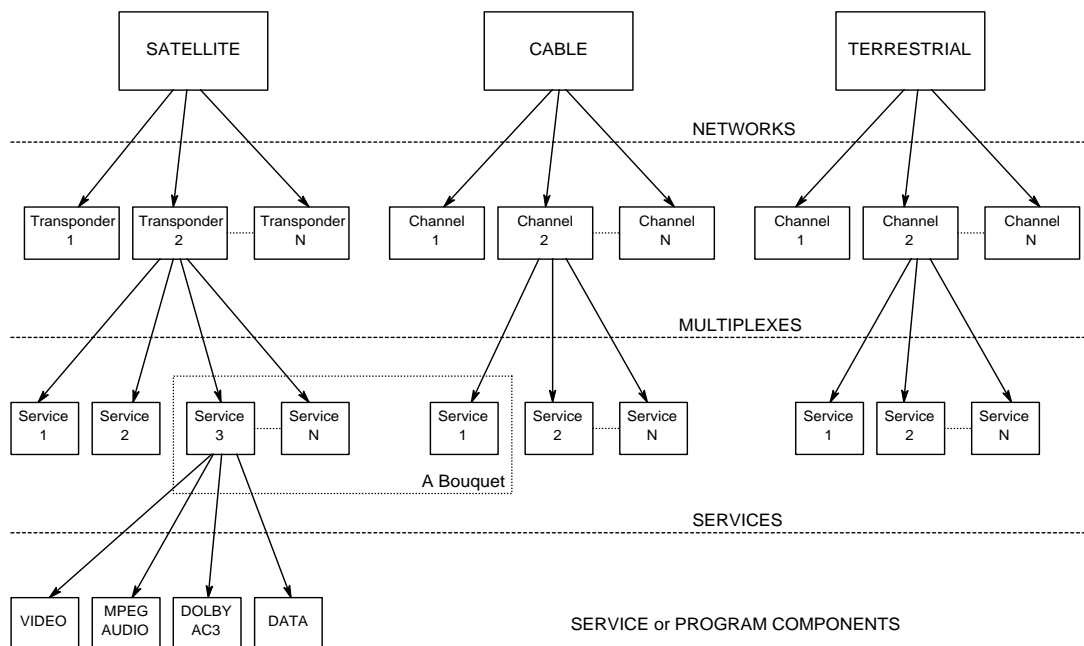


Figure 74. Service Delivery Model in Digital Video Broadcasting (DVB)

Transport streams can have fixed or variable bit rates with fixed or variable bit rate elementary A/V streams within them. The transport stream rate is given by the Program Clock Reference (PCR) field/s in the TS which is extracted by the transport de-multiplexer to adjust its internal clock to receive TS data.

The system clock (27 MHz program clock reference timestamp) provides all the timing information in a transport stream. The timing model is defined as the **end to end delay** from the signal input to the encoder to the signal output from the decoder. This delay is basically the sum of the encoding + encoder buffering + multiplexing + *transmission or storage* + de-multiplexing + decoder buffering + decoding + presentation delays. The system clock timestamp (PCR) provides a good reference which allows all decoders on the transport multiplex to decode and present the access units (video/audio/data) encoded exactly once at the same time.

A Transport stream de-multiplexer does the following tasks,

1. For a particular encoded service, extracts the PCR encoded for that service and adjusts/synchronizes its internal system time clock (STC) to this reference PCR timestamp.
2. Extracts the program specific information (PSI) from the DVB MPEG2 tables (split into sections) shown below and builds up a database (or electronic program guide) which displays each and every service (along with its associated audio, video and data contents) in the transport multiplex in a user friendly graphical fashion.

PAT Program Association Table	PMT Program Map Table	CAT Conditional Access Table	NIT Network Information Table
BAT Bouquet Association Table	SDT Service Description Table	EIT Event Information Table	TDT Time and Date Table
RST Running Status Table	TOT Time Offset Table	ST Stuffing Bytes Table	

1. **PAT** – for each service in the multiplex the PAT indicates the location (or PID values) of the corresponding Program Map Table (PMT). It also gives the location of the Network Information Table (NIT)
2. **PMT** – this identifies and indicates the locations (or PID values) of the streams that make up each service, and the location of the Program Clock Reference fields for each service
3. **CAT** – this provides information on the Conditional Access (CA) systems used in the multiplex. This information is private and dependant on the CA system. This includes the location (PID) of the EMM (Entitlement management message) stream
4. **NIT** – this provides information about the physical network (satellite, cable or terrestrial)
5. **BAT** – this provides information about bouquets. It not only provides the name of the bouquet, but provides a list of services for each bouquet
6. **SDT** – this contains information describing the services in the system (e.g., names of services, the service provider, etc.)
7. **EIT** – this contains information describing the events (or programs) such as event name, start time, duration, etc. Using different event descriptors, event information for different services types can be transmitted
8. **TDT** – gives periodic information related to the present time and date
9. **RST** – gives the status of an event (is the program running/not running). This information is updated in the RST periodically allowing timely (and automatic) switching to events/programs
10. **TOT** – gives periodic information relating to the present date/time time and the local time offset
11. **ST** – is used to invalidate existing sections at delivery system boundaries

3. Extracts the audio/video/data PES payloads of the service, descrambles them (if scrambled at the transport layer) based on the keys provided in the EMM/ECM sections and delivers them to the A/V decoders.
4. If the transport multiplex is **not free to air** (if service components are scrambled at the transport layer) and scrambled using a proprietary conditional access system, then the transport de-multiplexer has to filter the EMM/ECM (entitlement management & entitlement control messages) sections, extract the encrypted control words (or keys) from the ECMs, decrypt them using a proprietary algorithm (some form of asymmetric algorithm like RSA using public/private keys) residing on a smartcard and then use these keys to descramble the PES A/V/data payloads (descrambling of the PES payload is achieved via a built in symmetric descrambler like AES, DES, Triple DES, DVB descrambling algorithms which use a combination of stream and block cipher techniques).

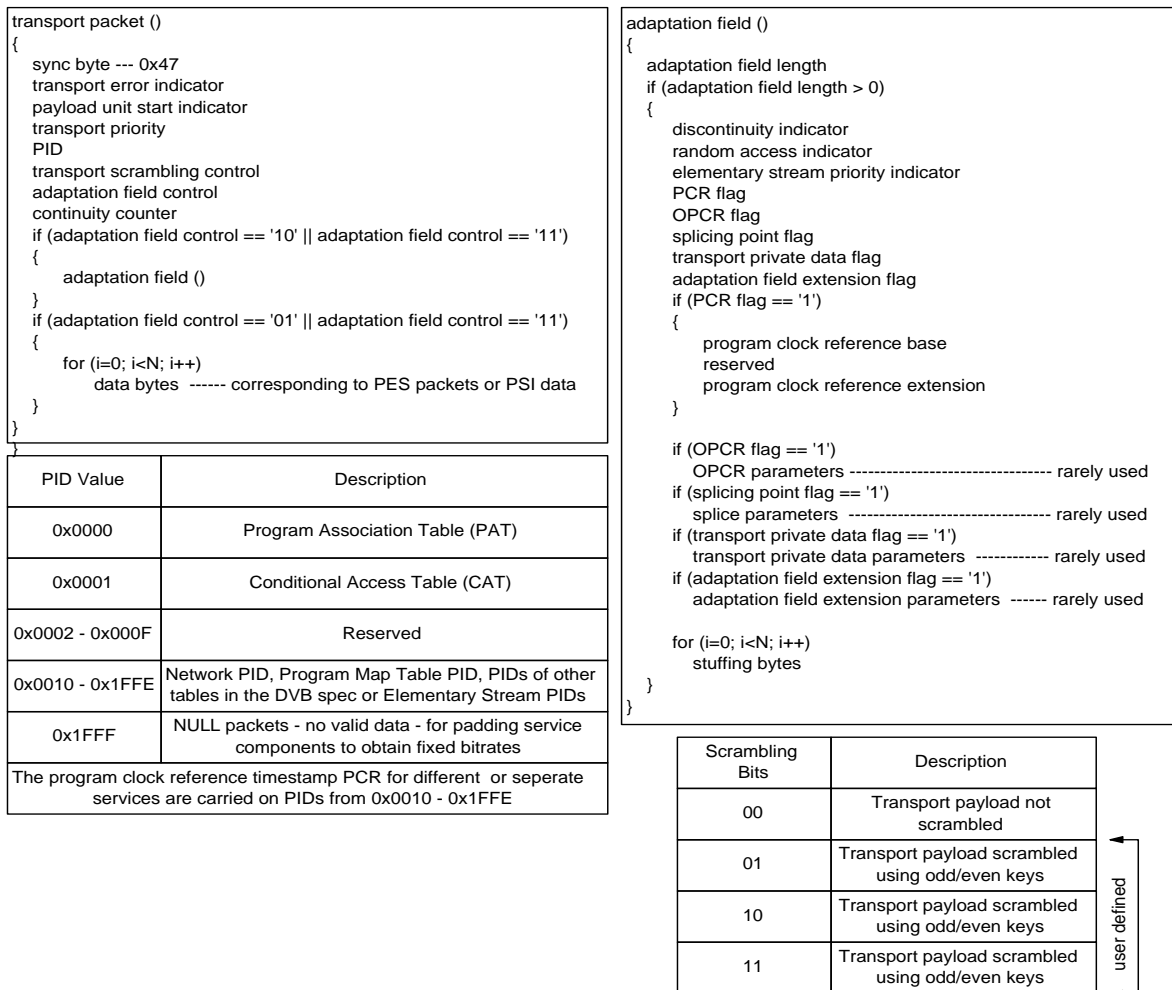


Figure 75. Transport packet and its Adaptation Field (if present)

The important fields in a transport packet and its associated adaptation field if present are shown below,

1. **Sync Byte** - is 0x47 for MPEG2 transport streams. This is the delineating byte between 2 transport packets.
2. **Transport Error Indicator** – this tells a decoder if the transport packet is erroneous or uncorrected. This is usually set by the block outer decoder in a FEC if it cannot correct the transport packet.
3. **Payload Unit Start Indicator** – If this flag is set when the PID in a transport packet belongs to an elementary A/V stream, then the start of the PES packet (PES header) belonging to that elementary stream is in the transport packet. If this flag is set when the PID belongs to a PSI table, then the first byte of that PSI section is in the transport packet (the index of this 1st byte of PSI data is given by an **8 bit pointer field** which is the **1st byte in the data bytes of the transport header → pointer field + 4**).
4. **PID** – this 13 bit value is the unique program or elementary stream identifier. Every elementary stream and PSI section (standard and private tables) have their own unique PID.
5. **Transport Scrambling Control** – these bits determine if the transport payload (PES packets are scrambled or un-scrambled). If scrambled then they have to be de-scrambled (using a Conditional Access Subsystem) before sending them along to the PES/ES decoders. The standard PSI sections are never scrambled (however the private tables can be scrambled).
6. **Adaptation Field** – if this field is present, then it usually contains the PCR base and PCR extension (total of 42 bits) which is used as the periodic Time-Stamp in the clock recovery mechanism to adjust the transport decoder's internal system clock. This Time-Stamp basically adjusts the decoder's system clock to the encoder's system clock, such that the decode mechanism is synchronized to the encode mechanism.

$$\text{PCR} = (\text{PCR base} * 300) + \text{PCR extension}$$

7. **Continuity Counter** – this is a 4 bit number which goes from 0 to 15 and is unique to a transport packet. This number gives an indication as to whether any transport packets are lost before entering the transport decoder (indicates discontinuities). If this number repeats itself in 2 consecutive transport packets, then those 2 transport packets carry identical data (basically the second one has to be discarded by the transport demux). Also if there is a discontinuity the transport de-multiplexer can add a tag in the PES packet or PSI sections to inform/allow the following PES or PSI decode stages to re-sync themselves.

- **ATM encapsulation for MPEG2 transport packets**

MPEG2 transport packets can be encapsulated into ATM AAL5 layer **SDUs (Service Data Units)** and transmitted over an ATM network using routed or bridged protocols (IP, IPX and Ethernet protocols).

Shown next is how data packets of higher protocol layers (in this case **MPEG2 transport data**) are encapsulated in the lower AAL5 layer payload or Service Data Unit (SDU) fields. The data passed to the lower layer is called the Protocol Data Unit (PDU), because it contains all the protocol information in the header necessary for the protocol entity to do its work. PDUs of the upper layers are passed down as SDUs of the lower layers. Thus, no signalling is needed, just the encapsulation of packets over an already established connection. Thus transport streams can be encapsulated and transmitted as ATM cells over an ATM (Asynchronous Transfer Mode) transport network using almost any connectionless protocol.

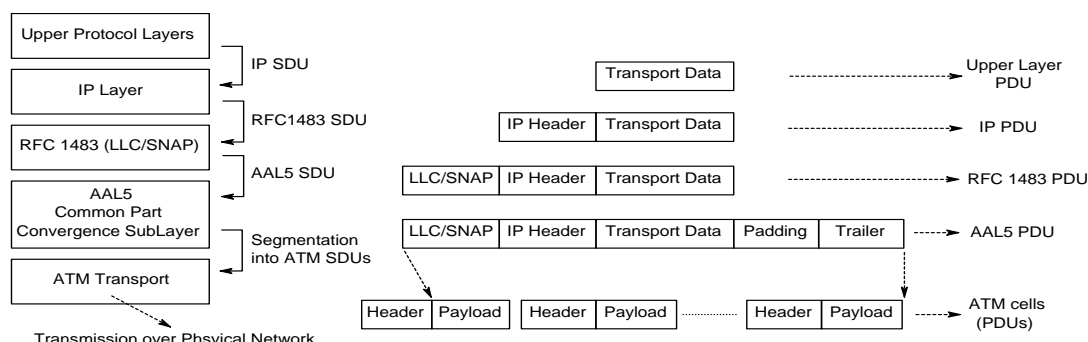


Figure 76. Transport packet encapsulation in a AAL5 ATM layer

• Building a Electronic Program Guide using DVB tables

The PSI tables are segmented into one or more sections before being inserted into the transport stream packets. These tables are never scrambled with the exception of the EIT and any private tables (carrying private user data), which can be scrambled. The main purpose of these tables is to build a user friendly electronic program guide (graphical database representing programs and their contents available for viewing at present and in the near future). The tables are divided into sections which are 1024 bytes in length, except for sections within an EIT or private tables which can have a length of 4096 bytes. The PSI tables have the following parts,

1. **Table id** – this identifies the table to which a PSI section belongs. Some table ids are defined and some belong to a specific service.
2. **Section number** – this allows the decoder to reassemble all the sections of a particular table in their original order.
3. **Version number** – when characteristics of the components in a transport stream change (e.g. new events start, composition of service changes, etc.), new PSI data is sent containing the updated information. A new version of PSI data is signaled by sending a section with the same identifiers as the previous section but with the next value of the version number.
4. **Current Next Indicator** – each section is numbered as valid “now” or valid in the immediate “future”. This allows the transmission of a future version of a PSI section, such that the decoder can prepare itself for the future change.

<pre> program association section () { table id section syntax indicator '0' reserved section length transport stream id reserved version number current next indicator section number last section number for (i=0; i<N; i++) { program number reserved if (program number == '0') network_PID else program_map_PID } CRC_32 } </pre>	<pre> conditional access section () { table id section syntax indicator '0' reserved section length reserved version number current next indicator section number last section number for (i=0; i<N; i++) { descriptor () } CRC_32 } </pre>	<pre> program map section () { table id section syntax indicator '0' reserved section length program number reserved version number current next indicator section number last section number reserved PCR_PID reserved program info length for (i=0; i<N; i++) { descriptor () } stream type reserved elementary_PID reserved ES info length for (j=0; j<N2; j++) { descriptor () } CRC_32 } </pre>	<pre> private section () { table id section syntax indicator private indicator reserved private section length if (section syntax indicator == '0') { for (i=0; i<N; i++) private data byte } else { table id extension reserved version number current next indicator section number last section number for (i=0; i<private_section_length-9; i++) private data byte } CRC_32 } </pre>
---	--	--	--

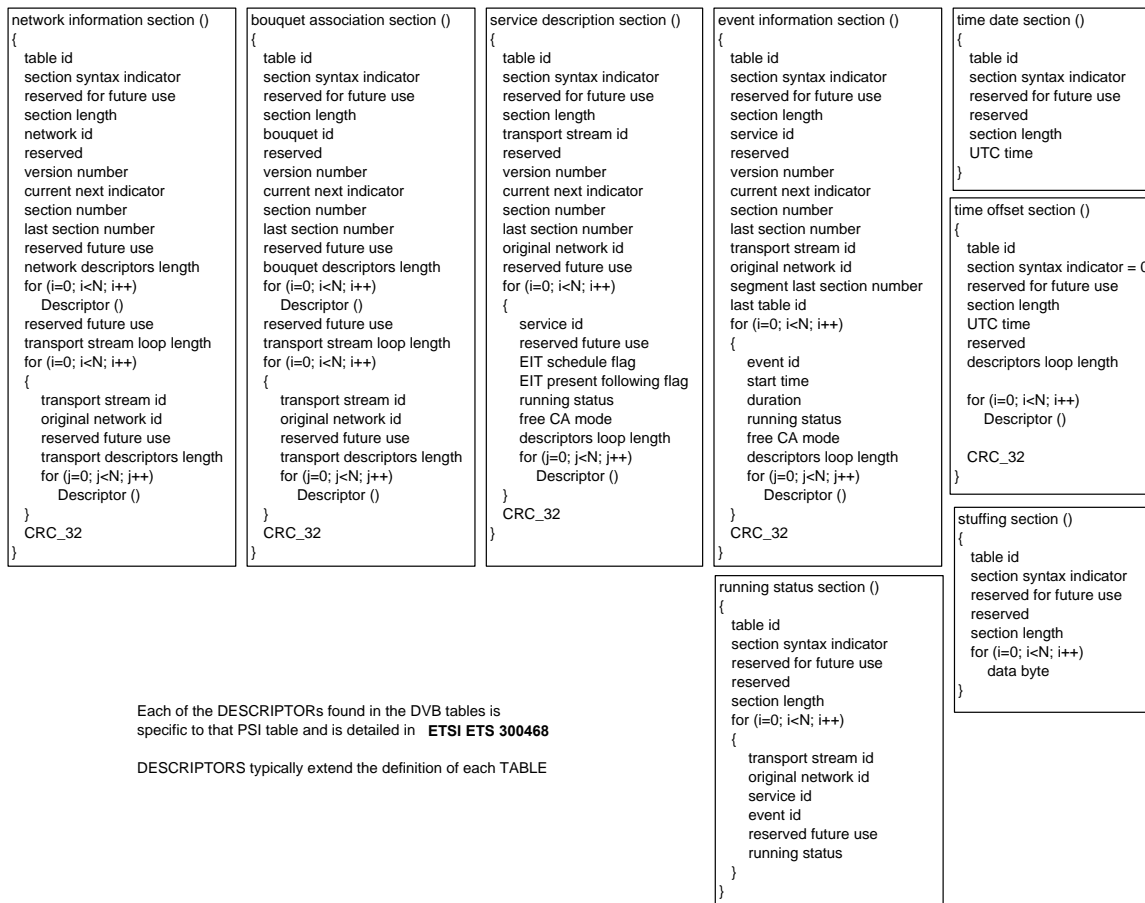
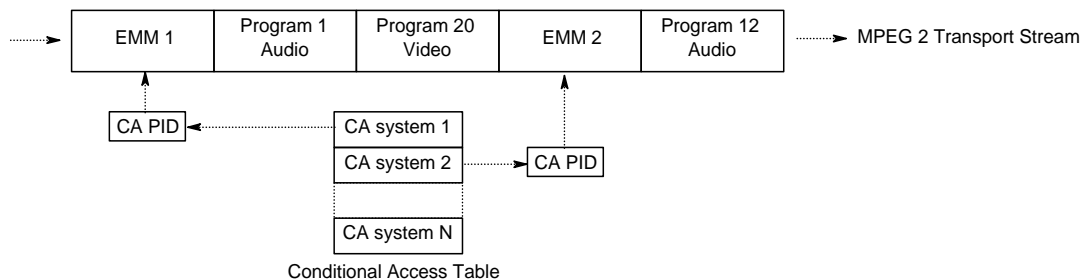


Figure 77. DVB tables in MPEG2 packetization

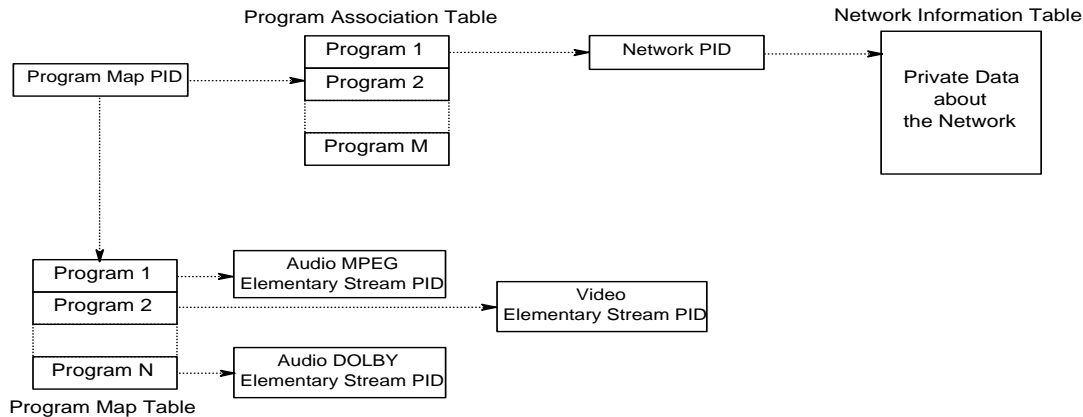
1. The program association section (PAT) provides the correspondence between a program number and the PID value of the transport stream packets which carry the program definition. The program number is the numeric label associated with a program. This table can be contained in one or more sections.
2. The conditional access table (CAT) provides the association between one or more conditional access systems, their EMM (entitlement management message) streams and any special parameters associated with them. This table can be contained in one or more sections.



3. The program map table (PMT) provides the mappings between program numbers and the program elements that comprise them. A single instance of this mapping is referred to as a

“program definition”. The PID for this table is selected by the encoder. This table can occupy one or more sections.

4. When private data sections are to be sent in the transport stream, they are usually sent with a table id which is a designated private table id/s. The private section allows data to be transmitted with a minimum section structure if the section syntax indicator = 0. The rest of the data is private data bytes and could be anything (video/audio/data related or un-related to any standard).



5. The network information table (NIT) conveys information relating to the physical organization of the multiplexes/transport streams carried via a given network, and the characteristics of the network itself. The network information table typically contains all the transponder frequencies present in a satellite system or all the channel frequencies in a cable TV system. This information can be stored in non-volatile memory after parsing the NIT, by the decoder, to enable fast switching between transponders or channels. This table can be segmented into one or more sections.
6. The bouquet association table (BAT) provides information about bouquets. A bouquet is typically a collection of services which can traverse the boundary of a network. A bouquet may be collection of X services from a satellite network + Y services from a cable network. This table can be segmented into one or more sections.
7. The service description table (SDT) describes services that are contained within a particular transport stream using different service ids (which are basically the **program number** in the corresponding PMT). This table can have one or more sections
8. The event information table (EIT) can be split into 2 types, **present/following event information table** and **event schedule information table**. The present/following table for a transport multiplex contains information pertaining to the present event and the chronologically following event in that service. The event schedule table for a transport multiplex contains a list of events, in the form of a schedule which includes current events in time and events going to happen in the near chronological future (future is usually 12 hours). The EIT can have one or more sections.
9. The time and date table (TDT) carries only the current Coordinated Universal Time (UTC) and Date information. This table is in a single section.
10. The time offset table (TOT) carries the UTC time and date information and also the local time offset. This table is in a single section.

11. The running status table (RST) allows accurate and rapid updating of the timing status of one or more events. This may become necessary if the event/s (movie or news) starts early or late due to scheduling changes or delays. This table can have one or more sections.
12. The stuffing table (ST) is used to invalidate existing sections at delivery system boundaries. If one section of a table is overwritten, then all the sections of that table are also overwritten (or stuffed) in order to retain the integrity of the section number field.

Please refer to the ETSI ETS 300468 spec for information on all the **descriptor fields** found in each table - these descriptor fields are essential in forming/presenting an EPG.

• MPEG2 program streams (DVD)

Program streams result from combining **one or more streams of PES packets which have a common time base into a single stream**. Program streams are tailored for environments (transmission or storage media) where data errors are unlikely and are suitable for applications which involve software processing of system information such as interactive multi-media applications or DVDs. These streams form the base layer of DVD media.

These streams can be fixed or variable rate and the packetized elementary streams (PES packets) carried within them can also be fixed or variable rate. The program stream rate is defined by the SCR (system clock reference) and the mux_rate fields. All timing within a program stream is defined by the system time clock (STC) whose timestamp (SCR) is encoded within a program stream. This clock usually has an exact ratio (or a slightly different frequency which differs from the exact ratio) to the video/audio sample clocks and provides end to end timing and clock recovery.

In a program stream PES packets are organized in packs. A pack starts with a pack header and is followed by zero or more PES packets. The pack header is used to store timing and bit-rate information related to the program stream.

<pre> program stream () { do { pack () } while (nextbits () == pack start code) MPEG program end code } </pre>	<pre> pack () { pack header () while (nextbits () == packet start code prefix) { PES packet () } } </pre>	<pre> pack header () { pack start code '01' system clock reference base [32..30] marker bit system clock reference base [29..15] marker bit system clock reference base [14..0] marker bit system clock reference extension marker bit program mux rate marker bit marker bit reserved pack stuffing length for (i=0; i<pack stuffing length; i++) stuffing byte if (nextbits() == system header start code) system header () } </pre>	<pre> system header () { system header start code header length marker bit rate bound marker bit audio bound fixed flag CSPS flag system audio lock flag system video lock flag marker bit video bound packet rate restriction flag reserved bits while (nextbits () == '1') { stream id '11' P-STD buffer bound scale P-STD buffer size bound } } </pre>
---	---	---	---

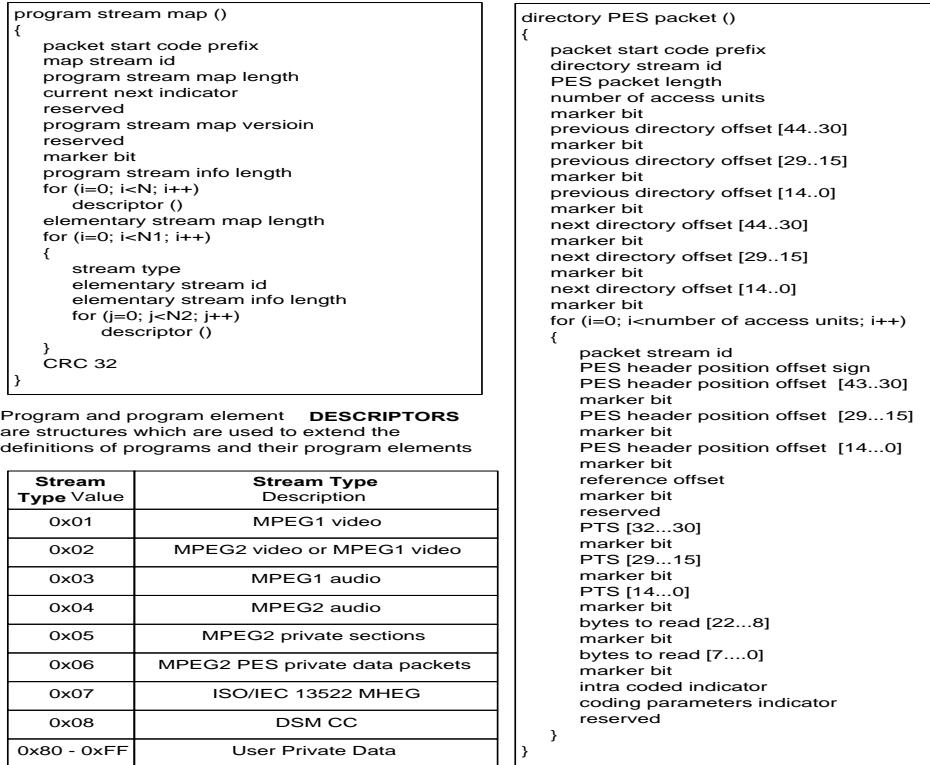


Figure 78. MPEG2 Program Stream Structure

Some of the important fields in the program stream syntax are given below,

The **program end code** is the bit string 0x000001B9 which terminates the program stream. The **pack start code** is 0x000001BA and identifies the start of a pack. The **system header start code** is 0x000001BB and identifies the start of the system header.

1. Pack Header

System clock reference – this is a 42 bit field coded in 2 parts. The first part is a 33 bit system clock reference base and the second part is a 9 bit system clock reference extension.

$$SCR = SCR \text{ base} * 300 + SCR \text{ extension}$$

Program mux rate – is a 22 bit integer specifying the rate at which the P-STD (program stream theoretical decoder) receives the program stream during the pack which includes this field.

2. System Header

Rate bound – is a 22 bit integer with a value greater than or equal to the maximum value of the program mux rate which is encoded in the pack header. It can be used by the decoder to assess whether it is capable of decoding the program stream.

Fixed flag – when set to 1 indicates a fixed bit-rate program stream, and when set to 0 indicates a variable bit-rate program stream.

System audio lock flag – when set to 1 indicates a fixed constant relationship between the audio sampling rate (for all audio elementary streams in the program stream) and the system clock reference. Basically (system clock frequency / audio sampling rate) in the theoretical P-STD decoder is a constant.

System video lock flag – when set to 1 indicates a fixed constant relationship between the video frame rate (for all video streams in the program stream) and the system clock frequency. Basically (system clock frequency / video frame rate) in the theoretical P-STD decoder is a constant.

Stream ID – all the stream id definitions are shown in **figure 73**. The following P-STD buffer fields are for the elementary stream with this particular stream id.

P-STD buffer bound scale flag – this indicates the scaling factor used to interpret the subsequent P-STD buffer size.

P-STD buffer size bound – is a 13 bit integer which defines a value greater than or equal to the maximum P-STD buffer size (this is also the maximum buffer size needed to decode a elementary stream within a program stream correctly) given by the following scaling formula,

If (P-STD buffer bound scale flag == 0)

P-STD buffer size \leq P-STD buffer size bound * 128

Else

P-STD buffer size \leq P-STD buffer size bound * 1024

3. **Program Stream Map** – provides a description of the all the elementary streams (in PES packets) in the program stream and their relationship to each other.

Packet start code prefix + Map stream ID – this has a value equal to 0x000001BC.

Program stream map length – indicates the total number of bytes in the program stream map immediately following this field. Its maximum value is 1018.

Program stream map version – indicates the version number of the program stream map. It is incremented whenever the program stream map's definition changes. If '0' then the next program stream map is to be used by the decoder.

Program stream info length – indicates the total length of descriptors following this field.

Elementary stream map length – indicates the total length of all the elementary stream information in this program stream map.

Stream type – this identifies the type of the elementary streams found in the PES packets. Its values are shown in Figure 78.

Elementary stream ID – this is the stream id of the elementary stream's PES packets.

Elementary stream info length – this is the total length of the elementary stream descriptors following this field.

4. **Directory PES packet** – the directory information of an entire program stream is contained within directory PES packets.

Packet start code prefix + Directory stream ID – this has a value equal to 0x000001FF.

Number of access units – this gives the total number of access units (video, audio, etc.) that are referenced within this directory PES packet.

Previous directory offset – gives the byte address offset of the previous directory packet. The value of '0' indicates no previous directory packet.

Next directory offset – gives the byte address offset of the next directory packet. The value of '0' indicates no next directory packet.

Packet stream ID – gives the stream id of the elementary stream's PES packets that contains the access unit referenced.

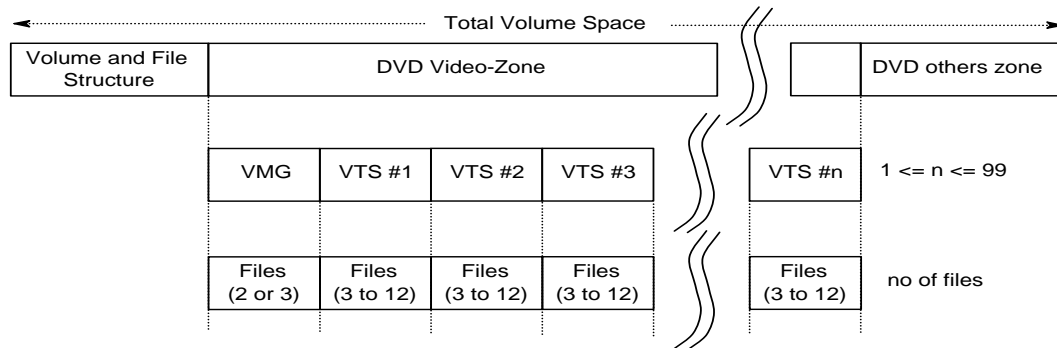
PES header position offset sign – gives the sign of the offset of the PES header whose PES packet contains this access unit. (0 – positive w.r.t the directory packet, 1 – negative w.r.t the directory packet).

PES header position offset – gives the byte offset address (+/- depending on the sign field) of the PES header of the PES packet which contains this access unit.

PTS – gives the presentation time stamp of the access unit referenced.

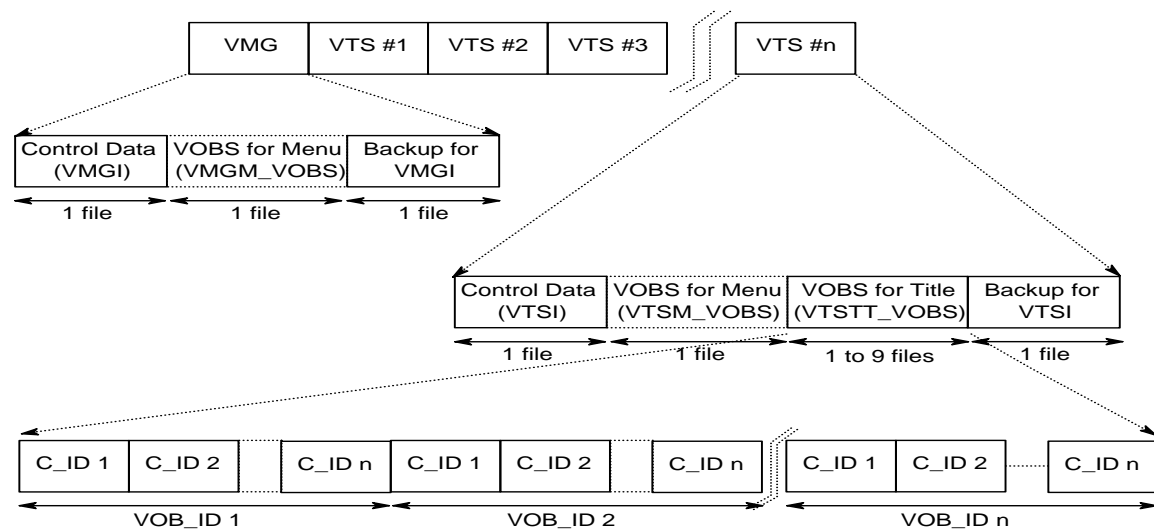
The organization of the **Logical Structure or Volume Space** on a DVD is divided up into 3 parts as follows,

1. Volume and File Structure
2. **DVD Video-Zone** – this is the **DATA** structure of the DVD video disk
3. DVD others-zone



There is only 1 DVD Video-Zone on the entire DVD disk and it consists of a single Video Manager (VMG) and at least 1 or more (max 99) Video Title Sets (VTS). The VMG is allocated at the front of the DVD-Video zone and is composed of 2 or 3 files. Each VTS can be made up of 3 or more (max 12) files.

The **VMG** and **VTS** structures are as follows,



C_ID # : CELL ID number within a VOB
VOB_ID # : VOB ID number within a VOBS

The VMG is the table of contents for all video title sets which exist in the DVD-video zone. A VMG consists of video manager control data (VMGI), video object set for VMG menu (VMGM_VOBS) and an identical backup of the video manager control data (VMGI_BUP). The VMGI consists of static information to playback video titles and provides information to support user operation. The

VMGM_VOBS is a collection of video objects used for menus that support volume access.

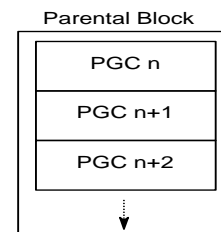
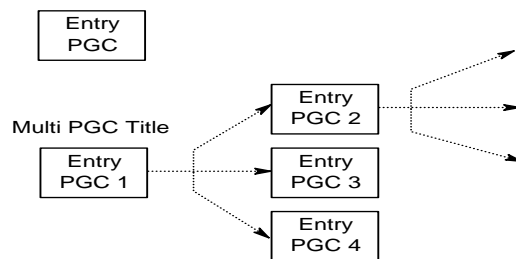
The VTS is a collection of video titles. Each VTS is composed of video title control data (VTSI), video object set for VTS menu (VTSM_VOBS), video object set for titles in a VTS (VTSTT_VOBS) and an identical backup of the video title set control data (VTSI_BUP). The VOBS for a video title can be divided into a total of 9 files.

The VOBS are a collection of video objects. A video object (VOB) is a collection of video, audio or sub-picture data. A VOB can be made of one or more cells.

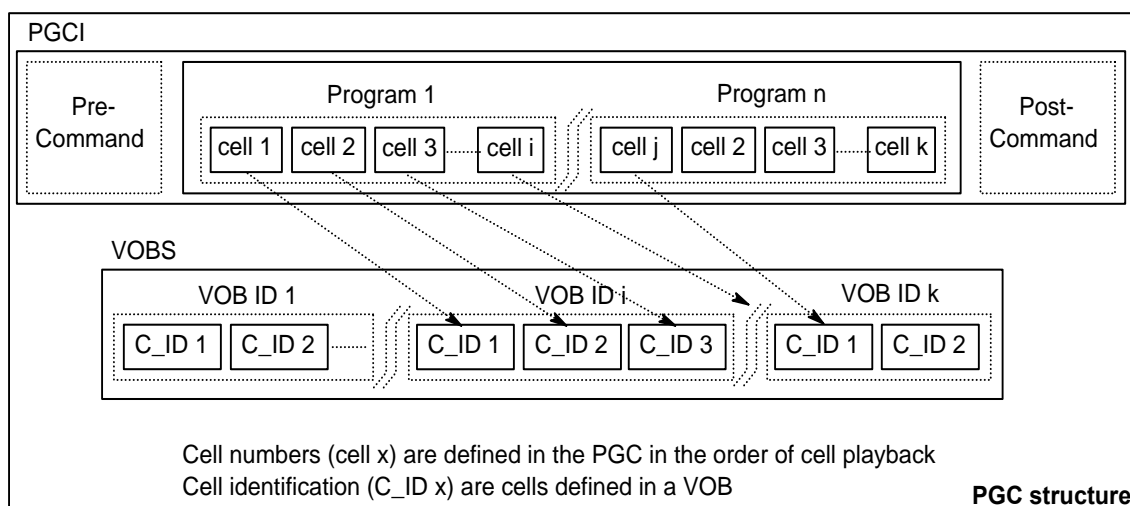
The **Presentation Structure** consists of 2 parts,

1. **Title structure** – is composed of one or more PGCs (program chain structures) having one or more cells. Also a title may have one or more parental blocks. The PGC at the beginning of a title is called the entry PGC.

One Sequential PGC per Title



2. **Program chain structure** – is composed of playback information (program chain information - PGCI) and the cells in VOBS needed for playback of the PGC. The PGCI contains navigation commands and the order of cell playback. A PGC is played back by following the playback order of cells.



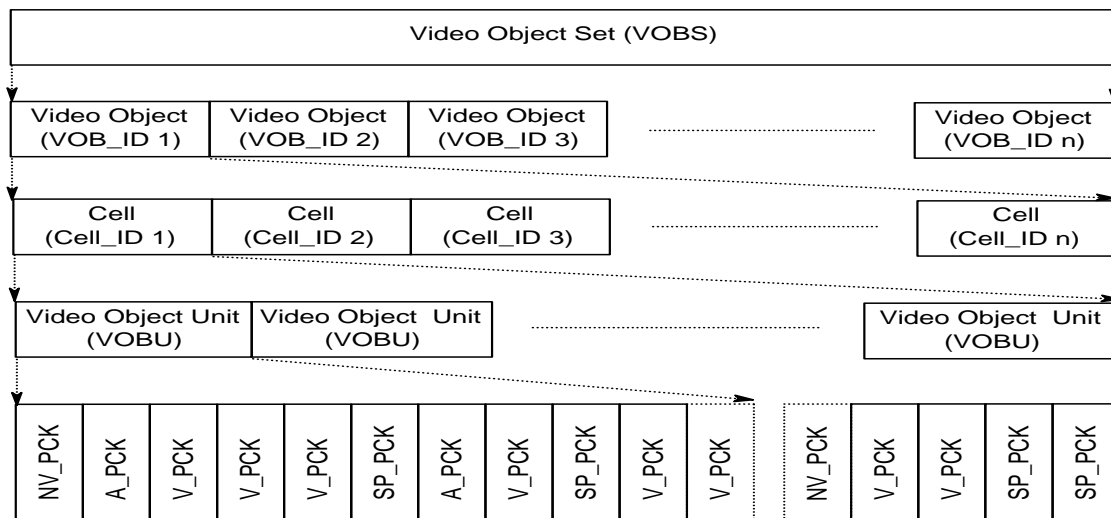
The 4 types of PGC structures are,

1. **First Play PGC (FP_PGC)** – this PGC is automatically executed at the initial access of a disc.
2. **Video Manager Menu PGC (VMGM_PGC)** – this PGC is used to playback the video manager menu common to all VTSS.
3. **Video Title Set Menu PGC (VTSM_PGC)** – this PGC is used to playback the video title set menu common to all titles in a VTS.
4. **Title PGC (TT_PGC)** – this PGC is used for playing back 1 title.

Video Object (VOB) contains presentation data and a part of the navigation data. The navigation data in a VOB is made up of presentation control information (PCI) and data search information (DSI). The presentation data is video data, audio data and sub picture data. A VOB is made up of **packs** carried in **PES packets**. The DVD environment contains 4 types of packs listed below,

1. **Navigation pack (NV_PCK)** – Presentation control information (PCI) and Data search information (DSI).
2. **Video pack (V_PCK)** – Video (MPEG 2 16/9 aspect ratio) data.
3. **Audio pack (A_PCK)** – Audio (MPEG1/2, AC3, DTS, linear PCM) data.
4. **Sub-picture pack (SP_PCK)** – Sub-picture data used for providing subtitling information in a DVD.

A VOB can have 1 video stream, up to 8 audio streams and up to 32 sub picture streams. A video object set (VOBS) is a collection of video objects as shown. A VOB is divided into cells. A cell is further divided into video object units (VOBU). A video object unit is sub divided into the above defined packs.



A video object unit (VOBU) is a sequence of packs in recording order. It starts with exactly one NV_PCK, encompasses all the following packs (if any) and ends before the next NV_PCK. A VOBU of a cell represents a presentation period of at least 0.4 seconds and at the most 1 second. A VOB contains an integer number of VOBUs. The following rules apply to a VOBU,

1. The presentation period of a VOBU is equal to an integer multiple of video fields. This is also the case when the VOBU does not contain any video packs.

2. The presentation and termination time of a VOBU is defined in 90 KHz units.
3. When the VOB contains video data, the video data shall represent one or more GOPs.
4. When a VOBU with video data is followed by another VOBU without video data (in the same VOB), the last coded picture shall be followed by a sequence end code.
5. When the presentation period of the VOBU is longer than the presentation period of the video it contains, the last coded picture shall be followed by a sequence end code.
6. The video data in a VOBU shall never contain more than one sequence end code.

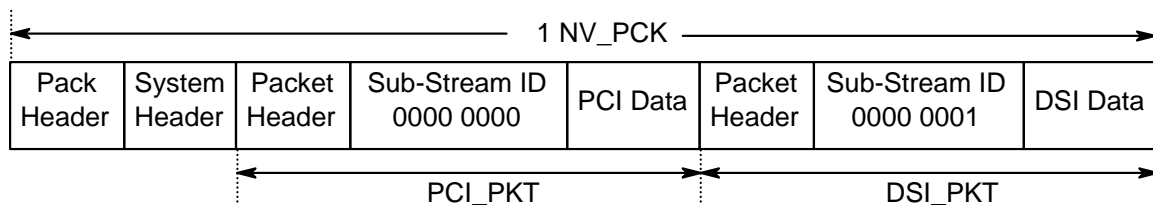
In DVD the **Sub-Stream-ID** is used to differentiate between different types of data packs carried in PES packets.

Stream ID	Type of Stream Coded
110x, 0***	MPEG 1/2 audio stream (*** is the audio stream number, x - don't care)
1110, 0000	MPEG 2 video stream
1011, 1101	private stream 1 - see sub stream id's
1011, 1111	private stream 2 - see sub stream id's

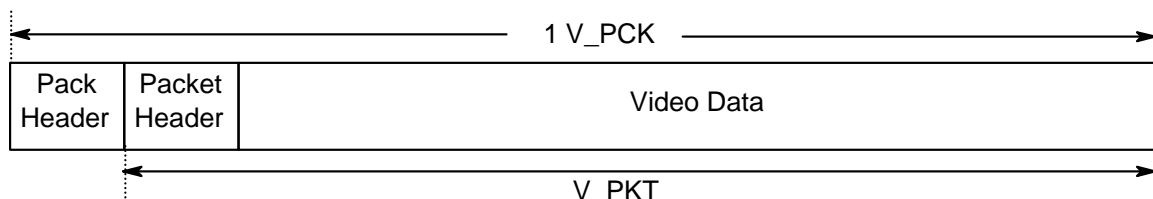
Private Stream 1 (Sub Stream ID)	Type of Stream Coded
001*, ****	Sub-picture stream (***** is the stream number)
1000, 0***	Dolby ac3 audio stream (** is the stream number)
1000, 1***	DTS audio stream (** is the stream number)
1010, 0***	Linear PCM audio stream (** is the stream number)

Private Stream 2 (Sub Stream ID)	Type of Stream Coded
0000, 0000	Navigation Data PCI stream
0000, 0001	Navigation Data DSI stream

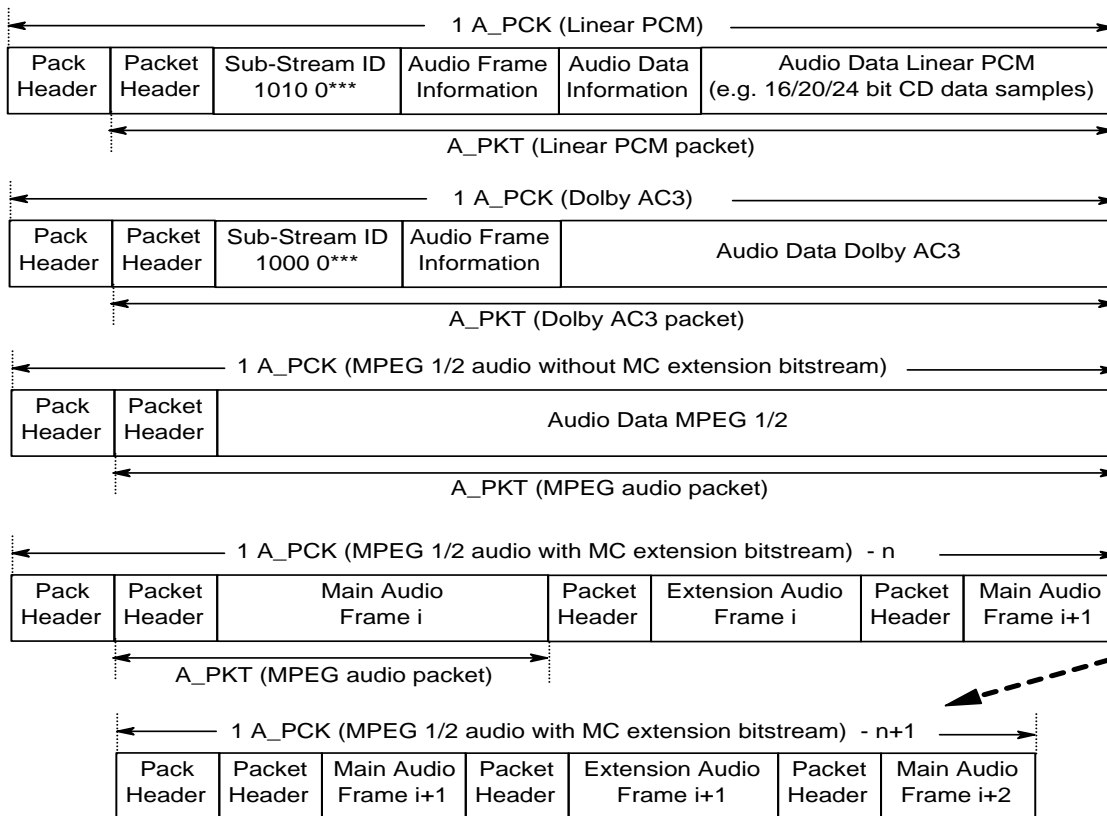
1. The **Navigation Pack Structure** (NV_PCK) consists of a pack header, system header, a PCI packet (PCI_PKT) and a DSI packet (DSI_PKT). This pack is the 1st pack in a VOBU.



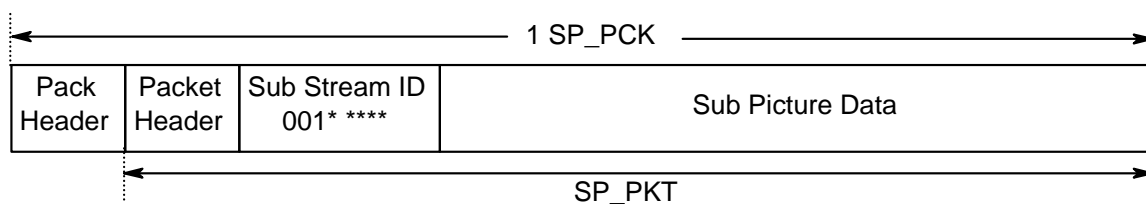
2. The **Video Pack Structure** (V_PCK) comprises of a pack header and a video packet (V_PKT).



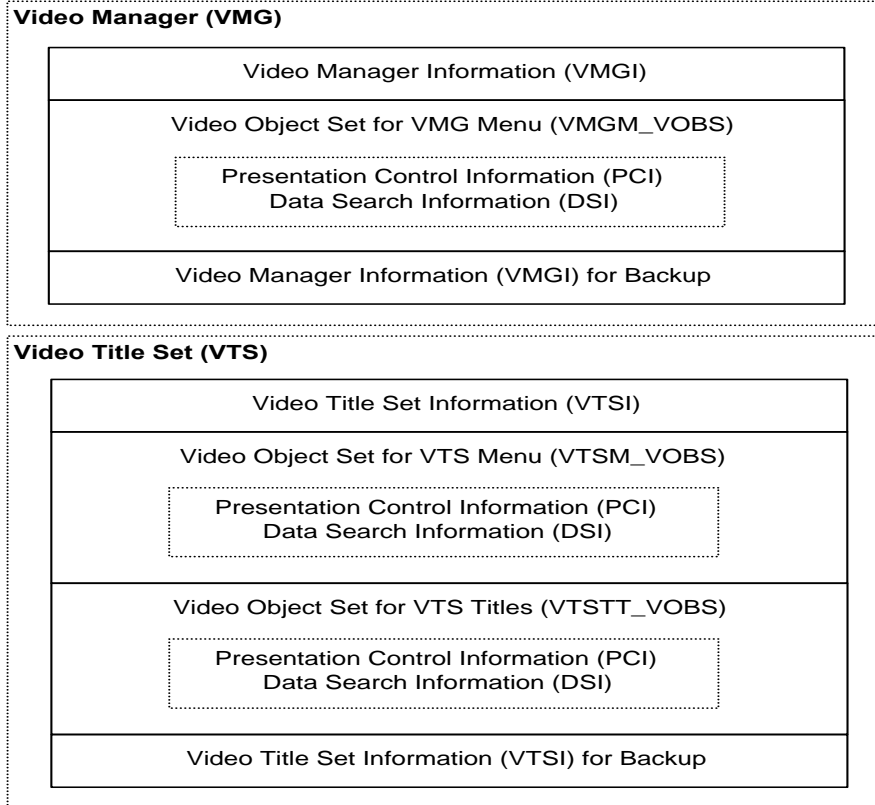
3. The **Audio Pack Structure** (A_PCK) comprises of a pack header and audio packets (A_PKTs). Due to the different types of audio streams an audio pack can have different formats.



4. The **Sub-Picture Pack Structure** (SP_PCK) comprises of a pack header and a Sub-Picture packet (SP_PKT) with its own sub stream id.



- **Navigation Guide (DVD)** – Navigation Guide Data is the information about attributes and playback control for the presentation data (video, audio, sub-picture). There are 4 types of Navigation Data, Video Manager Information (VMGI), Video Title Set Information (VTSI), Presentation Control Information (PCI) and Data Search Information (DSI).



Video Manager Information (VMGI) – this contains information to search the DVD video titles, presentation of the program chains and the video manager menus. It also contains information on parental management of video titles, attributes on each video title and any text data associated with each video title.

Video Manager Information Management Table (VMGI_MAT)
Title Search Pointer Table (TT_SRPT)
Video Manager Menu PGCI Unit Table (VMGM_PGCI_UT)
Parental Management Information Table (PTL_MAIT)
Video Title Set Attribute Table (VTS_ATRT)
Text Data Manager (TXTDT_MG)
Video Manager Menu Cell Address Table (VMGM_C_ADT)
Video Manager Menu Video Object Unit Address Map (VMGM_VOBU_ADMAP)

1. **VMGI_MAT** describes the size of the VMG and VMGI, the start address of each information found in the VMG, and attribute information on Video Object Set (VOBS) for Video Manager Menu (VMGM_VOBS).

2. **TT_SRPT** describes the search information of Video Titles using title search pointers for every video title on the DVD.
3. **VMGM_PGCI_UT** describes the information on VMG Menu Program Chains (VMGM_PGCI) which presents the Video Manager Menu in each language. It consists of Video Manager Menu Language Unit Search Pointers for every language followed by Video Manager Menu Language Units.
4. **PTL_MAIT** describes the information on Parental Level for VMG and each VTS using parental management information pointers for each designated country.
5. **VTS_ATRT** gives the attribute information on every Video Title set (VTS) using video title set attribute search pointers.
6. **TXTDI_MG** provides information such as the Volume Name, the Title Name, the Album Name and the Producer Name for every title on the DVD in text style.
7. **VMGM_C_ADT** describes the start/end addresses of all cells and interleaved units in the VMGM_VOBS. A cell piece is defined as a cell if it belongs to a VOB which is allocated in a contiguous block. A cell piece is defined as an interleaved unit if it belongs to a VOB which is allocated in an interleaved block.
8. **VMGM_VOBU_ADMAP** describes the start addresses of all the VOBUs in the VMGM_VOBS in the ascending order of their Logical Block Numbers.

Video Title Set Information (VTSI) – this contains information for one or more Video Titles and Video Title Set Menu. VTSI provides the management information of these video titles, such as information to search part of a title, information to playback a Video Object Set (VOBS) or a Video Title Set Menu (VTSM) and also provides the attributes of VOBS.

Video Title Set Information Management Table (VTSI_MAT)
Video Title Set Part_of_Title Search Pointer Table (VTS_PTT_SRPT)
Video Title Set Program Chain Information Table (VTS_PGCIT)
Video Title Set Menu PGCI Unit Table (VTSM_PGCI_UT)
Video Title Set Time Map Table (VTS_TMAPT)
Video Title Set Menu Cell Address Table (VTSM_C_ADT)
Video Title Set Menu Video Object Unit Address Map (VTSM_VOBU_ADMAP)
Video Title Set Cell Address Table (VTS_C_ADT)
Video Title Set Video Object Unit Address Map (VTS_VOBU_ADMAP)

1. **VTSI_MAT** provides the start address of each information item in the VTSI and the attributes of VOBS in the VTS.
2. **VTSI_PTT_SRPT** contains search pointers for the entry point within titles. The entry point within a title is defined as Part_of_Title (PTT) and this table contains search pointers for each and every PTT.
3. **VTS_PGCIT** describes the information on VTS Program Chains which present the Video Title Set Program Chains. It contains program chain information search pointers followed by one or more VTS program chain information items.
4. **VTSM_PGCI_UT** describes the information on VTS Menu Program Chains which present the Video Title Set Menu in each language. It consists of VTS Menu Language Unit Search Pointers for every language followed by VTS Menu Language Units.

5. **VTSM_TMAPT** describes the information on the recording position of VOBUs in each PGC of this VTS for every definitive time of presentation.
6. **VTSM_C_ADT** provides the start/end addresses for all cells and interleaved units in the VTSM_VOBS. It consists of VTS Menu Cell Address Table Information followed by VTS Menu Cell Piece Information for every cell piece.
7. **VTSM_VOBUS_ADMAP** contains the start addresses of all the VOBUs in the VTSM_VOBS in the ascending order of their Logical Block Numbers.
8. **VTSM_C_ADT** provides the start/end addresses for all cells and interleaved units in the VTSTT_VOBS.
9. **VTSM_VOBUS_ADMAP** contains the start addresses of all the VOBUs in the VTSTT_VOBS in the ascending order of their Logical Block Numbers.

Program Chain Information (PGCI) is the Navigation Data to control the presentation of the Program Chains (PGC) and is made up of the below tables.

Program Chain General Information (PGC_GI)
Program Chain Command Table (PGC_CMDT)
Program Chain Program Map (PGC_PGMAP)
Cell Playback Information Table (C_PBIT)
Cell Position Information Table (C_POSIT)

Presentation Control Information (PCI) is the Navigation Data to control the presentation of a VOB unit (VOBU). The PCI is described in the PCI packet (PCI_PKT) found in the Navigation Pack (NV_PCK). PCI is made up of 4 pieces of information,

1. **PCI_GI** – PCI general information.
2. **NSML_AGLI** – is the information on the destination address of the Angle Change for non-seamless playback of a VOB. A VOB could be composed of up to 9 angle cells (with each angle cell having VOBUs which could have different camera angles of the same video scene) selectable via the user interface.
3. **HLI** – Highlight information is used to highlight one rectangular block in the sub-picture display area.
4. **RECI** – This provides the Recording Information (International Standard Recording Code) for every video data, audio data and sub-picture data which are recorded in a VOB.

Data Search Information (DSI) is the Navigation Data to search and carry out seamless playback of the VOB unit (VOBU). The DSI is described in the DSI packet (DSI_PKT) found in the Navigation Pack (NV_PCK). DSI is made up of 5 pieces of information,

1. **DSI_GI** – DSI general information.
2. **SML_PBI** – Seamless Playback Information provides information for the seamless presentation of a VOB.
3. **SML_AGLI** – provides the information on the destination address of the Angle Change for seamless playback of a VOB.

4. **VOBU_SRI** – VOB unit search information describes the start address of VOBUs presented 0.5 x n seconds before and after the presentation start time of the VOB which includes this DSI in a particular cell.
5. **SYNCD** – Synchronization information is the addresses of audio and sub-picture data presented synchronously with the video data in a VOB which includes this DSI.

MPEG4 Access Unit, FlexMux and TransMux Layers in Multi-Media Streaming Applications – (ISO/IEC 14496 systems)

Audio Visual objects (AVOs) for any multi-media MPEG4 application are transmitted in **elementary streams**. These elementary streams are characterized by the Quality of Service (QoS) requested for transmission (e.g. maximum bit rate, bit error rate, etc.), as well as other parameters, including information about the stream type to determine the decoder resources and the precision for encoding timing information. The transmission of this streaming information in a synchronized manner from source to destination, exploiting different QoS as available from the network, is specified using an Access Unit Layer and a Two Layer Multiplexer as shown in figure 79.

- The **Access Unit (stream multiplexing) Layer** allows identification of access units (e.g. video or audio frames, scene description commands) in elementary streams, recovery of the AV object's or scene description's time base and enables synchronization among them. The access unit header can be configured in a large number of ways, allowing its use in a broad spectrum of systems.
- The **FlexMux (flexible multiplexing) Layer** is completely specified by MPEG4 and contains a multiplexing tool which allows grouping of elementary streams with a low multiplexing overhead. This may be used to group elementary streams with similar QoS requirements.
- The **TransMux (transport multiplexing) Layer** offers transport services matching the requested QoS. Only the interface to this layer is specified by MPEG4. Any existing transport protocol stack such as UDP/IP (RTP), ATM (AAL5), or MPEG2 transport (PES encapsulation) may be used as a TransMux instance. This choice is left to the end user/service provider, and thus allows MPEG4 to be used in a wide variety of operating environments.

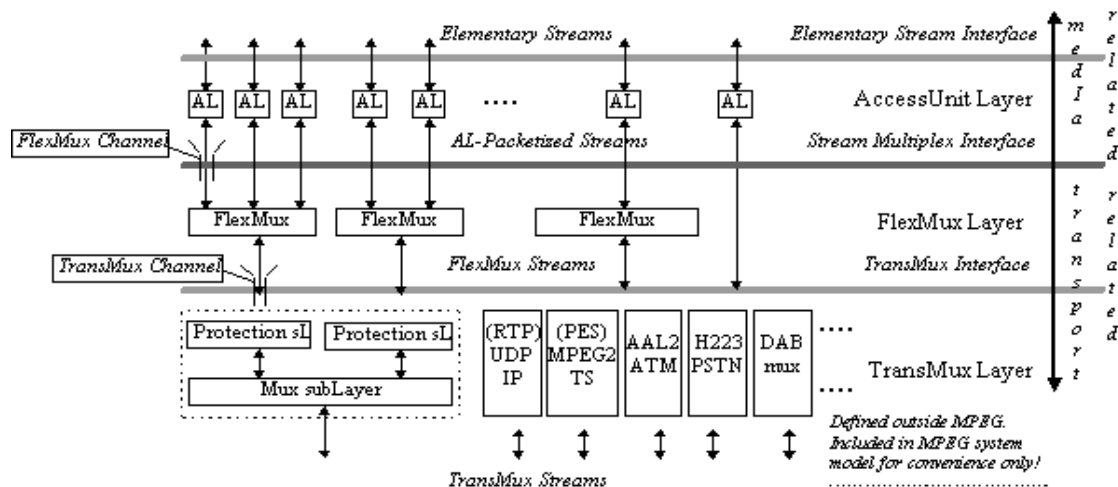


Figure 79. MPEG4 System Layer Packetization Method

In MPEG4, the Access Unit Layer is always present. However, use of the FlexMux multiplexing layer is optional, and may be bypassed if the underlying TransMux Layer instance provides equivalent functionality.

The **Delivery Multimedia Integration Framework (DMIF)** in MPEG4 addresses operation of multimedia applications over interactive networks, in broadcast environments and from disks. The DMIF architecture is such that applications which rely on DMIF for communications do not have to be concerned with the underlying communications method. The implementation of DMIF takes care of the network details presenting the application with a simple interface.

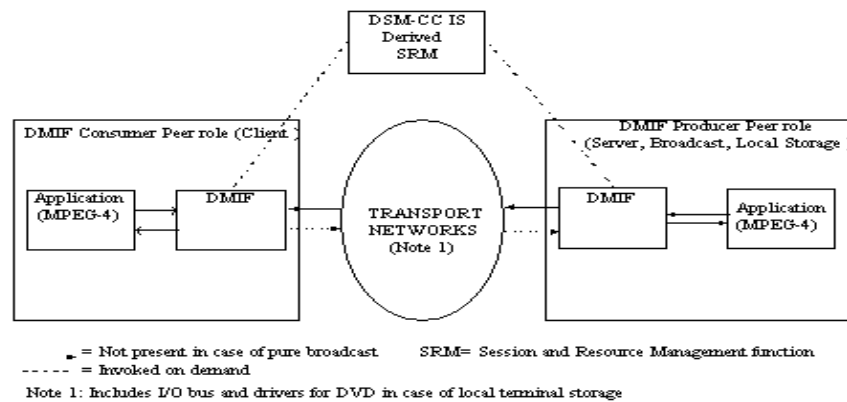


Figure 80. DMIF architecture

To an MPEG4 application, DMIF presents a consistent interface irrespective of how TransMux streams are received,

- By interacting with a remote interactive DMIF peer over networks
- By interacting with broadcast or storage media over networks

The MPEG4 application can request from DMIF the establishment of channels with specific QoSs and bandwidths for each elementary stream. DMIF ensures the timely establishment of the channels with the specified bandwidths while preserving the QoSs over a variety of intervening networks between the interactive peers. DMIF allows each peer to maintain its own view of the network, thus reducing the number of software protocol stacks supported at each terminal.

Control of DMIF spans both the FlexMux and the TransMux layers. In the case of FlexMux, DMIF provides control of the establishment of FlexMux channels. In the case of TransMux, DMIF uses an open interface which accommodates existing and future networks through templates called connection resource descriptors. MPEG4 offers a transparent interface with signaling primitive semantics. These MPEG4 semantics at the interface to DMIF are interpreted and translated into the appropriate native signaling messages of each network, with the help of relevant standards. The mappings of these translations are beyond the scope of MPEG4 and are defined by network providers.

In a typical operation an end user may access AVOs distributed over a number of remote interactive peers, broadcast and storage systems. The initial network connection to an interactive peer may consist of a best effort connection over a ubiquitous network. If the content warrants it, the end-user may seamlessly scale up the quality by adding enhanced AVO streams over connection resources with guaranteed QoS.

The **De-multiplexing, buffer management and timing recovery** in MPEG4 streams is achieved as follows,

Individual elementary streams have to be retrieved from incoming data from some network connection or a storage device. Each network connection or file is considered a TransMux channel in the MPEG4 system model. The de-multiplexing is partially or completely done by layers outside the scope of MPEG4, depending on the application. For the purpose of integrating MPEG4 in system environments, the **Stream Multiplex Interface** as shown in figure 79 is the reference point. Adaptation Layer (AL) packetized streams are delivered at this interface. The TransMux Interface specifies how either AL packetized streams (no FlexMux used) or FlexMux streams are to be retrieved from the TransMux Layer.

In the same way that MPEG2 described the behavior of an idealized decoding device along with the bitstream syntax and semantics, MPEG4 defines a **System Decoder Model**. This allows defining a terminal's operation without making unnecessary assumptions about its implementation details. This allows implementing real MPEG4 terminals and decoding devices in a variety of ways. These devices range from television receivers which have no ability to communicate with the sender to computers which are fully enabled with bi-directional communication. Some devices will receive MPEG4 streams over isochronous networks while others will use non-isochronous means (e.g. the Internet) to exchange MPEG4 information. The System Decoder Model provides a common model on which all implementations of MPEG4 terminals can be based.

1. **De-multiplexing** - The retrieval of incoming data streams from network connections or storage media consists of two tasks. First, the channels must be located and opened. This requires a transport control entity, e.g., DMIF. Second, the incoming streams must be properly de-multiplexed to recover the elementary streams from downstream channels (incoming at the receiving terminal). In interactive applications, a corresponding **multiplexing stage** will multiplex upstream data in **upstream channels** (outgoing from the receiving terminal). These elementary streams can carry either AV object data, scene description information, or control information related to AV objects or to system management. The Access Unit Layer has a set of tools for consistency checking, padding, to convey time base information and to carry time stamped Access Units of an elementary stream. The **TransMux protection sublayer** includes error protection and error detection tools suitable for the given network or storage medium.

TransMux streams coming from the network (or storage device) as shown in figure 81, are de-multiplexed into FlexMux Streams and passed on to appropriate FlexMux de-multiplexers that retrieve the elementary streams. These elementary streams are parsed and sent on to the appropriate decoders. Decoding recovers the data in an AV object from its encoded form and performs the necessary operations to reconstruct the original AV object for rendering on an appropriate display device. The reconstructed AV object is made available to the composition layer for potential use during scene rendering. Decoded AVOs, along with scene description information, are used to compose the scene as described by the author. The user can, to the extent allowed by the author, interact with the scene which is eventually rendered and presented.

To be able to relate elementary streams to AV objects within a scene, Object Descriptors and Stream Map Tables are used. Object Descriptors convey information about the number and properties of elementary streams that are associated to particular AV objects. The Stream Map Table links each stream to a Channel Association Tag that serves as a handle to the channel that carries this stream. Resolving Channel Association Tags to the actual transport channel as well as the management of the sessions and channels is addressed by the DMIF part of the MPEG4 standard.

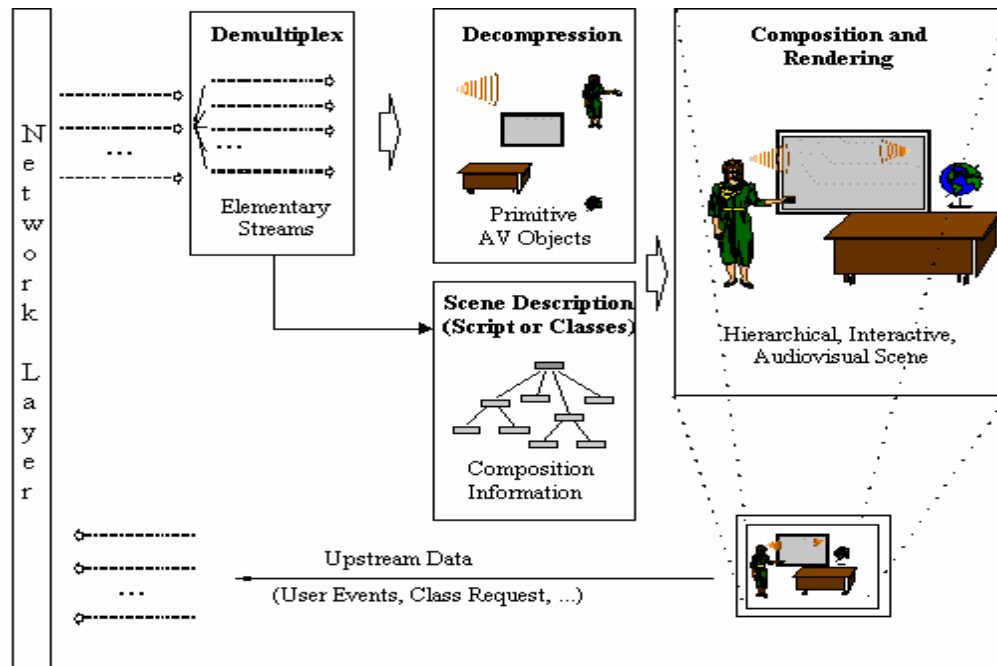
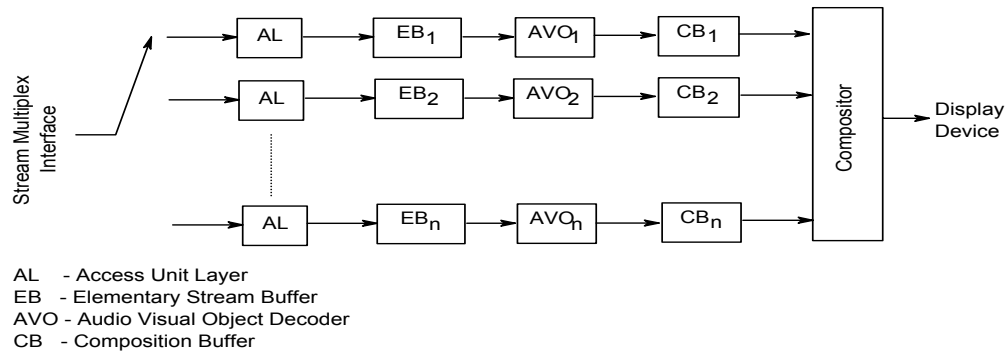


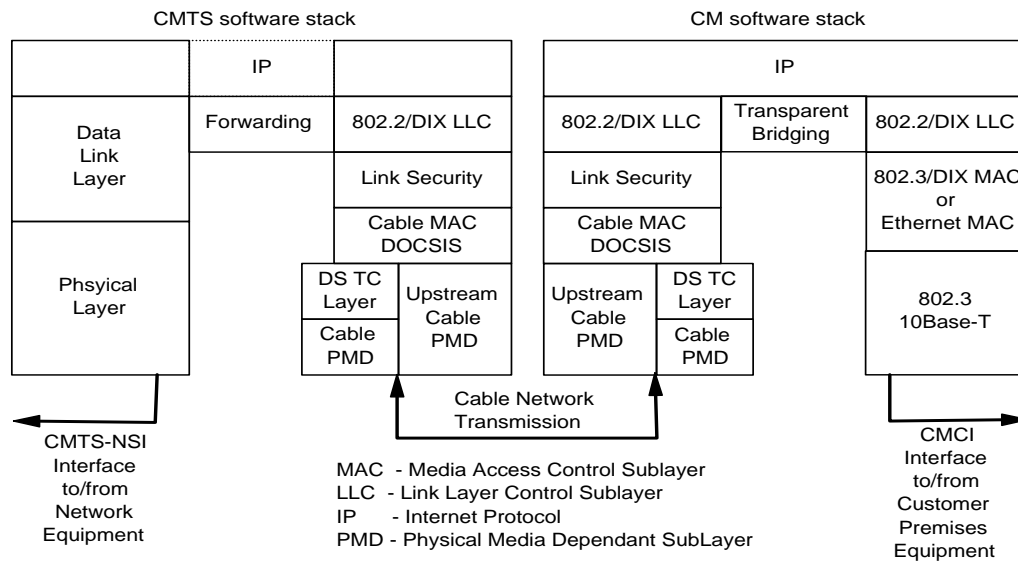
Figure 81. MPEG4 de-multiplexing, decoding and rendering components

2. **Buffer Management** - To predict how the MPEG4 decoder will behave when it decodes the various elementary data streams that form an MPEG4 session, the Systems Decoder Model enables the encoder to specify and monitor the minimum buffer resources that are needed to decode a MPEG4 session. The required buffer resources are conveyed to the decoder within Object Descriptors during the setup of the MPEG4 session, so that the decoder can decide whether it is capable of handling this session.

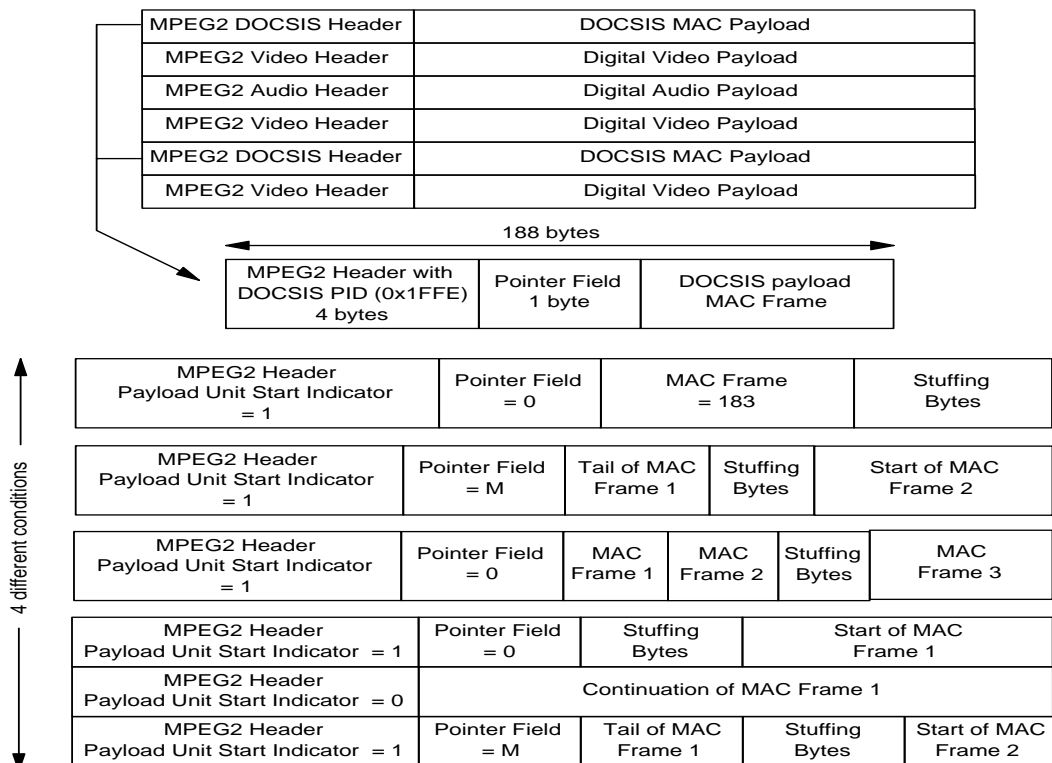


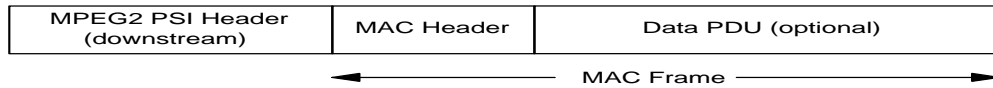
3. **Time Identification** – For a real time operation, a timing model is used in which the end-to-end delay from the signal output from an encoder to the signal input to a decoder is constant. There are two types of timing information. The **first** is used to convey the speed of the encoder clock, or time base (similar to PCR in MPEG2 transport streams or SCR in MPEG2 program streams), to the decoder. The **second**, consisting of time stamps attached to portions of the encoded AV data, contains the desired decoding time (similar to DTS/PTS timestamps used in MPEG2) for Access Units or composition and expiration time for Composition Units. This information is conveyed in AL-PDU Headers generated in the Access Unit Layer. With this timing information, the inter-picture interval and audio sample rate can be adjusted at the decoder to match the encoder's inter-picture interval and audio sample rate for synchronized operation.

The downstream and upstream encapsulated data packets are forwarded through the various CMTS and the CM layers as shown in the software stacks running on them,

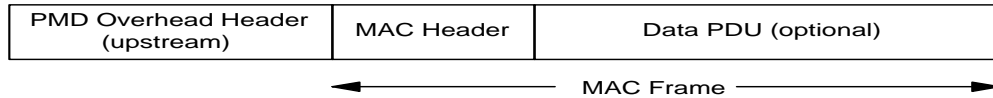


The **downstream channel bit-stream** (CMTS → CM) after QAM demodulation in the Cable PMD (physical media dependant layer) enters the downstream TC layer as an MPEG2 bitstream. These MPEG2 packets have video/audio payloads as well as the DOCSIS MAC payload as shown below.





The **upstream channel bit-stream** (CM → CMTS) is a burst of MAC frames with PMD overhead headers.



A MAC frame is the **basic unit of data/information transfer** between the MAC sub-layers at the CMTS and at the CM. The same basic structure is used in both the upstream and downstream directions. MAC frames are variable in length.

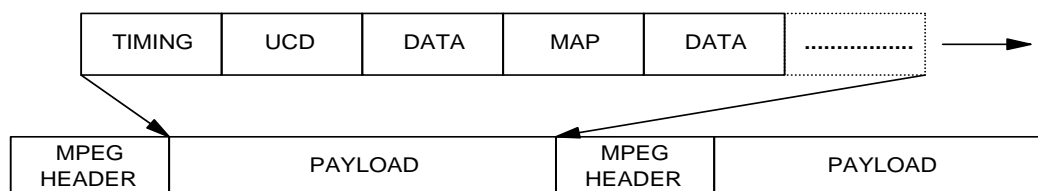
A **Service Flow** is central to the operation of the MAC protocol and provides a mechanism for maintaining upstream and downstream quality of service management between a CM and the CMTS. A **Service Flow ID** (SFID) is a unidirectional mapping between a CM and the CMTS. Service Flow IDs have their associated **Service IDs** (SIDs). A certain upstream bandwidth is allocated to all SIDs and hence to all CMs by the CMTS.

In a basic CM implementation, a CMTS assigns 2 service flows (one upstream and one downstream) to that CM.

An upstream channel's time line is divided into intervals by the upstream bandwidth allocation mechanism. Each interval is an integral multiple of mini-slots (mini-slot being the unit of granularity for upstream transmission). An upstream channel is identified by a unique upstream channel ID and consists of a contiguous stream of mini-slots which are described and allocated by the Upstream Channel Descriptor (UCD) & Upstream Channel Map (MAP) messages associated with that upstream channel ID. A cable modem registers for one and only one upstream channel.

In a typical Cable TV / Cable Modem implementation there are several CMTS which serve a wide area of homes with each CMTS having 1 downstream and several upstream channels.

The **Downstream Message Data Format** is contiguous and is shown next,



TIMING is the synchronization information transmitted periodically by the CMTS to all CMs, for synchronizing their internal sampling clocks with the CMTS clock.

DATA is the downstream data to CMs on that downstream channel and could consist of web-pages, streaming audio/video, etc.

UCD is the upstream channel descriptor which is broadcasted periodically to all CMs by the CMTS to define the physical layer and burst attributes of a particular upstream channel. It has 2 descriptions,

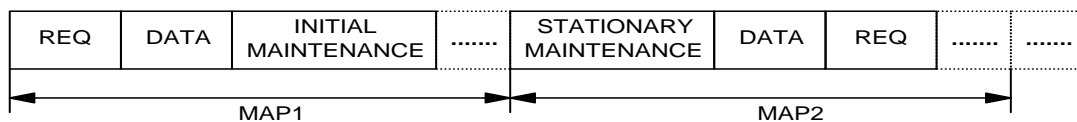
- Channel Attributes – main characteristics of the upstream channel.
- Burst Attributes – additional characteristics of the upstream channel as associated with the Interval Usage Codes (IUC) of a MAP.

MAP defines the bandwidth allocation on a specific upstream channel or what opportunities are available on an upstream channel during a specific interval of time. This is transmitted periodically to all CMs to define the specific opportunities during an interval of time on the upstream channel. MAP associates IUC codes with SIDs, thus specifying what each CM is allowed to transmit during that interval.

- **IUC** defines what type of upstream burst is possible during a specific interval of time on the upstream channel. An IUC is usually of the type, REQ to send, DATA, Maintenance Intervals, etc.

Each upstream channel has its own UCD and MAP.

The **Upstream Message Data Format** consists of **bursts** (as shown below) from the CMs connected to a CMTS network when they receive the opportunity by the CMTS via the MAP. Which CMs are allowed to transmit, when and what they transmit, is specified by the MAP sent down by the CMTS in the downstream channel.



A single upstream message burst has the following format at the PMD layer before modulation.



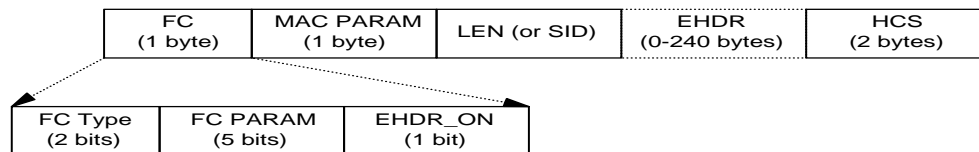
A given upstream channel's bandwidth is designated through the IUC in the MAP and can contain the following messages from the CM,

- **REQ** – this is a request for permission to transmit.
- **REQ/DATA** – this is a request to transmit plus DATA.
- **INITIAL MAINTENANCE** – this is for initial ranging (or power level calibrations of the CM burst signal).
- **STATION MAINTENANCE** – this is for adjusting a CM's transmission parameters.
- **DATA Grant** – this is the interval for CM to transmit the actual DATA (could be user pay per view requests, interactive gaming with another user, or user requests for downloading webpages).

In cases of collisions on the upstream channel between 2 or more CM's messages (resulting in lost or in-correct messages to the CMTS), the CMs have to re-transmit those messages at another transmit opportunity.

A MAC header structure and some important MAC headers types are shown below,

- **MAC HEADER**



The **Frame Control (FC)** field is the first byte and uniquely identifies the rest of the contents within the MAC header.

The **MAC Parameter** (MAC PARAM) field,

1. gives the Ethernet Header Field (EHDR) when EDHR_ON = 1
2. gives the number of MAC frames if concatenated MAC frames are used and if the MAC header is a concatenation header
3. gives the number of mini-slots requested if MAC

The **Length or Service ID** (LEN/SID) field,

1. gives the length of the MAC frame, which is the sum of the extended header (if present) plus the number of bytes following the HCS field
2. is the Service ID if the MAC header is a request header

The **Extended MAC Header** (EHDR) is present when EHDR_ON = 1 and any DOCSIS compliant CM must support this header. It is mainly used for 4 things,

1. supporting piggyback requests whereby a CM can request bandwidth for subsequent transmissions
2. supporting upstream privacy headers
3. enhancing service flow operations
4. supporting payload header suppression to provide increased bandwidth efficiency

The **Header Check Sequence (HCS)** is a 2 byte CRC check to ensure the integrity of the MAC header. This is checked for every MAC frame because of the harsh environment in which a cable modem (CM) operates.

- **MAC Frame Control Field (FC)** fields

The **Frame Control Type Field (FC_TYPE)** is defined as follows,

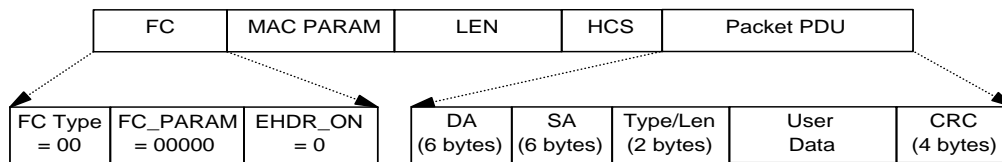
1. 00 – Packet PDU MAC Header
2. 01 – ATM PDU MAC Header – reserved for future in MAC frames are in ATM cells
3. 11 – MAC Specific Header

The **Frame Control Parameter (FC_PARAM)** specifies the following MAC specific headers,

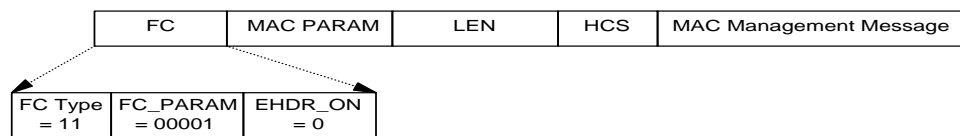
1. 00000 – Timing or Synchronization Header
2. 00001 – MAC Management Message Header
3. 00010 – Request Frame
4. 00011 – Fragmentation Header
5. 11100 – Concatenation Header

The **Ethernet Header ON (EHDR_ON)** field specifies whether an Ethernet Header is present in the MAC header.

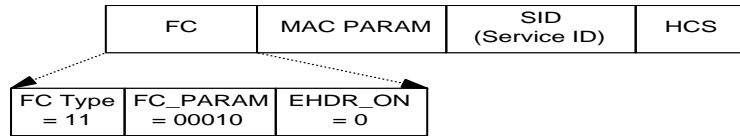
- **Packet PDU (Data PDU) MAC Header** – shown below is a MAC header followed by a typical Ethernet 802.3 packet PDU,



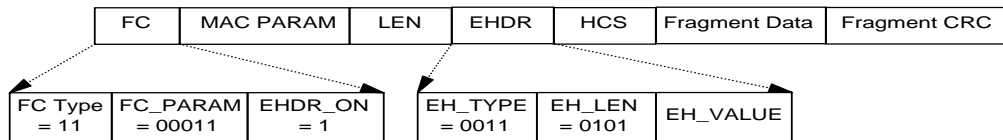
- **Timing MAC Header** – is used to transmit timings and adjustments in the downstream for CMs to adjust (synchronize) their internal clocks to the Global (CMTS) timing reference. In the upstream, this MAC header is used to transmit ranging messages for adjusting the cable modem's timing and power requirements. This header is followed by a Packet PDU which is a MAC Management Message as follows,
 1. SYNC (synchronization) message in the downstream
 2. RNG-REQ (ranging request) message in the upstream
- **Management MAC Header** – MAC management messages (upstream/downstream) follow this header. These messages are probably the most important part of the MAC software layer because they define how the MAC layer functions.



- **Request Frame MAC Header** – is the mechanism used by the cable modem to request bandwidth in mini-slots and thus is only used in the upstream.



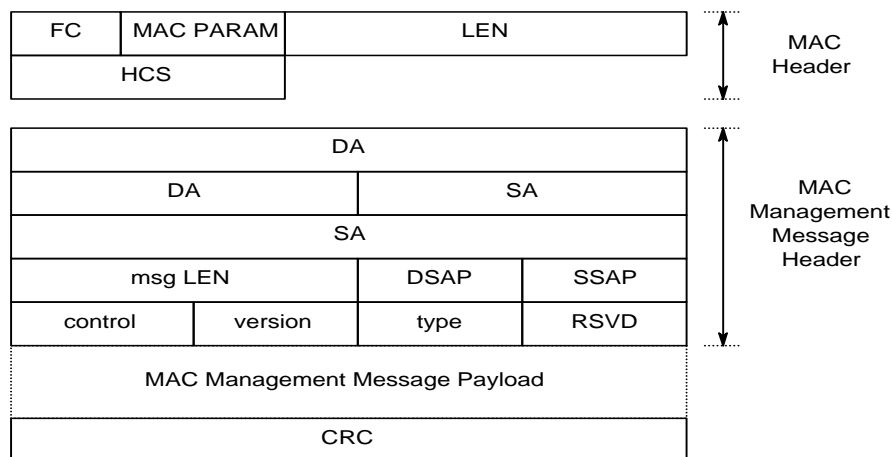
- **Fragmentation MAC Header** – this header provides a basic mechanism to split a large MAC PDU into smaller pieces that can then be transmitted individually and re-assembled at the CMTS. This header is only applicable in the upstream and any CM on the network has to support this header. To reduce the burden on the CMTS and to reduce unnecessary overhead, fragmentation headers **MUST NOT** be used on un-fragmented frames.



- **Concatenation MAC Header** – this header allows multiple MAC frames to be concatenated. This allows a single upstream MAC “burst” to be transferred across the network by the CM. The PHY overhead and the concatenation MAC header is found only once in the entire burst. This header is found only in the upstream channel and must be supported by all CMs.



- **Extended MAC Header** – every MAC header, **except the timing, concatenation and request frame** MAC headers have the capability of defining an Extended Header Field (EHDR) indicated by the EHDR_ON flag. Its applications have been shown earlier.
- **MAC Management Message Header** – MAC management messages must be encapsulated in an LLC un-numbered frame which in turn is encapsulated within the MAC frame as shown below. These messages are used in both the downstream and upstream directions and form an effective 2-way communication between the CM and the CMTS to maintain QoS.



FC, MAC PARAM, LEN, HCS – common MAC frame header. All MAC management messages use a MAC-specific header.

Destination Address (DA) – MAC management frames will be addressed to a specific CM's unique MAC address or to a multicast MAC address (01-E0-2F-00-00-01) while transmitting certain MAC messages like the MAP.

Source Address (SA) – the MAC address of the source CM or the CMTS.

Msg Length – length of the MAC message from DSAP to the end of payload.

DSAP – the LLC null destination SAP (00) as defined by ISO8802-3.

SSAP – the LLC null source SAP (00) as defined by ISO8802-3.

Control – un-numbered information frame (03) as defined by ISO8802-3.

Version & Type – messages with version 1 are understood by CM/CMTS compliant with DOCSIS 1.0, 1.1 and 2.0. Messages with version 2 are understood by CM/CMTS compliant with DOCSIS 1.1 and 2.0. Messages with version 3 are understood by CM/CMTS compliant with DOCSIS 2.0.

RSVD – is used to align the message payload on 32 bit boundary.

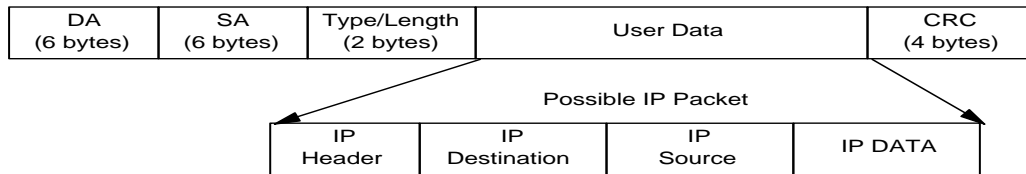
Management Message Payload – is variable length as defined for each specific MAC management message.

CRC – covers entire message including header fields. Polynomial defined in ISO8802-3.

Type #	Version #	Message Name	Message Description
1	1	SYNC	Timing Synchronization
2 or 29	1 or 3	UCD	Upstream Channel Descriptor
3	1	MAP	Upstream Bandwidth Allocation MAP
4	1	RNG-REQ	Ranging Request
5	1	RNG-RSP	Ranging Response
6	1	REG-REQ	Registration Request
7	1	REG-RSP	Registration Response
8	1	UCC-REQ	Upstream Channel Change Request
9	1	UCC-RSP	Upstream Channel Change Response
10	1	TRI-TCD	Telephony Channel Descriptor
11	1	TRI-TSI	Termination System Information
12	1	BPKM-REQ	Privacy Key Management Request
13	1	BPKM-RSP	Privacy Key Management Response
14	2	REQ-ACK	Registration Acknowledge
15	2	DSA-REQ	Dynamic Service Addition Request
16	2	DSA-RSP	Dynamic Service Addition Response
17	2	DSA-ACK	Dynamic Service Addition Acknowledge
18	2	DSC-REQ	Dynamic Service Change Request
19	2	DSC-RSP	Dynamic Service Change Response
20	2	DSC-ACK	Dynamic Service Change Acknowledge
21	2	DSD-REQ	Dynamic Service Deletion Request
22	2	DSD-RSP	Dynamic Service Deletion Response
23	2	DCC-REQ	Dynamic Channel Change Request
24	2	DCC-RSP	Dynamic Channel Change Response
25	2	DCC-ACK	Dynamic Channel Change Acknowledge
26	2	DCI-REQ	Device Class Identification Request
27	2	DCI-RSP	Device Class Identification Response
28	2	UP-DIS	Upstream Transmitter Disable
30	3	INIT-RNG-REQ	Initial Ranging Request
31	3	TST-REQ	Test Request Message

IEEE 802.3 (Ethernet) Packetization

Ethernet 802.3 data packets are transferred across **wired or wireless** Ethernet links between 2 or more computers, between a cable modem (CM) and a computer (CPE), between a cable modem and a router and between a router and several computers. An Ethernet packet is of the form,



DA – 48 bit destination MAC address

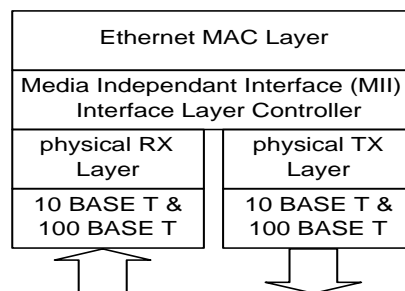
SA – 48 bit source MAC address

Type/Length – 16 bit Ethernet type or Length

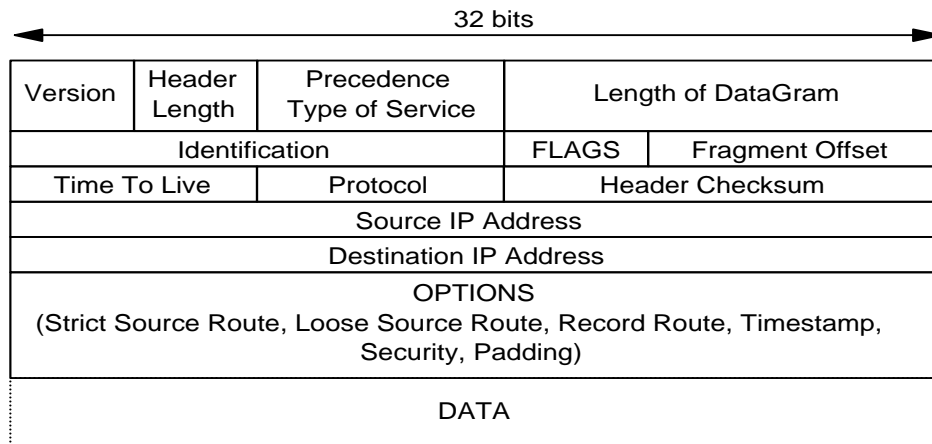
User Data – variable length, 0-1500 bytes

CRC – 32 bit CRC over the Ethernet frame

The Ethernet MAC layer processes the above Ethernet packets. This combined with a Media Independent Interface (MII) sits behind the Ethernet Physical Layer as shown.



Internet Protocol (IP) Packetization



Version/Header Length/Datagram Length - this gives the protocol version number, total length in bytes of the IP header and total length in bytes of the datagram.

Precedence and Type of Service – determines the priority level of the datagram and quality of service. This affects how the datagram is handled and processed at the destination.

Time-to-Live – this determines the amount of time the datagram will be allowed to live (exist) in the internet network before it is discarded.

Header Checksum – is the checksum computed on the total IP header. This checksum is re-calculated at every router because of the variable time-to-live field, fragmentation or optional fields.

Fragmentation – the Identification, flags and fragment offset fields enable IP data-grams to be fragmented and re-assembled at various points in the network.

Options – the options field can contain up to 40 bytes to specify different routing paths for an IP datagram (source to destination, or source to destination via several routers), timestamps and security. When IP data-grams pass from a source to a destination through several routers, the routers modify the IP source and destination addresses within the datagram to make the travel route transparent to the source and destination.

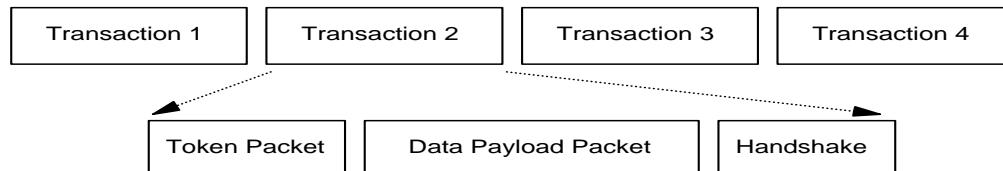
The **IP datagram** can be routed to any destination (Computer, SETTOP or PDA Network Interface Card - NIC) with a fixed or assigned (via DHCP) IP address. Once the datagram has reached its destination, the protocol field can be used to deliver it to the destination's client server (UDP or TCP/IP).

USB Packetization

The USB is a polled bus. The host controller (on a PC or SETTOP or gaming device) initiates all data transfers. All bus transactions invoke the transmission of up to 3 packets. Each transaction begins when the host controller, on a scheduled basis, sends a USB **token packet** describing,

1. the type and direction of the transaction
2. the USB device address
3. endpoint number

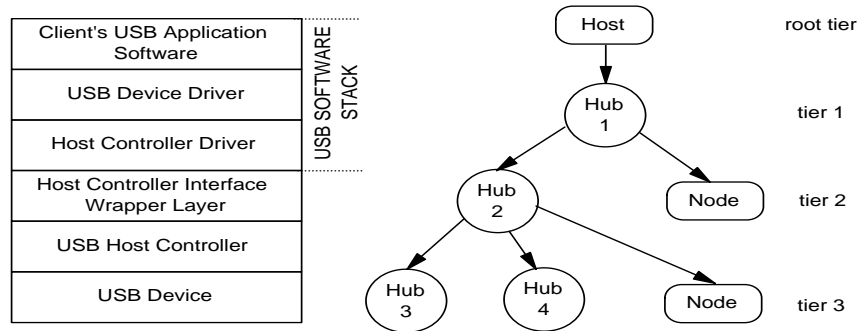
The USB device that is addressed selects itself by decoding the appropriate address fields. In any given transaction, data is transferred either from the host to a client device or from the client device to the host. This is specified in the token packet. The source of the transaction then sends a data packet or indicates it has no data to send. The destination then responds with a handshake indicating a successful or unsuccessful transfer.



The USB protocol generation is handled in the host controller. The host controller driver provides the specific instructions for the type of data to send and the destination address. The host controller is mainly responsible for the protocol

layer in a USB host when sending data to a USB device or reception of data from a USB device. It is responsible for adding the protocol packets (token and handshake) to a transaction or stripping off the protocol packets from a transaction.

The major functional blocks of a USB system and USB tiers are shown below,



The Client's USB application, USB device driver and the host controller driver form the USB software stack running on a PC or a SETTOP box in a typical USB system. The host controller and the USB physical device form the hardware part of the USB system. The host controller interface wrapper layer forms the translation layer between the USB software stack and the USB hardware.

The USB system has a tiered topology with only one host. Each segment is a point-to-point connection between a host and a client node, host and a hub, hub and a hub or hub and a client node. A USB system can have up to 127 nodes.

The 2 most popular USB systems in use today are USB 1.1 (bitrates up to 12 mbits/sec) and USB 2.0 (bitrates up to 480 mbits/sec) and are found on most computers, SETTOP boxes, digital video cameras, digital still cameras and gaming devices like the SONY playstation and Microsoft XBOX.

Notes:

1. **The User Program Guide (which allows a user to select different programs) in SETTOP boxes (for satellite, cable, terrestrial transmission) is built from information (descriptors) extracted from all the tables (PAT, PMT, BAT, NIT, EIT, SDT, TDT) found in each of those transmission services.**
2. **The User Navigation Guide (which allows a user to control and select presentation data) in a DVD is built from the Navigation Guide Data extracted from all the tables found in the Video Manager Information (VMGI), Video Title Set Information (VTSI), Presentation Control Information (PCI) and Data Search Information (DSI).**
3. **DOCSIS 2.0 is the latest Data-Over-Cable Standard in the US. In this the Upstream Channel can be TDMA or Spread-CDMA (refer to RFI spec, SP-RFiv2.0-I04-030730).**

4. **Serial Packetization Standards widely used in consumer electronics are,**
 1. **DVI (Digital Video Interface, which can encapsulate and carry raw video bitrates at High Definition Resolutions to provide a fast serial connection between computers/settop boxes/gaming devices and HD monitors).**
 2. **HDMI (High Definition Multi-Media Interface, which can encapsulate and carry HD raw video bitrates and several 16/20/24 bit PCM audio channels to provide a fast serial A/V interface between computers/settop boxes/gaming devices and HD multi-media TVs/monitors).**
 3. **PCI express (serial PCI bus) which will eventually replace the 32 bit PCI bus used in desktop/laptop computers.**

CONDITIONAL ACCESS (SECURITY) SYSTEMS

Conditional Access Systems are found in,

- SETTOP boxes
- DVD players & recorders
- Cable Modems.

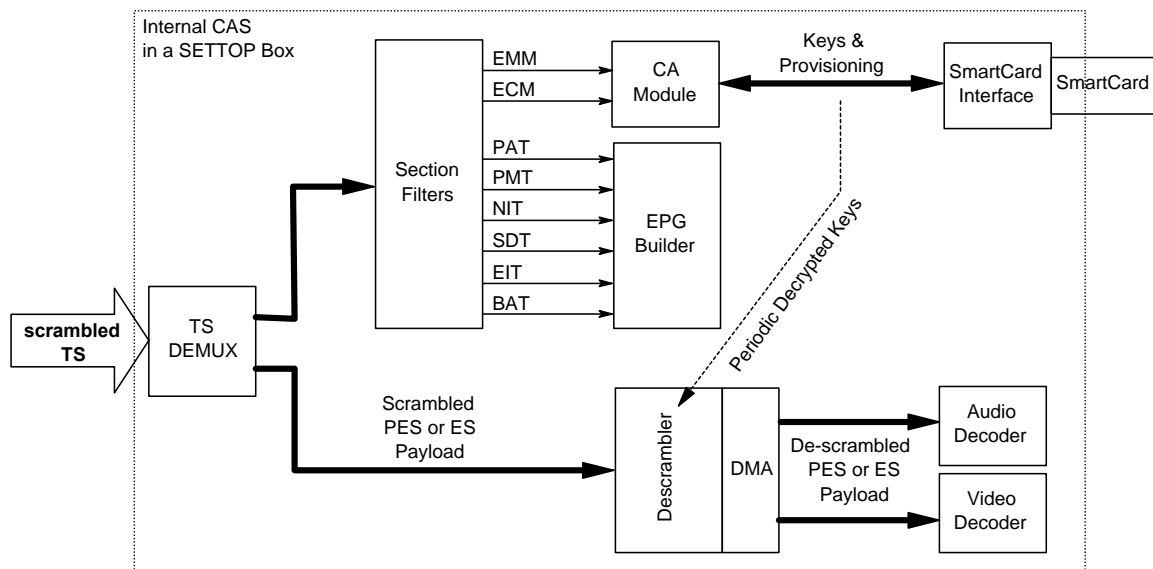
Conditional Access System typical in SETTOP Boxes

In a SETTOP box the CA system can be a separate or integrated module that provides limited access to some types of broadcast data (e.g. pay TV channels or pay per view programs) to each SETTOP box depending on whether that SETTOP box has been provisioned to receive it.

These types of conditional access systems have 2 parts,

- Key
- Descrambler

The **descrambler** using a certain pre-determined **symmetric** decryption algorithm (DES – data encryption standard, DVB – combination of block and stream cipher encryption, AES, Triple DES – DES algorithm recursed 3 times) descrambles the data (video/audio/data) payload fed to it using a **key**. The key or keys are usually sent along, encapsulated in the packetized broadcast data. These keys are generally valid for a certain time period (for better security) and are encrypted using **asymmetric** encryption algorithms (e.g. RSA). The key decryption is typically done using a secure micro running an asymmetric decryption algorithm on an external smartcard. The typical flow of a CA system in a SETTOP environment is shown below with the help of figure 83.



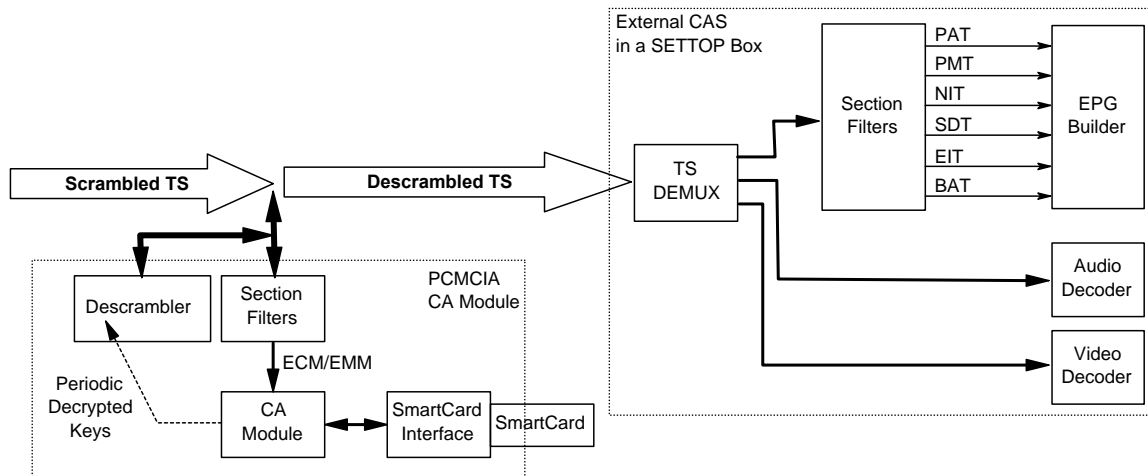


Figure 83. Internal and External CAS in a SETTOP Box Environment

The **first** flow diagram shows how a typical internal CAS works. Basically the descrambler and the conditional access module forming the CAS are implemented inside the SETTOP box chipset. The encrypted odd/even keys (or control words) for a specific scrambled A/V program are extracted from the corresponding EMM/ECM sections in the transport stream and decrypted via the secure micro on the external smartcard. These keys are first used to ARM the descrambler before sending it the scrambled A/V PES packets. The encrypted keys usually have a short life span (security purposes) in the range of a few micro-seconds. The software part of the conditional access module runs on the processor residing on the SETTOP box chipset and is responsible for parsing the encrypted keys from the ECM/EMM sections, sending them to the smartcard for decryption, receiving the decrypted keys, arming the descrambler and sending it the scrambled transport payload. This embedded processor is also responsible for extracting provisioning information about different A/V services from the EMM sections, interacting with the external smartcard to obtain information about which services the user is entitled to and then authorizing the descrambling of a specific A/V service.

The **second** flow diagram shows how a typical external CAS (shown here as a PCMCIA module) functions. Here the PCMCIA card contains all the CAS modules which include the embedded processor, descrambler, CA module, smartcard I/F and the smartcard. It sits directly in the path of the transport stream, extracts the transport packets corresponding to the scrambled A/V programs, descrambles the A/V programs as per the provisioning information, adds them back into the original transport stream and sends this new transport stream to the main chipset in the SETTOP box for further processing. This scheme is gaining popularity due to the myriad of CAS schemes (standardized and private) in the industry. A SETTOP box, PC or a LAPTOP enabled with a PCMCIA slot is ideal for implementing this kind of external CAS.

A typical MPEG2 transport multiplex contains 2 types of information which is used in any CAS module,

- Transport Program Payload (audio/video PES packets)
- Program Specific Information

Transport program payload is the scrambled/un-scrambled content (e.g. movies, news, soap operas, history or travel channels and pay per view programs) whereas program specific information (table/section data) shows the entire transport multiplex is put together.

The conditional access information is usually present in private sections (a table could be 1 or more sections) as follows,

- **EMM sections** – stores the entitlement messages or provisioning information
- **ECM sections** – stores the encrypted odd/even keys or control words

The CA software module is responsible for parsing the EMM sections to extract the provisioning information and provisioning services entitled to a user's smartcard. It is also responsible for parsing the ECM sections to extract the encrypted keys, feed them into the smartcard for decryption and then use them for arming the descrambler. The frequency of arrival of the EMM sections in a typical transport stream is in the order of seconds and that of the ECM sections is in the order of milliseconds.

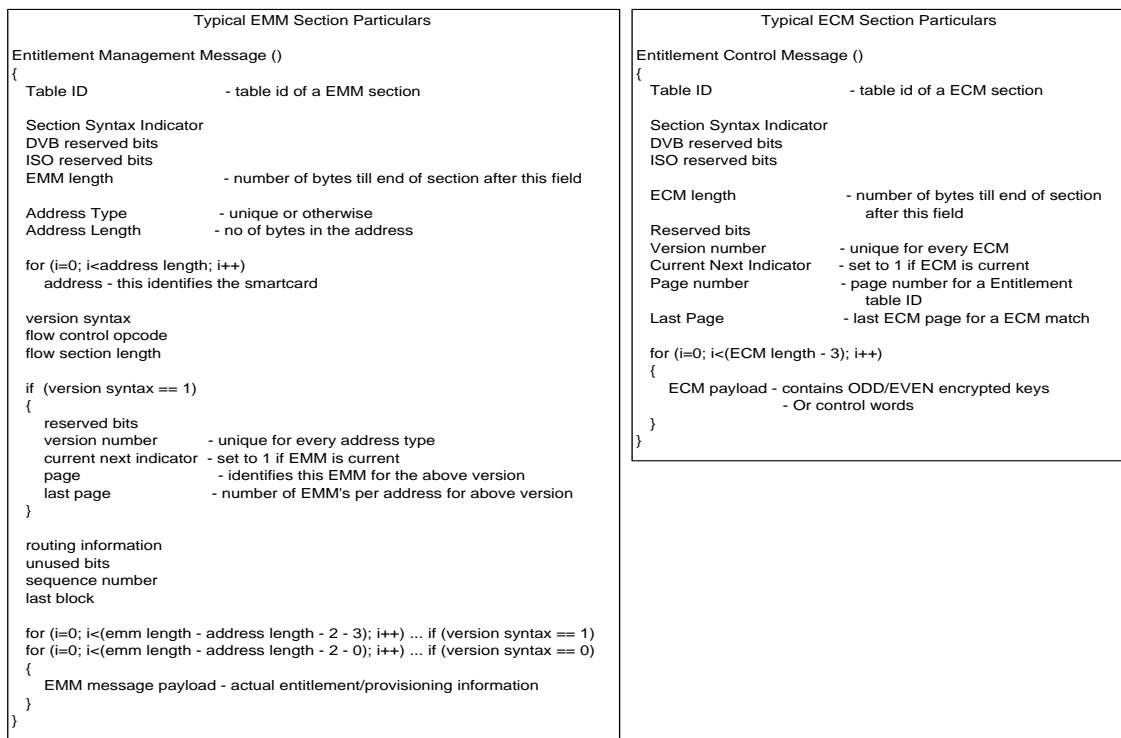


Figure 84. Typical EMM/ECM sections

Conditional Access System used in DVD Players/Recorders

Most DVD carrying movies are encrypted using a Content Scrambling System (CSS) to prevent unauthorized duplication of disc contents. This is achieved through encryption of the DVD contents (files) and storing the keys in hardware along with the encrypted contents (hardware being the DVD disk). CSS encryption is a 40 bit proprietary encryption algorithm.

CSS key sets are licensed to,

- DVD drive manufacturer's who incorporate them on their DVD drives (authentication keys)
- DVD disk manufacturer's who encrypt movie titles using these keys and store them on the DVD disc (title, disk, player keys)
- DVD player manufacturer's (player key - one key out of ~400 player keys)

DVD players have a CSS decryption module and DVD players/recorders have a CSS encryption (for encrypting and recording movies) and a decryption module.

CSS key is a collection of an authentication key, disc key, player key, title key, second disk key set, and/or encrypted key.

The CSS key are used as follows,

- Authentication is a process for a DVD drive and a CSS decryption module to recognize (or authenticate) each other. This step is necessary before reading data from DVD disks. Authentication keys are used for this process.
- Disk keys are used for decrypting title keys on a DVD disk.
- Player keys are used for decrypting disk keys on a DVD disk. Every DVD player manufacturer (software or hardware) is allocated one of approximately 400 player keys to incorporate in its DVD players.
- Title keys are used for scrambling and descrambling actual data on DVD disks called titles. A title could be complete motion picture, a movie trailer or a similar self contained unit.

DVD Disk Authentication

When a new DVD disk is inserted into a DVD drive, it must be authenticated before it can be used. This authentication uses a challenge-response protocol with CSS as the encryption method.

Once the disk authentication succeeds, any sector can be read from the DVD disk. The disk key data set stored on the disk is read immediately after the disk authentication is completed.

Decryption of Keys and Streams on the DVD Disk

A DVD sector contains 2048 bytes (the same as a CD-ROM sector). Each sector has an MPEG-2 PACK header which is followed by either stream data (MPEG-2 video, AC-3 audio, sub-picture, etc.) or other information (PCI and DSI) as we have shown before. If a sector contains stream data, the PACK header is followed by a stream header which contains 2 bits for the encryption type. These bits are set to 00 for no encryption, and 01 for CSS encryption.

Every DVD player is equipped with a player key. Every DVD disk has a disk key data block that is organized as follows,

- 5 bytes hash of decrypted disk key (hash)
- disk key encrypted with player key 1 (dk_1)
- disk key encrypted with player key 2 (dk_2)
-
-
- disk key encrypted with player key 400 (dk_{400})

Thus if a DVD player has a valid player key for slot 200, it will calculate the disk key (K_d),

$K_d = D_A (dk_{200}, K_{p200})$ where dk_{200} is the encrypted disk key with player key K_{p200}

To verify that the disk key calculated (K_d) is correct, the following check is done,

$K_d = D_A (\text{hash}, K_d)$ where hash is the 5 bytes decrypted disk key

The title key is decrypted with the decrypted/verified disk key and an additional key t_k ,

$K_t = D_B (t_k, K_d)$ where K_t is the title key, t_k is another key used in decrypting title keys

Each sector of the data files (video, audio, sub-picture) is encrypted using this title key K_t . All type of decryption is done using the CSS stream cipher primitive described next.

The CSS stream cipher is quite simple and is based on 2 LFSRs (linear feedback shift register) being added together to produce output bytes. There is no truncation, both LFSR are clocked 8 times for every byte output, and there are 4 ways of combining the output of the LFSRs to an output byte, thus supporting 4 modes for,

1. Disk Authentication
2. Decryption of Disk Key (D_A)
3. Decryption of Title Key (D_B)
4. Decryption of Data Blocks belonging to Stream Packs using Decrypted Title Keys

LFSR 1: 17 bits, is initialized by the 2 first bytes of key, and setting the most significant bit to 1 to prevent null cycling.

LFSR 2: 25 bits, is initialized with byte 3, 4, 5 of the key, and shifting all but the 3 least significant bits up 1 position, and setting bit 4 to prevent null cycling.

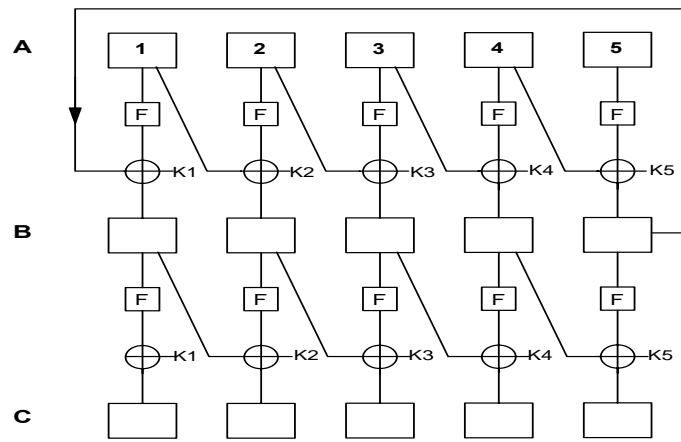
As new bits are clocked into the LFSRs, the same bits are clocked in with reversed order to the two LFSRs output bytes (with optional inversion of bits).

The output of LFSR1 is $O_1(1), O_1(2), O_1(3) \dots$

The output of LFSR2 is $O_2(1), O_2(2), O_2(3) \dots$

The final output is $O(i) = O_1(i) + O_2(i) + c \dots$ where c is carry bit from $O(i-1)$

When the CSS stream cipher is used to decrypt keys as in $D_A(\text{data}, \text{key})$ or $D_B(\text{data}, \text{key})$, an additional **mangling step** is performed on the data as illustrated in the next diagram,



1. $A(1,2,3,4,5)$ are the input bytes (data)
2. $C(1,2,3,4,5)$ are the output bytes (data)
3. $K_i = O(i)$, where $O(i=1,2,3,4,5)$ is the stream cipher output from the key
4. $B(1,2,3,4,5)$ are the temporary stages

The cipher is evaluated top down with exceptions indicated by arrows,

1. $B(j) = \text{xor} (F(A(j)), A(j-1), K_j) \dots$ where $j = 2,3,4,5$
2. $B(1) = \text{xor} (F(A(1)), B(5), K_1)$
3. $C(j) = \text{xor} (F(B(j)), B(j-1), K_j) \dots$ where $j = 2,3,4,5$
4. $C(1) = \text{xor} (F(B(1)), K_1)$

F is a function determined by a byte permutation table.

The detailed algorithms used in CSS decryption (all 4 modes) can be found in any text on copy protection for DVD video.

Conditional Access & Security used in DOCSIS 1.1/2.0 Cable Modems

The DOCSIS 1.1/2.0 spec uses a Conditional Access (+Security) System for Cable Modems called Baseline Privacy Plus (BPI+, refer to RFI specification SP-BPI+-I10-030730) which describes **MAC** layer security services. Its goal is to achieve the following,

1. Provides cable modem users with enhanced data privacy across the cable network by encrypting traffic flows between the CM and the CMTS.
2. Provides cable operators protection from theft of service.

The protected DOCSIS MAC data communications services fall into 3 categories,

1. Best-effort, high speed, IP data services
2. Quality of service (e.g. constant bit-rate) data services
3. IP multi-cast group services

Under BPI+, the CMTS protects against unauthorized access to these data transport services by enforcing encryption of the associated traffic flows across the cable network. BPI+ employs an authenticated client/server key management protocol in which the CMTS, the server, controls distribution of keying material to client CMs.

The original BPI specification (DOCSIS 1.0) had “weak” service protection because the underlying key management protocol did not authenticate CMs. BPI+ strengthens this service protection by adding digital certificate based CM authentication to its key management protocol.

Baseline Privacy Plus has **2 protocols**,

1. An **encapsulation protocol** for encrypting packet data across the cable network. This protocol defines,
 - The frame format for carrying encrypted packet data within DOCSIS MAC frames.
 - A set of supported *cryptographic suites*, i.e., pairings of data encryption and authentication algorithms.
 - The rules for applying those algorithms to a DOCSIS MAC frame’s packet data.
2. A **key management protocol** (Baseline Privacy Key Management, or “BPKM”) providing the secure distribution of keying data from CMTS to CMs. Through this key management protocol, CM and CMTS synchronize keying data. In addition, the CMTS uses the protocol to enforce conditional access to network services.

Packet Data Encryption

BPI+ encryption services are defined as a set of extended services within the DOCSIS MAC sub-layer. Packet Header information specific to BPI+ is placed in a Baseline Privacy Extended Header element within the MAC Extended Header.

BPI+ supports a single packet data encryption algorithm, the Cipher Block Chaining (CBC) mode of the Data Encryption Standard (DES) algorithm. BPI+

does not pair DES CBC with any packet data authentication algorithm. Additional data encryption algorithms may be supported in future enhancements to the BPI+ protocol specification, and these algorithms may be paired with data authentication algorithms.

BPI+ encrypts a DOCSIS MAC Frame's packet data, but the DOCSIS MAC Frame's Header is not encrypted. DOCSIS MAC management messages **MUST** be sent in the clear to facilitate registration, ranging, and normal operation of the DOCSIS MAC sub-layer.

Key Management Protocol

CMs use the Baseline Privacy Key Management protocol to obtain authorization and traffic keying material from the CMTS, and to support periodic reauthorization and key refresh. The key management protocol uses X.509 digital RSA certificates (a public-key encryption algorithm) and two-key triple DES to secure key exchanges between CM and CMTS.

The Baseline Privacy Key Management protocol adheres to a client/server model, where the CM, a BPKM "client", requests keying material, and the CMTS, a BPKM "server", responds to those requests, ensuring individual CM clients only receive keying material they are authorized for. The BPKM protocol uses DOCSIS MAC management messaging.

BPI+ uses public-key cryptography to establish a shared secret (i.e., an Authorization Key) between CM and CMTS. The shared secret is then used to secure subsequent BPKM exchanges of traffic encryption keys. This two-tiered mechanism for key distribution permits refreshing of traffic encryption keys without incurring the overhead of computation-intensive public-key operations.

A CMTS authenticates a client CM during the initial authorization exchange. Each CM carries a unique X.509 digital certificate issued by the CM's manufacturer. The digital certificate contains the CM's Public Key along with other identifying information, i.e., CM MAC address, manufacturer ID and serial number. When requesting an Authorization Key, a CM presents its digital certificate to a CMTS. The CMTS verifies the digital certificate, and then uses the verified Public Key to encrypt an Authorization Key, which the CMTS then sends back to the requesting CM.

The CMTS associates a cable modem's authenticated identity to a paying subscriber, and hence to the data services that subscriber is authorized to access. Thus, with the Authorization Key exchange, the CMTS establishes an authenticated identity of a client CM, and the services (i.e., specific traffic encryption keys) the CM is authorized to access.

Since the CMTS authenticates CMs, it can protect against an attacker employing a *cloned* modem, masquerading as a legitimate subscriber's modem. The use of the X.509 certificates prevents cloned modems from passing fake credentials onto a CMTS.

CMs MUST have factory-installed RSA private/public key pairs or provide an internal algorithm to generate such key pairs dynamically. If a CM relies on an internal algorithm to generate its RSA key pair, the CM MUST generate the key pair prior to its first Baseline Privacy initialization. CMs with factory-installed RSA key pairs MUST also have factory-installed X.509 certificates. Cable modems that rely on internal algorithms to generate an RSA key pair MUST support a mechanism for installing a manufacturer-issued X.509 certificate following key generation.

BPI+ Security Associations

A BPI+ *Security Association* (SA) is the set of security information a CMTS and one or more of its client CMs share in order to support secure communications across the cable network. BPI+ defines three types of Security Associations, *Primary*, *Static*, and *Dynamic*. A Primary Security Association is tied to a single CM, and is established when that CM completes DOCSIS MAC registration. Static Security Associations are provisioned within the CMTS. Dynamic Security Associations are established and eliminated, on the fly, in response to the initiation and termination of specific (downstream) traffic flows. Both Static and Dynamic SAs can be shared by multiple CMs.

A Security Association's shared information includes traffic encryption keys and CBC initialization vectors. In order to support, in future BPI+ enhancements, alternative data encryption and data authentication algorithms, BPI+ Security Association parameters include a cryptographic suite identifier, indicating the particular pairing of packet data encryption and packet data authentication algorithms employed by the security association. The 56-bit DES and 40-bit DES are the only packet data encryption algorithms supported, and neither of them are paired with a packet data authentication algorithm.

BPI+ identifies Security Associations with a 14-bit ***Security Association Identifier (SAID)***.

Each (BPI+ enabled) CM establishes an exclusive Primary Security Association with its CMTS. All of a CM's upstream traffic MUST be encrypted under the CM's exclusive, Primary Security Association. The SAID corresponding to a CM's Primary SA MUST be equal to the CM's Primary DOCSIS 1.1 Service ID (SID). On the other hand, while typically all downstream unicast traffic directed at CPE (Consumer Premises Equipment) device(s) behind the CM, are encrypted under the CM's exclusive Primary Security Association, selected downstream unicast traffic flows can be encrypted under Static or Dynamic SAs. That is, downstream

traffic MAY be encrypted under any of the three types of SAs. A downstream IP multicast data packet, however, is typically intended for multiple CMs and hence is more likely to be encrypted under Static or Dynamic SAs, which multiple CMs can access, as opposed to a Primary SA, which is restricted to a single CM.

The compliant CM MUST support a Primary SA, one or more Dynamic SAs, and one or more Static SAs. The compliant CMTS MUST support a Primary SA and one or more Dynamic SAs, and MAY support one or more Static SAs. BPI+ spec does not specify the number of the Static and Dynamic SAs required for the devices.

Using the BPKM protocol, a CM requests from its CMTS a SA's keying material. The CMTS ensures that each client CM only has access to the Security Associations it is authorized to access.

A SA's keying material (e.g., DES key and CBC Initialization Vector) has a limited lifetime. When the CMTS delivers SA keying material to a CM, it also provides the CM with that material's remaining lifetime. It is the responsibility of the CM to request new keying material from the CMTS before the set of keying material that the CM currently holds expires at the CMTS. The BPKM protocol specifies how CM and CMTS maintain key synchronization.

QoS SIDs and BPI+ SAIDs

The BPI+ Extended Header Element in downstream DOCSIS MAC frames contains the BPI+ SAID under which the downstream frame is encrypted. If the downstream frame is a unicast packet addressed to a CPE device behind a particular CM, the frame will typically be encrypted under the CM's Primary SA, in which case the SAID will be equal to the target CM's Primary SID. If the downstream frame is a multicast packet intended for receipt by multiple CMs, the extended header element will contain the Static or Dynamic SAID mapped to that multicast group. The SAID (Primary, Static or Dynamic), in combination with other data fields in the downstream extended header element, identifies to a receiving modem the particular set of keying material required to decrypt the DOCSIS MAC frame's encrypted Packet Data field.

Since all of the CM's upstream traffic is encrypted under its unique Primary SA, upstream DOCSIS MAC Frames, unlike downstream DOCSIS MAC Frames, need not carry a BPI+ SAID in their extended headers, instead, the Baseline Privacy EH element MAY contain any valid QoS SID assigned to the CM.

The Baseline Privacy extended header element serves multiple purposes in upstream DOCSIS Packet Data PDU MAC Frames. In addition to identifying the particular set of keying material used to encrypt the Frame's packet data, it also provides a mechanism for issuing piggybacked bandwidth requests, and it can carry fragmentation control data. These later two functions are tied to a particular

QoS SID, for this reason, upstream Baseline Privacy Extended Header Elements contain a QoS SID rather than a BPI+ Primary SAID, which can be inferred from the QoS SID.

SYMMETRIC & ASYMMETRIC ENCRYPTION/DECRYPTION ALGORITHMS

- **Symmetric algorithms** are the most common type of encryption algorithms. They are known as "symmetric" because the **same key** is used for both encryption and decryption. Unlike the keys used with public-key algorithms, symmetric keys are **frequently changed** (have a short life span). For this reason, they are also referred to as **session keys** (valid for a time-dependant session). Compared to public-key algorithms, **symmetric algorithms are very fast** and thus are preferred when encrypting large amounts of data (typical in radio, satellite and cable transmissions). Some of the more common symmetric algorithms are RC2, RC4, RC5, Data Encryption Standard (DES), Advanced Encryption Standard (AES) & the DVB super scrambling scheme.

The most commonly used symmetric algorithms are,

1. DES encryption/decryption (Block Cipher using the Electronic Code Book and Output Feedback Modes).
 2. DVB encryption/decryption (Super Scrambling Scheme for PES or ES packets using a combination of Block and Stream Ciphers).
 3. AES encryption/decryption (Advanced Encryption Standard which is a not a truly symmetric algorithm).
- **Public-key (asymmetric) algorithms** use a pair of different keys: a **public key** and a **private key**. The private key is kept private to the owner of the key pair, and the public key can be distributed to anyone who requests it. If one key is used to encrypt a message, the other key is required to decrypt the message. **Public-key algorithms are very slow**, on the order of a thousand times slower than symmetric algorithms. **Consequently, they are normally used only to encrypt session keys**. They are also used to digitally sign messages.

One of the most common public-key asymmetric algorithms is the **RSA Public-Key Cipher**, which we will discuss in this chapter.

Symmetric Encryption/Decryption Algorithms

- **Data Encryption/Decryption Standard (DES)** – in figure 85, we have shown the DES encryption/decryption mechanism which is a block based encryption scheme.

During **encryption** the stream of bits (plain-text) to be encrypted is segmented into blocks of 64 bits. Each 64 bit block is passed through an initial permutation, and is then split into 2, 32 bit blocks (Lo & Ro) which are

input into the transformation section. The transformation consists of 16 stages, where the data in each stage is scrambled using a sub-key (generated from the encryption key). The output of the last stage is subjected to a permutation, which is the inverse of the initial permutation. The resulting 64 bits are the cipher-text.

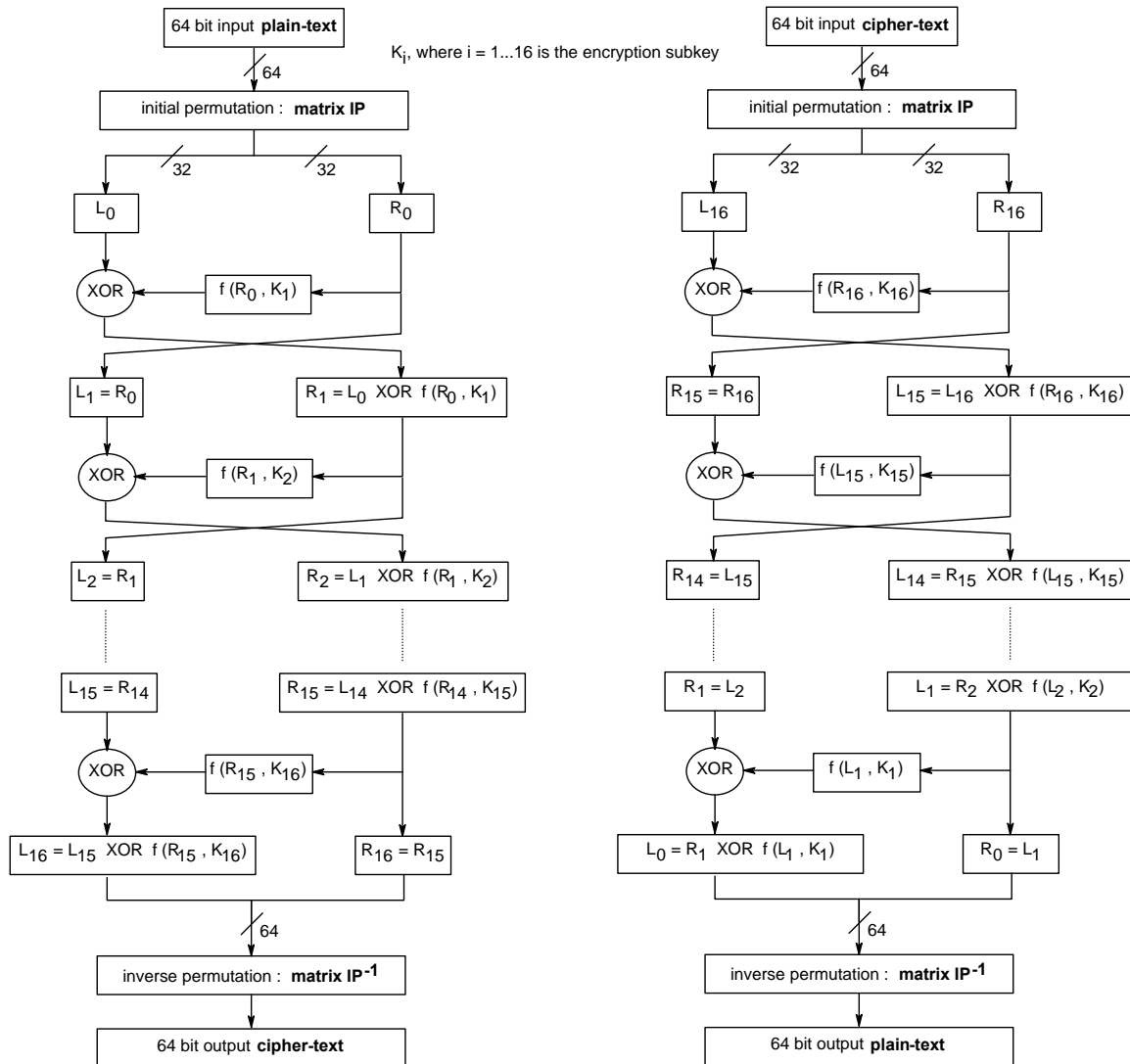


Figure 85. DES Encryption/Decryption Flow Diagrams

During decryption the stream of bits (cipher-text), it is again segmented into 64 bit blocks and the same procedure used in encryption is applied. The only difference being the transformation block is modified. The left and right 32 bit word locations are interchanged and the algorithm starts with the encryption sub-key 16 and ends with sub-key 1.

The matrix IP and IP^{-1} manipulate the bits in the 64 bit input to provide a new 64 bit input for additional security purposes.

The function $f(R, \text{SUBKEY})$ or $f(L, \text{SUBKEY})$ and the generation of the 56 bit sub-keys from the 64 bit encryption key are shown in figure 86.

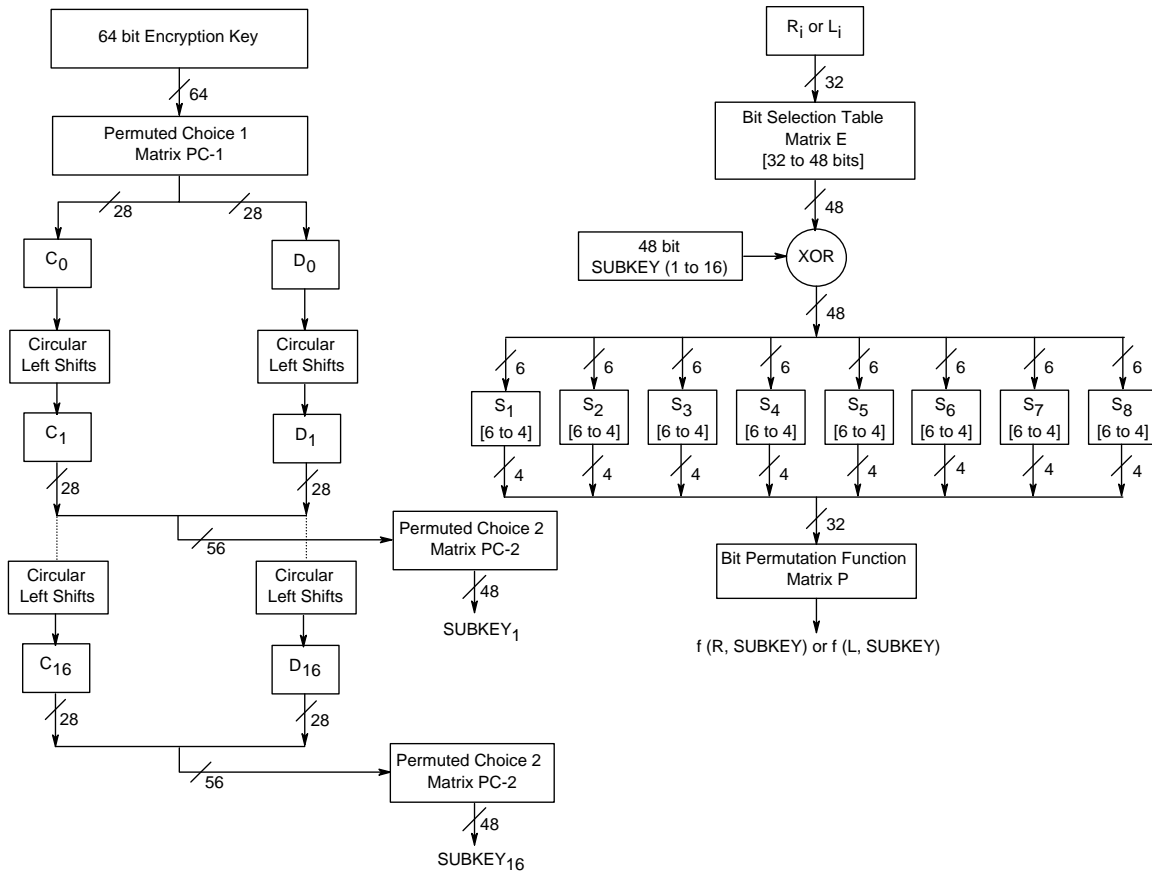
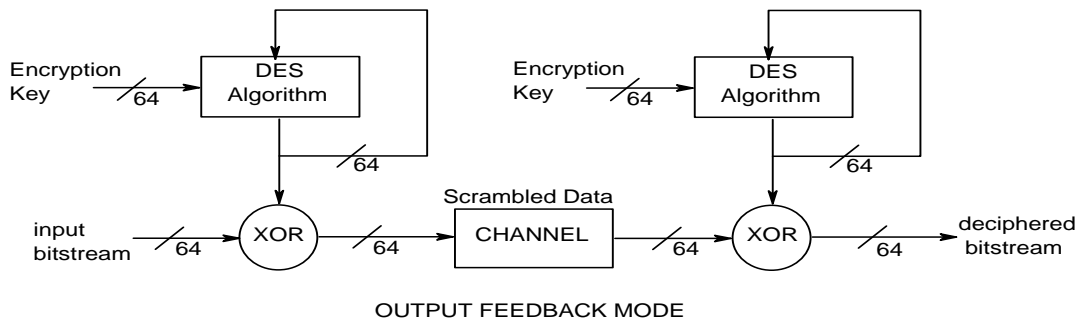
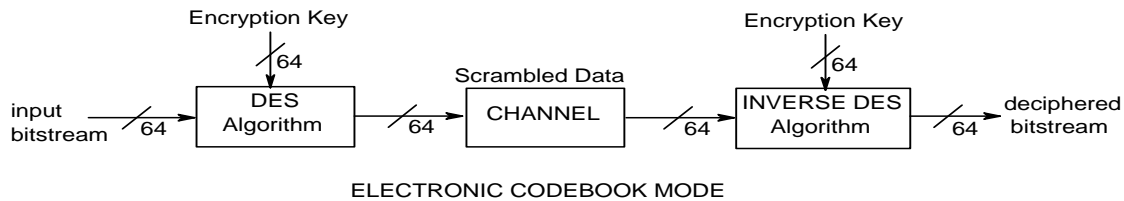


Figure 86. $f(R, \text{SUBKEY})$ or $f(L, \text{SUBKEY})$ and SUBKEY Generation

Sub-keys are generated from the main 64 bit encryption key by passing this key through a permutation matrix PC-1 which permutes 56 bits from 64 bits. The 56 bits are then split into 2, 28 bit pieces (C_i & D_i) which are then circularly left shifted by x bits depending on the iteration i , sent through another permutation matrix PC-2 which permutes 48 bits from the 56 bits to provide a SUBKEY _{i} .

The function for the i^{th} iteration takes a 32 bit R/L value, passes it through a bit selection table to generate 48 bits. This 48 bit value is then XOR-ed with the i^{th} iteration SUBKEY and split into 8, 6 bit numbers. Each 6 bit number is passed into a S-box which converts a 6 bit sequence to 4 bits. The 4 bit outputs from each of the 8 S-boxes are combined into a 32 bit value which is then passed into a permutation matrix to complete the function $f(R_i \text{ or } L_i, \text{SUBKEY}_i)$. The DES algorithm can be implemented in Electronic Codebook (ECB) mode or in Output Feedback (OFB) mode.

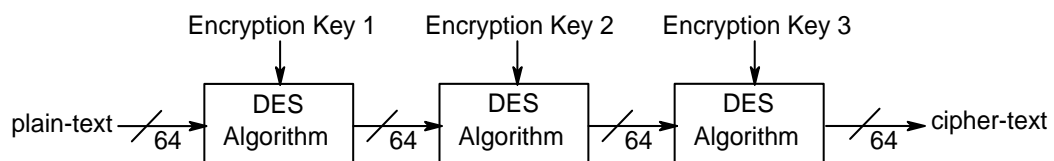
In Electronic Codebook mode the plain-text is encrypted by the 64 bit encryption key and the DES algorithm to give the cipher-text at the transmitter. This cipher-text is then sent over the channel. This cipher-text at the receiver is decrypted with the identical 64 bit encryption key and the inverse DES algorithm to give back the plain-text.



In output feedback mode the 64 bits of the output of the DES algorithm are OR-ed with the 64 bits of the input bitstream to generate the scrambled/encrypted bitstream for transmission across the channel. The same 64 bits which are the output of the DES algorithm are fed back as input to the DES algorithm for the next iteration (the 1st iteration is initiated by a fixed input block). The receiver undergoes the same operation as the transmitter using the identical fixed input block for the very 1st iteration.

The advantage of the output feedback mode over the electronic code book mode is that it confines any transmission errors to single bits rather than a block of 64 bits.

- **Triple DES** is a method in which the DES algorithm is passed 3 times over the plain-text for additional security or robustness. The order could be encrypt-decrypt-encrypt or decrypt-encrypt-decrypt and the keys used for encryption/decryption could be different or identical.



- **DVB Encryption/Decryption using Super-Scrambling** – this super scrambling (combination of block & stream ciphers) method was specifically

developed for scrambling A/V MPEG2 PES (transport level scrambling) or ES (PES level scrambling) packets by the DVB committee.

The **block cipher** works on blocks of 8 bytes of data in the reverse cipher block chaining mode (RCBC). The **stream cipher** works as a pseudo random byte generator. The stream cipher protects the outputs (encryption) and the inputs (decryption) of the block cipher to reduce the probability of a fatal attack. Both these ciphers are designed to work together.

Figure 87, shows a typical DVB super-scrambling mechanism applied to MPEG2 transport packets.

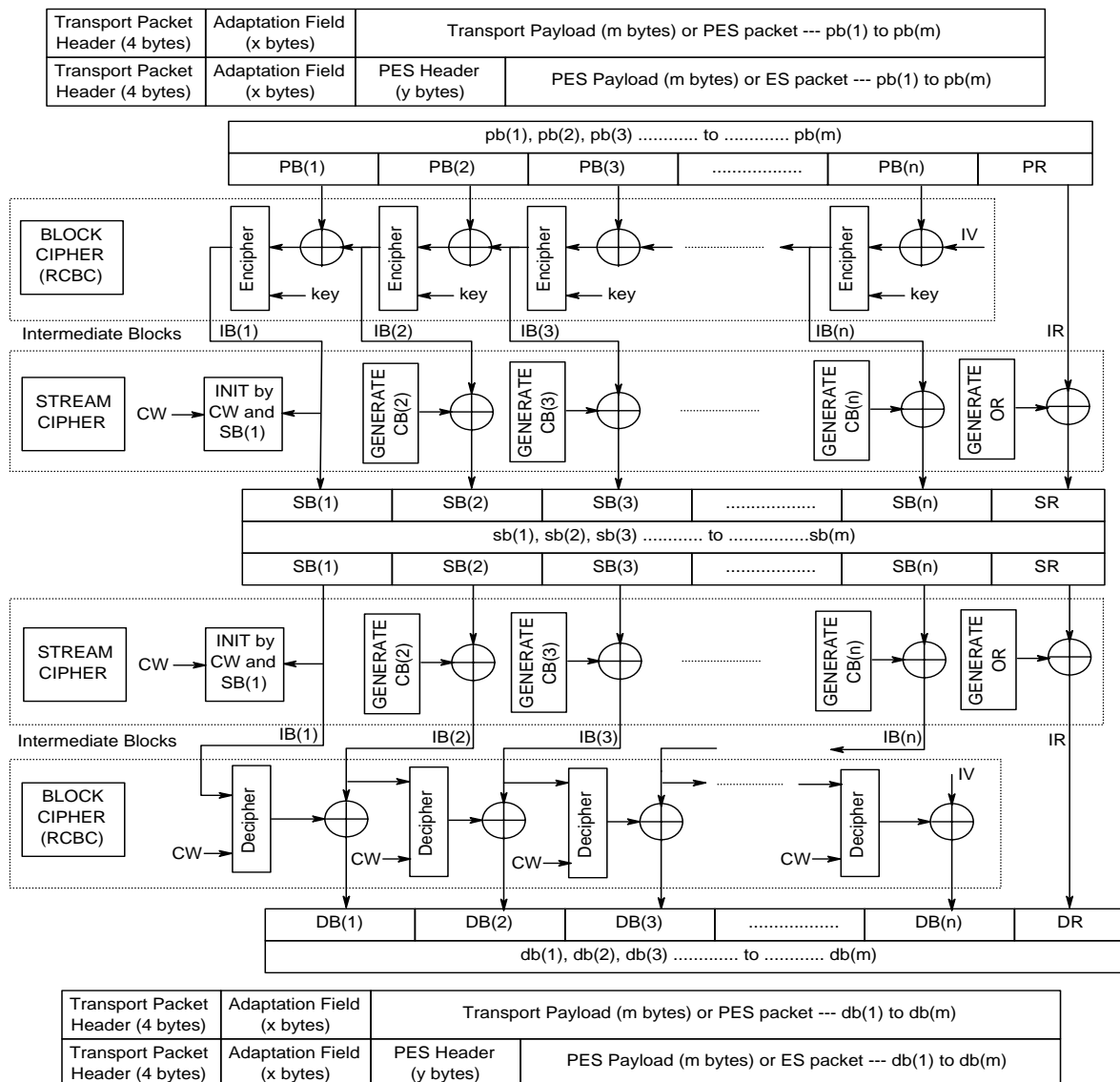
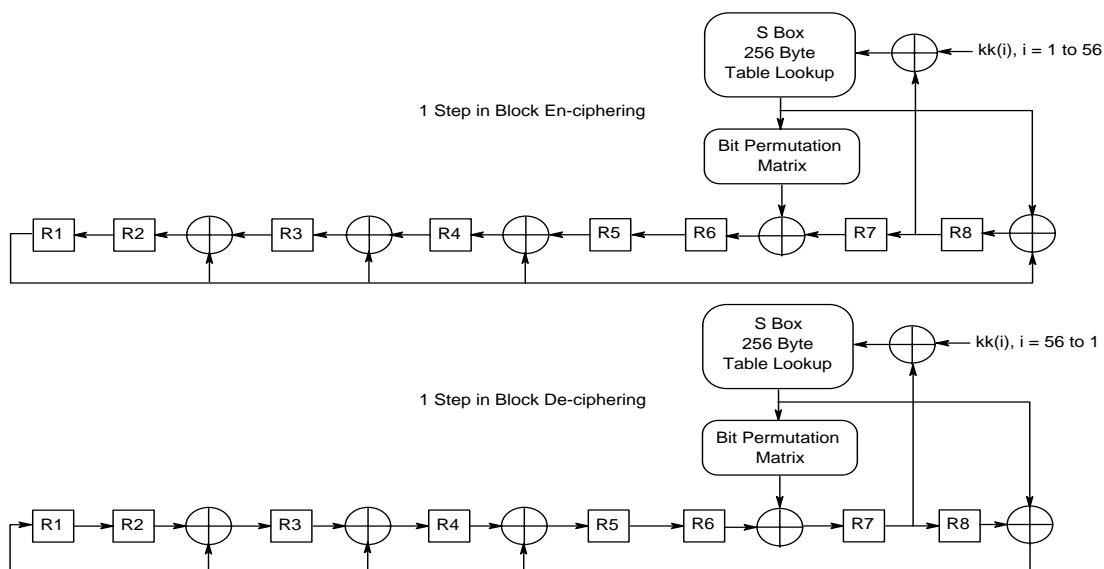


Figure 87. DVB Super-Scrambling Encryption/Decryption Method

The unscrambled transport or PES payload consisting of “m” consecutive bytes is denoted as pb(1), pb(2),...,pb(m). The maximum value of m is 184 bytes. This field of m bytes is split into “n” consecutive blocks of 8 bytes denoted as PB(1), PB(2),...,PB(n). The maximum value of n is 23. If m bytes are not a multiple of 8, then the payload has a plain residue (PR). Thus $m = (8 * n) + r$. The value of “IV” is 0. The first ciphered block is not affected by the stream cipher which uses it for initialization along with the 8 byte control word (CW). If a residue (PR) is present it is not affected by the block cipher. The final scrambled payload also consists of “m” consecutive bytes denoted as sb(1), sb(2),.....,sb(m) which are grouped as SB(1), SB(2),.....,SB(n) and SR.

At the receiver each scrambled field consisting of sb(1), sb2,...,sb(m) bytes is regrouped into scrambled blocks SB(1), SB(2),...,SB(n) where each block is 8 bytes wide. The residual scrambled bytes of this field are denoted SR. The value of "IV" is 0. The first scrambled block SB(1) is not affected by the stream cipher which uses it for initialization along with the 8 byte CW. The scrambled residue SR if present does not go through the block cipher, but is descrambled with the stream cipher.

Block Cipher – processes one 8 byte block in 56 steps of operations. The input block of 8 bytes is loaded into a block register R consisting of 8, byte wide registers (R1 to R8). The algorithm is performed in 56 steps of operations. At the end of the 56 steps, the output from the block register R is the ciphered block. Each one of the 56 steps performs operations on the 8 registers using a single key byte **kk(i)**. The sequence of 56 key bytes are generated from the 8 byte control word using a **key expansion scheduling technique** (uses 2 permutation matrices and 7 rounds of 8 steps to generate 56 key bytes from the 8 byte CW). During enciphering the key sequence used is kk(1) to kk(56), and during deciphering the key sequence is reversed as kk(56) to kk(1).

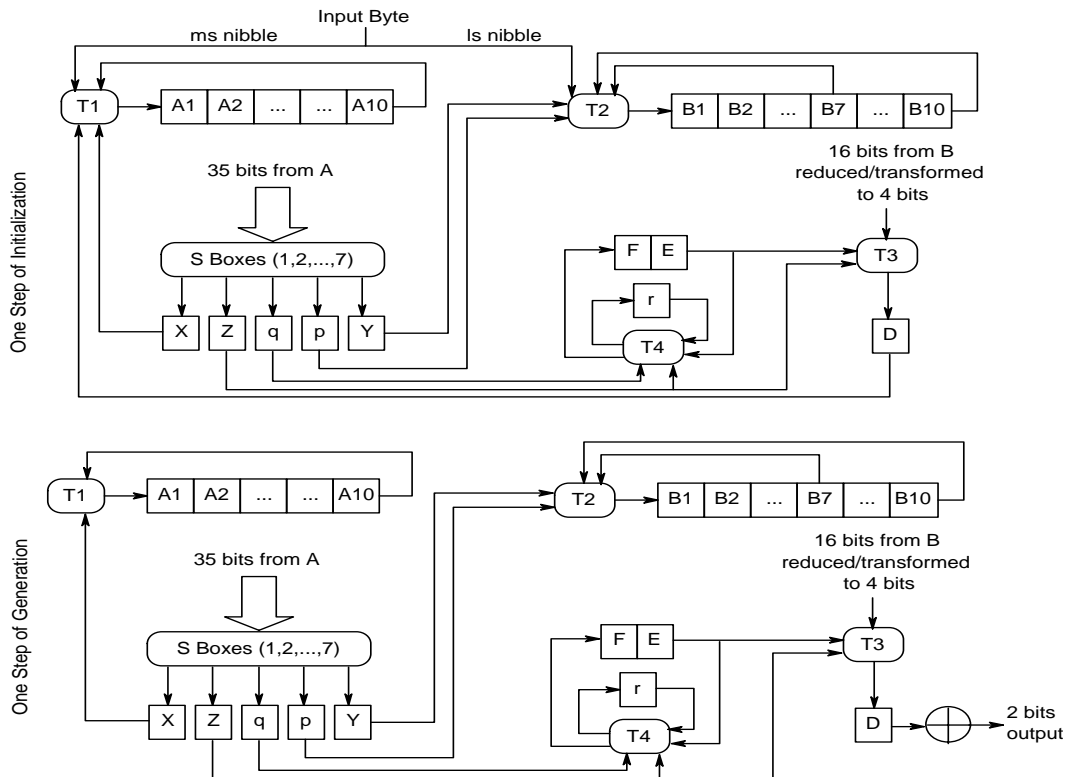


Stream Cipher – is a finite state machine. Its internal states consist of various registers with 107 bits organized as 26 nibbles and 3 bits. The bits are organized as follows,

1. 10 nibbles in register A, denoted as A1 to A10
2. 10 nibbles in register B, denoted as B1 to B10
3. 1 nibble in each register X, Y, Z, D, E, and F
4. 1 bit in each register p, q, and r

In the following state flow diagram we have shown 1 initialization stage and 1 generation stage of the stream cipher (identical to stream decipher). The transformation boxes in this flow diagram do the following,

1. T1 – takes input nibbles, XOR-s them together to generate 1 output nibble
2. T2 – takes input nibbles, XOR-s them together to generate 1 output nibble
3. T3 – takes input nibbles, XOR-s them together to generate 1 output nibble
4. T4 – takes its inputs and transforms them to generate a total of 5 bits (nibble+carry)
5. S – is a transformation that maps 5 bits to 2 output bits



A reset step is performed before using the stream cipher/de-cipher. The 1st 32 bits of the 8 byte control word (CW) are loaded in the state registers A1 to A8 and the last 32 bits of the CW are loaded into the stage registers B1 to B8. The other state registers are A9, A10, B9, B10, D, E, F, X, Y, Z, p, q, and r, are set to 0.

Each byte of an 8 byte block SB(i) is loaded in 4 steps of initializations. The 1st and 3rd steps enter the byte sb(i). The 2nd and 4th steps enter the byte

resulting from swapping the 2 nibbles of sb(i). Thus each block SB(i) is loaded in 32 steps of initialization as a sequence of 32 input bytes. At each initialization step, the input byte is separated into nibbles and fed into separate transformations.

At each step of the generation there is no input byte and no feedback of the nibble in register D. The 4 bits of register D are XOR-ed to produce 2 output bits. Thus 4 consecutive steps of generation are needed for producing 1 output (stream ciphered/de-ciphered) byte.

- AES Encryption/Decryption Standard** – the Advanced Encryption Standard is a block based ciphering technique (similar to the DES algorithm in ECB mode) which converts a 128 bit block of plain-text to a 128 bit block of cipher-text. The AES algorithm uses one of 3 cipher key strengths, 128, 192 or 256 bits. Each encryption key size causes the algorithm to behave differently. Increasing key sizes increases the algorithm complexity as well as allows for a bigger key size to encrypt data. This algorithm repeats its core function a number of times (rounds or iterations) depending on the key size and the ciphering technique also has pre-round and post-round operations. This algorithm is not truly symmetric like the DES, since the operations for the deciphering operation are the inverse of the ciphering operation and are performed in a different order.

Key Expansion and Scheduling Technique

The AES algorithm expands the initial encryption key into a table of 32 bit values. The algorithm then sub-divides the table into groups of 4, 32 bit values.

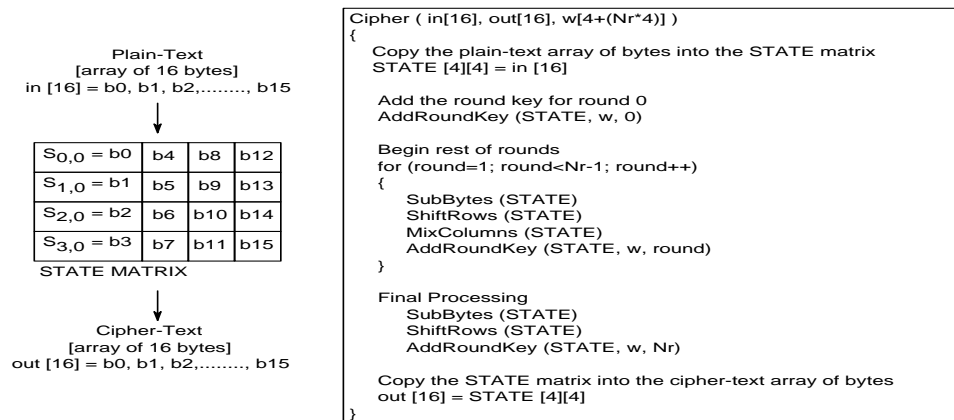
<pre> Key Expansion Algorithm (key[], w[4*Nr*4], key_size) { Copy the key into the 1st values of key schedule for (i = 0; i < key_size; i++) { Form 32 bit words from 4 consecutive key bytes w[i] = word [key [4*i], key [4*i + 1], key [4*i + 2], key [4*i + 3]] } Expand the rest of key schedule values for (i = key_size; i < (4 + Nr*4); i++) { temp = w [i-1] if (i mod key_size == 0) temp = SubWord (RotateWord(temp)) xor Rcon (i/key_size) else if (key_size > 6) and (i mode key_size == 4) temp = SubWord (temp) w [i] = w [i-key_size] xor temp } } </pre>	<p>key [] - is a array of bytes which contains the encryption key</p> <p>key_size - is the size of the key array divided by 4, 32 bit words</p> <p>w - is the final key-schedule array</p> <p>Nr - is the number of rounds or iterations depending on key_size</p> <p>SubWord - is a byte by byte substitution of a 32 bit word using a S-box look up table</p> <p>RotWord - rotates the bytes in a 32 bit word</p> <p>Rcon [i] - is a look up table value, that the word formed from the byte wide values, $(2^{i-1}, 0, 0, 0)$ provides</p> <p>One more thing worth noting is that the Key Expansion Algorithm calculates the 1st byte as an exponential within a "Galois Field"</p> <p>If $GF(2^8)$ is a bit-wide galois field, then a byte {b} can be shown as the below polynomial,</p> $\{b\} = b_7x^7 + b_6x^6 + \dots + b_1x^1 + b_0$ <p>if $b = 0x3A - \{0011\ 1010\}$, then $b = x^5 + x^4 + x^3 + x^1$</p> <p>Any addition or subtraction between any 2 bytes is an XOR operation. A multiplication between 2 bytes is done via polynomial multiplication with the result always constrained to a 8 bit value, due to the $GF(2^8)$. Thus any multiplication is done modulo a fixed polynomial, which for the AES algorithm is $\{x^8 + x^4 + x^3 + x^1 + 1\}$</p>
--	---

Each round (iteration) of the AES algorithm uses a key made up of 4, 32 bit values from the key schedule. The total number of scheduled keys, and thus the total table size, depends on the initial key size. A scheduled key consists of 4, 32 bit values for each iteration of the algorithm, and one final set of 4, 32 bit values.

For a 128 bit key, there are 10 rounds, thus the table of 32 bit values generated during key expansion has a size of 44. For a 192 bit key, the table size is 52, 32 bit values and for a 256 bit key, the table size is 60, 32 bit values.

AES Encryption

The AES encryption algorithm converts 128 bits of plain-text to 128 bits of cipher-text. The main encryption processing loop first moves the plain-text (array of 16 bytes) to a 4x4 matrix called a "STATE". Further sub-operations of the AES algorithm are done on this STATE matrix. After all the sub-operations are complete the main encryption loop moves the contents in the state matrix back to a cipher-text array in the identical byte order. The number of rounds (iterations) depends on the key size (Nr=10 for 128 bit key, Nr=12 for 192 bit key and Nr=14 for a 256 bit key). The main sub-operations are AddRoundKey(), SubBytes(), ShiftRows() and MixColumns().



This function performs an XOR of each column in the STATE matrix with a 32 bit value from the key schedule for that round. The column number "c" acts as an offset into the key schedule for that specific round to address the 32 bit key.

$\{S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}\} \text{ XOR } w[\text{round} \cdot 4 + c]$

AddRoundKey Function

This function substitutes each byte in the STATE matrix with a new value. Each byte in the STATE matrix is split into 2, 4 bit nibbles. Each of these nibbles are used to address a 2D 16x16 SBOX byte lookup table as follows

$S' = \text{SBOX} [S_{\text{high}}] [S_{\text{low}}]$

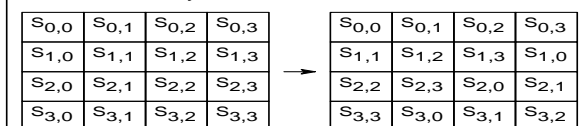
SubBytes Function

This mixes the values for any given column in the STATE matrix with other values from the same column. This operation uses the concept of multiplying 2 numbers in a Galois Field GF(2⁸). This function multiplies every column in the STATE matrix by a 4x4 matrix within the GF.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

MixColumns Function

This function switches the order of column entries for a single row in the STATE matrix. This function acts independently upon each row. Depending on the row number the columns are shifted circularly x times.



ShiftRows Function

AES Decryption

The AES decryption (de-cipher) algorithm converts the ciphered text produced by the AES encryption algorithm to give back the original plain-text. The key expansion and scheduling technique remains the same, but the keys scheduled are used in the reverse (opposite) order in the AddRoundKey() sub-operation.

The deciphering starts by converting the 16 byte cipher-text block array into a 4x4 STATE matrix and ends by converting this STATE matrix back into a 16 byte plain-text block array (similar to the cipher). Also the deciphering algorithm performs the same number of rounds (iterations) as the ciphering algorithm. The main sub-operations InvSubBytes(), InvMixColumns() and InvShiftRows() are basically the inverse of the ciphering sub-operations with a few caveats.

```

InvCipher ( in[16], out[16], w[4+(Nr*4)] )
{
    Copy the cipher-text array of bytes into the STATE matrix
    STATE [4][4] = in [16]

    Add the round key for the last round
    AddRoundKey (STATE, w, Nr)

    Begin rest of deciphering rounds
    for (round=Nr-1; round>0; round--)
    {
        InvShiftRows (STATE)
        InvSubBytes (STATE)
        AddRoundKey (STATE, w, round)
        InvMixColumns (STATE)
    }

    Final Processing Operations
    InvShiftRows (STATE)
    InvSubBytes (STATE)
    AddRoundKey (STATE, w, 0)

    Copy the STATE matrix into the cipher-text array of bytes
    out [16] = STATE [4][4]
}

```

This mixes the values for any given column in the STATE matrix with other values from the same column. It uses the same concept of multiplying 2 numbers in a Galois Field $GF(2^8)$ as the cipher. The only difference being a "different" 4x4 constant matrix is used.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

InvMixColumns Function

This function switches the order of column entries for a single row in the STATE matrix, to reverse the effect of the ShiftRows function in the cipher.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

 \rightarrow

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,3}$	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,0}$

InvShiftRows Function

This function uses a different 2D 16x16 SBOX byte lookup table than the ciphering SubBytes function.

$$S' = \text{SBOX} [S_{\text{high}}] [S_{\text{low}}]$$

InvSubBytes Function

Public-Key Asymmetric Encryption/Decryption (using RSA algorithm)

The RSA algorithm is named after its creators (Ron Rivest, Adi Shamir and Len Adleman). The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the computation difficulty of factoring large integers.

Key Generation Technique

1. Generate two large random primes, p and q , of approximately equal size such that their product $n = p \cdot q$ is of the required bit length, e.g. 1024 bits.

To generate the primes p and q , generate a random number of bit length $b/2$ where b is the required bit length of n ; set the low bit (this ensures the number is odd) and set the two highest bits (this ensures that the high bit of n is also set); check if number is a prime; if not, increment the number by two and check again. This is p . Repeat for q starting with an integer of length $b-b/2$. If $p < q$, swap p and q (this only matters if you intend using the CRT form of the private key). In the extremely unlikely event that $p = q$, check your random number generator. For greater security, instead of incrementing by 2, generate another random number each time.

2. Compute $n = p \cdot q$ and $\phi = (p-1) \cdot (q-1)$.
3. Choose an integer e , $1 < e < \phi$, such that $\text{GCD}(e, \phi) = 1$.

In practice, common choices for e are 3, 17 and 65537. These are Fermat Primes and are chosen because they make the modular exponentiation operation faster. Also, having chosen e , it is simpler to test whether $\text{GCD}(e, p-1)=1$ and $\text{GCD}(e, q-1)=1$ while generating and testing the primes in step 1. Values of p or q that fail this test can be rejected there and then.

4. Compute the secret exponent d , $1 < d < \phi$, such that $e \cdot d = 1 \pmod{\phi}$.

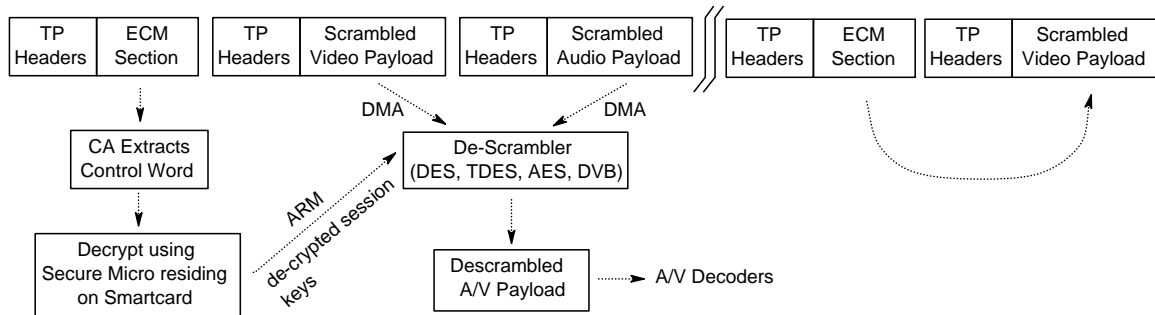
To compute the value for d , use the *Extended Euclidean Algorithm* to calculate $d = e^{-1} \pmod{\phi}$ (this is known as *modular inversion*).

5. The public key is (n, e) and the private key is (n, d) . The values of p , q , and ϕ should also be kept secret.

n is the **modulus**. e is the **public exponent or encryption exponent**. d is the **secret exponent or decryption exponent**.

Typical Applications

- Encryption of time-dependant session keys (using a public key) embedded in a transport multiplex in a satellite, cable or terrestrial feed. The user's smartcard could store the private key (hidden key) for decrypting these session keys. These decrypted session keys can then be used to arm the descrambler (which may use a symmetric algorithm like DES, TDES, DVB super scrambling, AES, etc.) for descrambling the scrambled transport payload or PES payload.



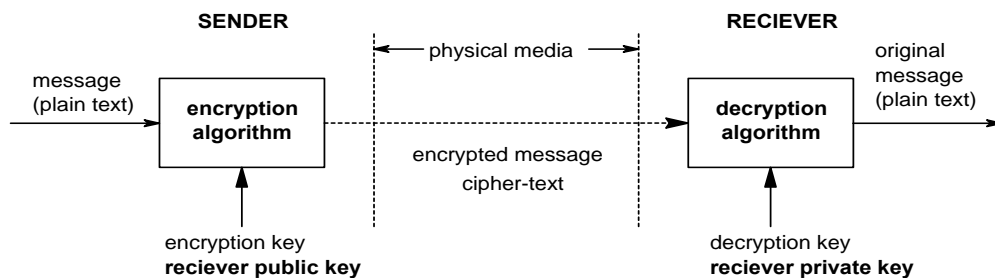
- Sending messages securely between a sender and the recipient.

At the encryption end, the sender does the following,

1. Obtains the recipient's public key (n , e).
2. Represents the plain-text message as a positive integer $m < n$.

When representing the plaintext bytes as the representative integer m , it is usual to add random padding characters to make the size of the integer m large and less susceptible to certain types of attack. If $m = 0$ or 1 there is no security as the cipher-text has the same value.

3. Computes the cipher-text $c = m^e \bmod n$.
4. Sends the cipher-text c to recipient.



At the decryption end, recipient does the following,

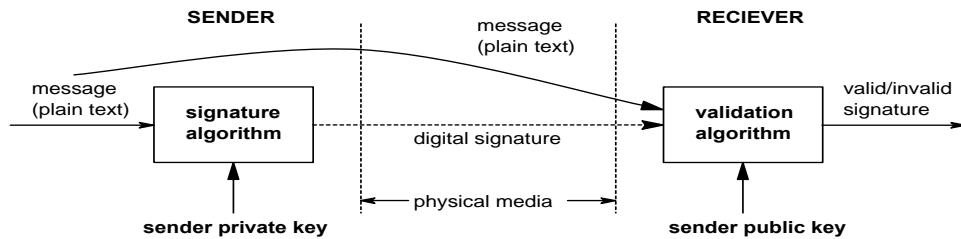
1. Uses his private key (n , d) to compute $m = c^d \bmod n$.
 2. Extracts the plaintext from the integer representing m .
- Digital signing of documents – the next figure shows how a document can be digitally signed and exchanged between a sender and a recipient, thus making sure it originated from the sender.

Sender does the following to digitally sign a message,

1. Creates a *message digest* of the information to be sent.
2. Represents this digest as an integer m between 0 and $n-1$.

Decryption and Signing are identical as far as the mathematics is concerned. Both use the private key. Similarly, Encryption and Verification both use the same mathematical operation with the public key.

3. Uses her *private* key (n, d) to compute the signature, $s = m^d \bmod n$.
4. Sends this signature s bundled with the message digest m to the recipient.



Recipient does the following for digitally verifying the signed message,

1. Uses sender's public key (n, e) to compute integer $v = s^e \bmod n$.
2. Extracts the message digest from this integer.
3. Compares this message digest with what the sender has sent. If both message digests are identical, then the signature is valid.

TYPICAL DVB SETTOP BOX APPLICATION

A DVB SETTOP box compatible with MPEG2 transport streams has the following components as shown in figure 88.

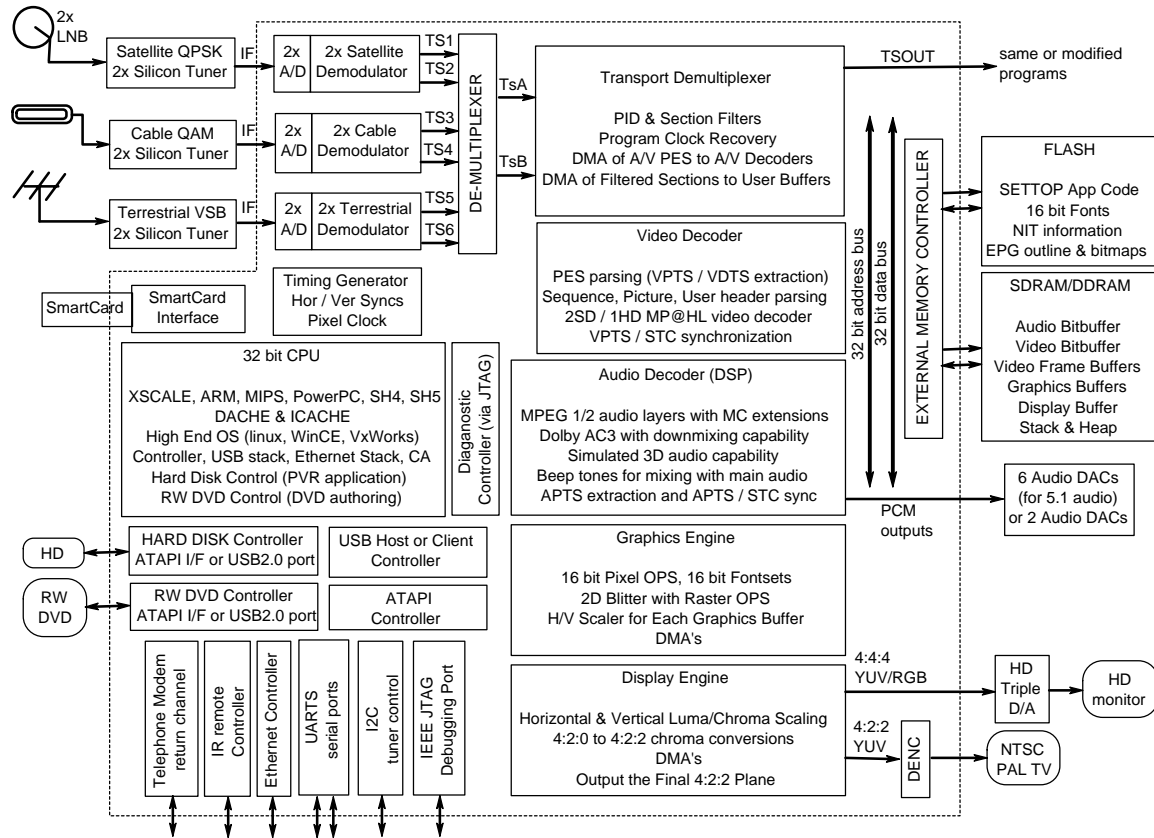


Figure 88. DVB SETTOP Box used in Satellite/Cable/Terrestrial Networks

Figure 88 generalizes the components found in a high end SETTOP box compatible with all the 3 transmission systems, running a high end OS like WinCE/LINUX/VXWORKS, which can be used as a PVR via a ATAPI/USB2.0 hard disk or can be used for authoring DVD's via a ATAPI/USB2.0 DVD-RW.

Let us discuss the functionality of each component in this SETTOP box.

1. **Silicon Tuner** – this is an analog chipset which takes a RF signal (satellite, cable or terrestrial feed), demodulates it to its IF (I/Q) components (just I for VSB) and sends these components to the digital demodulator. A silicon tuner replaces the standard box tuner with a silicon chipset which has no moving parts. A silicon tuner is thus immune to any acoustic interference and has better stability. Two (2x) silicon tuners are shown for satellite, cable and terrestrial to demodulate 2 separate transponders (QPSK) or 2 separate channels (QAM or VSB) to facilitate watch and record PVR functionality.

2. The demodulator is a device discussed previously which takes the IF signal components, converts them into the digital domain, equalizes I/Q components to open the constellation's eyes for optimal slicing (removal of ISI and ghosts), de-rotates the constellation to remove residual phase errors and performs forward error correction (concatenated error decoding) on the recovered signal. The output of the FEC has the following parts,

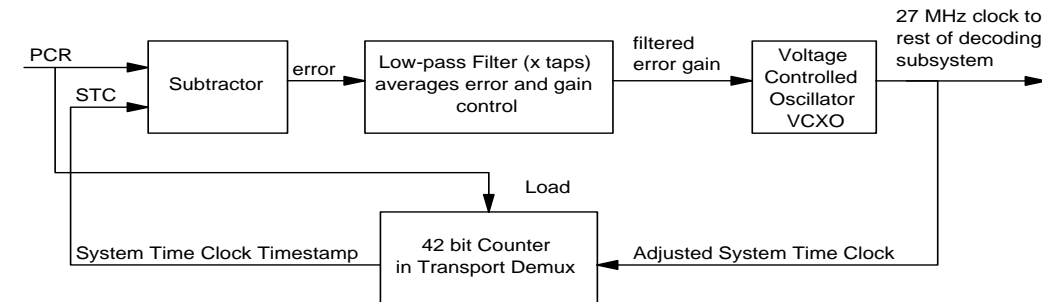
- Transport data byte stream (consisting of transport packets).
- Error signal indicating whether the FEC has been able to correct all the bytes in that transport packet. This is high if the FEC (or the final Reed Solomon stage) is unable to correct all its bytes (RS parity/2) in transport packet. This signal is identical to the transport error indicator bit which is set to 1 for an erroneous transport packet.
- Data valid signal indicating which bytes in the transport stream are valid for sampling using the byte clock and which bytes are the RS parity bytes. This can be high or low within a transport packet depending on how the FEC RS stage sends out the parity bytes. In a MPEG2 transport packet this is high for exactly 188 bytes within 2 transport packet start signals.
- Byte clock for sampling the transport data stream in the transport de-multiplexer.
- Transport packet byte wide start signal indicating start of a transport packet.

Two (2x) demodulators are shown for each transmission for a total of 6 transport streams which are de-multiplexed to provide a total of 2 transport streams to the transport de-multiplexer.

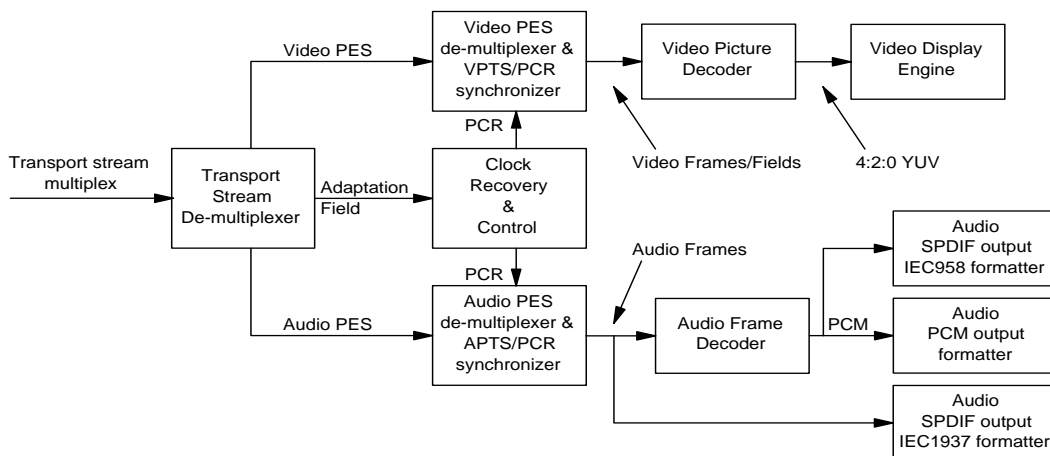
3. The transport de-multiplexer provides the following functionality,

- Inputs the 2 transport streams (combination of the 6 input transport streams) as selected by the user's application, buffers each transport packet into a 188 byte FIFO and applies a PID filter to discard the TP or process it further downstream.
- PID filters the transport packets corresponding to the PAT, PMT, NIT, BAT, SDT, TDT, EMM, ECM sections, filters these sections using section filters, buffers the processed sections and with the help of the CPU builds an online program guide in memory (using one of the graphics planes in SDRAM/DDRAM) which lists all the programs in the transport multiplex for a given service (satellite only, cable only, terrestrial only OR satellite & cable, satellite & terrestrial, cable & terrestrial), and their times of presentation.
- When the user selects a program from the EPG, the CPU first checks to see if this program is in the clear or scrambled using the scrambling control bits in the filtered transport header.
 1. If this is a pay (scrambled) program, the Conditional Access Module running on the CPU checks (using filtered EMM sections) if the user's smartcard is entitled for that program. If it is, the control words are extracted from the filtered ECM sections, decrypted using the smartcard into odd/even keys, and used to arm the descrambler. The transport packets containing the scrambled A/V PES packets corresponding to that program are PID filtered, their PES extracted and sent to the descrambler which descrambles these PES packets using the keys provided and then forwards the PES streams to the A/V decoders and display engine for decoding that program and presenting it to the user's display/sound device.
 2. If this is an in the clear program, the CPU just forwards the PID filtered A/V PES packets corresponding to that program, to the A/V decoder and display engine for decoding that program and presenting it to the user's display/sound device.

- PID filters the transport packets as per a program's elementary PIDs provided by the electronic program guide (EPG).
- Extracts the Program Clock Reference (PCR) timestamp and its Extension from the adaptation field in the transport packets belonging to a certain program, compares this 42 bit value to a 42 bit System Time Clock Counter (STC counter is initiated by the 1st PCR and clocked using the 27MHz system clock), forms a +/- error, averages the error using a FIR filter, loads it into a Pulse Width Modulator which is used to change the voltage of a VCXO which corrects the phase of the 27MHz clock to provide exact sampling points for the transport data bytes further down the processing pipeline in the SETTOP box. Thus process is called 27MHz clock recovery.

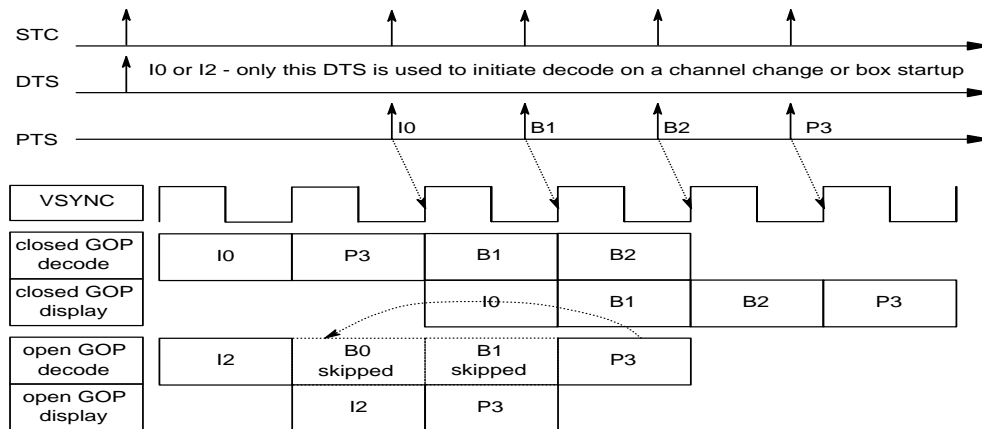


- DMA's the filtered sections to user buffers for further processing by the CPU running the user's application (typically to build the EPG) and the Conditional Access Module. Some of the filtered sections could be private sections which could carry user specific data.
- DMA's the filtered A/V PES packets to A/V decoders.



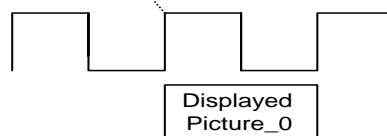
4. The video decoder can be a complete sequence or a picture decoder which under the CPU control can be setup to decode complete video sequences or just interlaced/progressive pictures down to the 4:2:0 macro-blocks. It accomplishes the following functions,

- Extracts the PTS/DTS timestamps from the PES packets. The 32 bit DTS timestamp tells the video decoder when to start decoding the first picture which belongs in the PES packet containing the DTS after a channel change or box startup. This is determined by comparing the 32 bits of the extracted DTS with the 32 LS bits of the STC and starting the 1st picture decode when they are equal. The 32 bit PTS timestamp is used by the display engine to present a decoded picture when its PTS equals the STC.



- The video decoder, if used as a sequence decoder, decodes a complete video sequence without CPU intervention. All the start codes in the video bitstream are processed within the video decoder. This includes parsing of the sequence header, sequence extension, GOP header, picture header, picture coding extension, picture display extension and user headers. All the relevant information from these parsed headers is used to setup the video decoder state machine (or internal custom micro) to decode and reconstruct every picture in the sequence without CPU intervention. The slice, macro-block and luminance/chrominance block headers are also processed by the video decoder to build complete reconstructed frames/fields within the SDRAM/DDRAM. Every picture's decoding and display time frame is 2 VSYNCS.
- The video decoder, if used as a picture decoder, to enable trick modes during PVR or DVD Read functionality, under CPU intervention, decodes 1 picture at a time. All the header parsing until a slice is done by the CPU. The CPU uses the information in the parsed headers to setup the video decoder's state machine to decode all the slices in a picture for reconstructing a complete frame or field. This mode is very useful during trick modes like Fast Backward and Fast Forward when the pictures in a program (transport or PES packets) stored on the Hard Disk or DVD are to be extracted, decoded and displayed at speeds other than the normal video decoding/display rates.
- When decoding field predicted sequences, each picture in the sequence will have 2 sets of picture headers, 2 sets of picture coding extensions and 2 sets of picture display extensions. The video decoder parses these 2 sets to decode and reconstruct every field of a picture. Every field's decoding and display time frame is 1 VSYNC period.
- The video decoder can decode 1 HD (MP@HL with 1920x1088 P max resolution) or 2 SD (MP@ML with 720x576 P max resolution) pictures within 2 VSYNCS (@60Hz frame rate) provided by the display engine. This is to enable a user to watch 2, 4/3 SD programs on a 16/9 display.
- The video decoder is also responsible for parsing the user header after the user start code to extract the closed captioning characters for insertion into the Digital Encoder lines or make them available to the graphics engine for display via an OSD font-set.
- The extracted PTS of a frame is compared with the STC, during the VSYNC when that frame is displayed to obtain +/- differences. Comparing this difference with a threshold, that frame is skipped or repeated. When the same technique is used for synchronizing audio frames, a perfect audio/video sync (lip-sync) mechanism is achieved.

$(\text{sampled STC}_0) - (\text{PTS}_0) = \text{difference}$

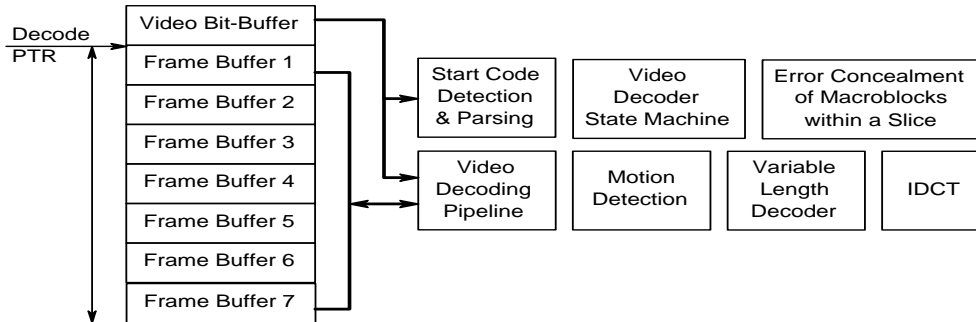


```

if (abs(difference) > 1 Field)
{
    if (difference is +ve)
        skip the picture
    else
        repeat the picture
}

```

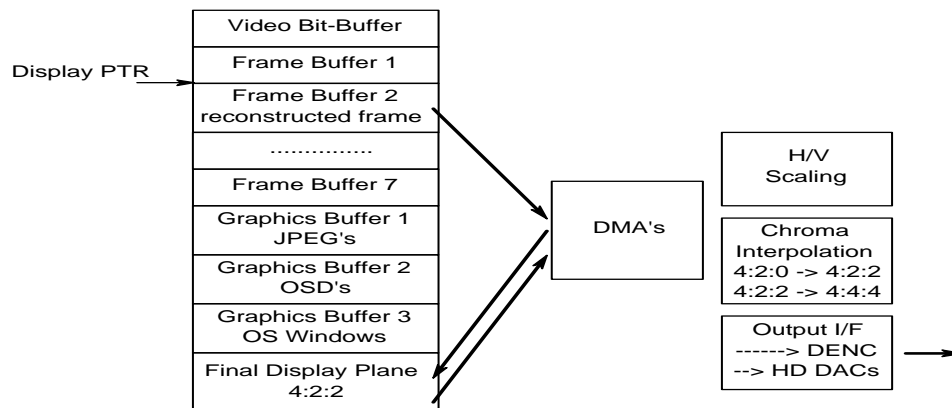
- The video decoder's state machine runs independently of the video display's state machine. Basically the video decoder decodes pictures and reconstructs them into the frame buffers. The display pipeline displays the reconstructed pictures and frees up the frame buffers. These are again filled by the video decoder's state machine.
- The video decoder provides error concealment when macro-blocks within a received field/frame are corrupted by either replacing those macro-blocks from a slice in the previous picture or by interpolating a new set of macro-blocks for that corrupted slice. The interpolating scheme yields much better results (to the viewer's eye) depending on how many slices are used in the interpolation.
- The video decoder's pipeline is shown below,



The video bit-buffer varies in size depending on the max resolution of the picture being processed as discussed previously. The frame buffer size is in 4:2:0 pixels and is given by Horizontal Resolution x Vertical Resolution x 12 bits.

5. The display engine is responsible for the following steps functions,

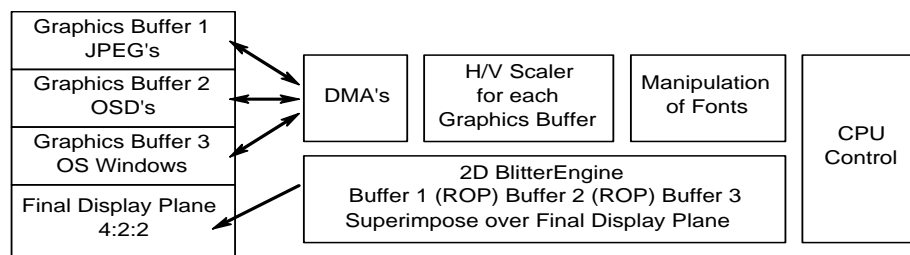
- Taking the reconstructed 4:2:0 macro-block video frames and up-sampling the chrominance vertically to 4:2:2 macro-blocks.
- Up-sampling the video frames horizontally and vertically to convert the decoded resolution to the needed display resolution. This is a 4:2:2 display frame plane over which all other planes (graphics, windowing environment of the high end OS, jpeg images) are superimposed.
- The display engine using DMA's moves the reconstructed frames in and out of SDRAM/DDRAM memory for up-sampling. The up-sampling in the horizontal and vertical directions is done by 4-tap filters which generate new lines (from 4 original lines) and pixels (from 4 original pixels). Also the vertical interpolation is restricted to fields (new line generated from 4 original lines in the same field) when the picture is field predicted.
- The display engine keeps track of the display pointer.



- The final 4:2:2 display plane (scaled video, superimposed graphics, OS windowing environment) is sent via a DMA to the Digital Encoder (DENC) → NTSC/PAL TV, or in the case of HD streams (HD formats) the chrominance is horizontally up-sampled to 4:4:4 pixels and sent to a HD RGB/YUV triple DAC → HD monitor.

6. The graphics engine under CPU control does the following,

- The EPG information collected by the CPU can be rendered using 16 bit pixels and put into the graphics buffer 2.
- The windowing environment of the high end OS can also be rendered using 16 bit pixels under CPU control and put into graphics buffer 3.
- The CPU being over 200 MIPS can decode JPEG images on the fly. These can be rendered and put in graphics buffer 4.
- The pixel information stored in all the 3 graphics buffers is in 4:2:2 pixel format.

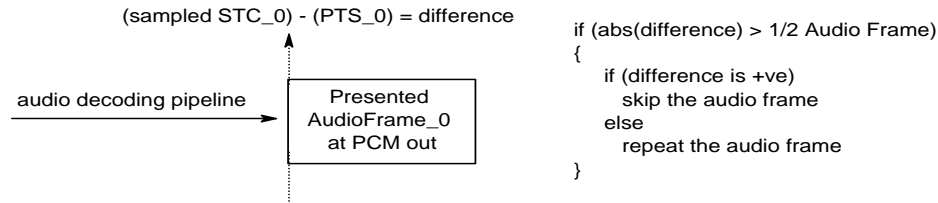


- The 3 graphics buffers can be sent to the blitter engine which can SCALE each individual buffer in the horizontal and vertical directions, blit them together using Raster Ops (XOR, AND, etc.) and superimpose them over the final 4:2:2 display plane. This final display plane is sent to the display device/s via the display engine.

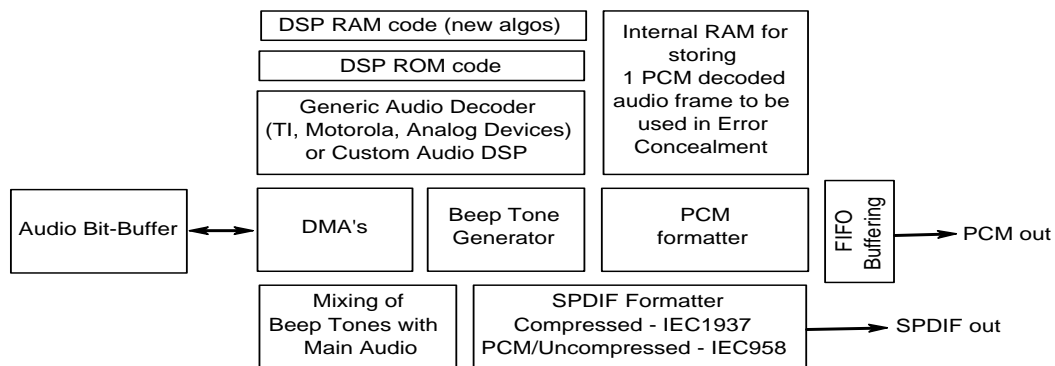
7. The audio decoder could be a generic DSP core like the one from Texas Instruments, Analog Devices or Motorola which can be programmed with standard MPEG audio, DOLBY AC3 & 3D audio decoding algorithms and used as an embedded DSP. The audio decoder could also be a custom DSP whose instruction set and hardware has been defined for specific audio decoding and processing algorithms. The audio decoder shown in the SETTOP box has the following functions,

- Inputs PID filtered audio PES packets which could contain MPEG audio, DOLBY AC3 audio or PCM CD audio, extracts the PTS timestamps and forwards the elementary streams to the audio decoding pipeline.
- Decodes all the MPEG1 layers (1, 2 & 3) along with its multi-channel extensions to provide 5 audio channels & 1 LFE channel. It should also be able to handle MPEG2 multi-channel (MC) audio.
- Decodes DOLBY AC3 5.1 channel audio.
- Down-mixes 5.1 MPEG MC audio or 5.1 DOLBY AC3 audio to 2 Left/Right down-mixed channels to support a SETTOP box with only 2 audio DACs.
- Decoding 2 channel MPEG audio, processing the audio channels using custom 3D audio algorithms to provide 3D simulated audio using just 2 audio DACs.
- The audio decoder also has the capability of generating beep tones (single frequency sine waves) which can be mixed with main audio using weighted mixing.
- The audio decoder performs audio synchronization by comparing the APTS extracted from the PES packet with the sampled STC, when the audio frame in that PES packet is

presented (decoded and sent out on PCM out). The difference between the APTS and STC is compared with a threshold and the audio frame is repeated or skipped.



- The audio decoding pipeline is shown below. The code in the ROM is used to provide very specific decoding/processing audio algorithms. The RAM is used to provide custom or new algorithms which can be added in by the CPU to support new audio decoding formats. The audio decoder has storage for 1 pcm frame for error concealment purposes.



- The timing generator is used to provide all the timing information to the SETTOP environment. It typically provides the horizontal sync, vertical sync and pixel clock for each interlaced/progressive resolution (HD or SD) the SETTOP box supports.
- The digital encoder (DENC) inputs a digital 4:2:2 SD YUV stream, and generates either a analog Y/C SVHS signal or a analog composite video base-band signal (CVBS) which can be accepted by any NTSC/PAL TV set.
- The 32 bit CPU used in a SETTOP box is usually one of the processors like XSCALE, PowerPC, ARM, MIPS, SH3 or SH4 providing the following functionality,
 - The CPU runs the windowing environment of a high end OS like WinCE, VXWORKS or LINUX. It provides the graphical rendering of this environment using a standard font-set supported by the graphic engine.
 - The CPU is also responsible for controlling or setting up all the features in the SETTOP box like transport de-multiplexing, audio decode, video decode, graphics blitter engine, selection of fonts, display engine, etc.
 - The CPU also runs the USB2.0 or Ethernet (wired or wireless) software stacks, when any one of these devices is used in the SETTOP.
 - The CPU is responsible for controlling all the peripheral devices in the SETTOP box like IR controller for user input via an IR remote, UARTs for serial communications with a PC host, I2C for communication with the I2C based front-end tuners.

- The CPU also runs a soft telephone modem stack like V.22 or V.90 which provides the return channel to the head-end in a SETTOP environment when a user selects a pay-per-view program (movie or a boxing match).
- The CPU runs the software part of the Conditional Access Module which interacts with the transport de-multiplexer to extract the control words from the filtered ECM sections, send them to the external smartcard for decryption and uses these decrypted keys to arm the descrambler. The CPU also extracts the entitlement messages from the filtered EMM sections to check provisioning of programs with the user's smartcard.
- In most SETTOP box chipsets a Diagnostic Controller is used along with the CPU, which provides a debugging environment via a JTAG interface to debug a user's application.

11. The External Memory Controller is used to arbitrate and prioritize accesses by the various subsystems within a SETTOP box chipset to/from the memories (Flash & SDRAM/DDRAM) on the external address and data buses. Typically the external memory controller prioritizes the following accesses,

- CPU accesses to FLASH for downloading code and data.
- CPU accesses to SDRAM/DDRAM for running the OS windowing environment, software stacks, control, software modem, hard disk and PVR applications.
- Transport de-multiplexer accesses to SDRAM/DDRAM for storing processed sections (heap).
- Video decoder accesses to SDRAM/DDRAM (video bit-buffer or frame buffers).
- Audio decoder accesses to SDRAM/DDRAM (audio bit-buffer).
- Display engine accesses to SDRAM/DDRAM (reconstructed pictures in the frame buffers or display plane buffer).
- Graphics engine accesses to SDRAM/DDRAM (graphics buffers and display plane buffer).

12. An Instruction Cache (~16kbytes) and Data Cache (~16kbytes) is provided within the CPU to speed up the code and data accesses to/from external memories. The data cache becomes especially important when implementing data intensive applications like USB & Ethernet software stacks or building the Electronic Program Guide. The instruction cache becomes important when implementing math intensive (DSP) applications like software modems.

13. The FLASH is usually 32 bits wide and can be of any size, ranging from 4Mbytes to 64Mbytes. This is usually used to store the application code snapshots (OS windowing environment, application control code, software stacks, etc). The code snapshots in flash could be loaded at the factory where the SETTOP box is assembled or loaded in real time via the channel medium (satellite, cable or terrestrial) sometimes in the future (updates to the application code, etc.). The flash is also used to store different font-sets to be used in the application. It also stores the Network Information Table entries such as transponder frequencies on a satellite feed, channel frequencies on cable or terrestrial feeds to aid in fast tuning to different transponders or channels.

14. The SDRAM/DDRAM is usually 32 bits wide and can be of any size ranging from 16Mbytes to 64Mbytes. DDRAM (double data rate DRAM – data is clocked on both edges of the DRAM clock to support very fast transfer rates)

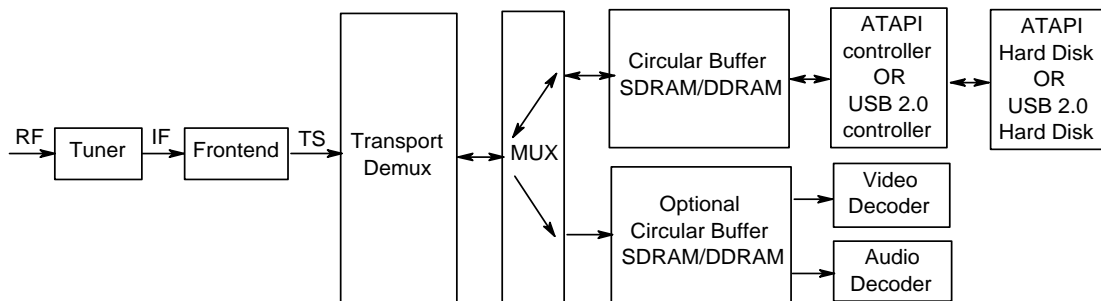
is selected for supporting decoding of high definition pictures. The SDRAM/DDRAM memory has the following sections,

- Audio bit-buffer – this is used to provide buffering for the audio PES packets sent via DMA from the transport de-multiplexer. This is usually couple of frames deep.
- Video bit-buffer – this is used to provide buffering for the video PES packets sent via DMA from the transport de-multiplexer. Depending on the profile and level of the video being decoded it can be up to 10 Mbits.
- Video frame buffers are typically 7 (in 4:2:0 chrominance format) in order to support separate decoding and display pipelines, and also to support various trick modes for a PVR or DVD application.
- There are typically 3 graphics buffers (4:2:2 chrominance format) as shown before typically to support the high end windowing environment of the OS, on screen display for the electronic program guide and for JPEG still pictures.
- There is 1 final display buffer (4:2:2 format) which contains the horizontally and vertically scaled reconstructed pictures. All graphics planes are superimposed (after going through a 2D blitter engine) over this display plane using appropriate weighting as defined by the display engine. This final display plane is dma'ed out to the display device.
- Finally the SDRAM/DDRAM contains the stack and heap of the user's application.

PVR application on a SETTOP box using a ATAPI or USB 2.0 Hard Disk

Typically a Hard Disk (ATAPI or USB 2.0) is added to a SETTOP to enable PVR (personal video recorder) functionality. Two things are to be achieved,

- Pause a live program being decoded for some period of time, record that same program on a hard disk while in pause mode and play it back from the hard disk. This is typically called time-shifting of live programs using the hard disk.



The original decode path of the single A/V program is as follows:
tuner → frontend → transport demux → optional buffering → A/V decoders.

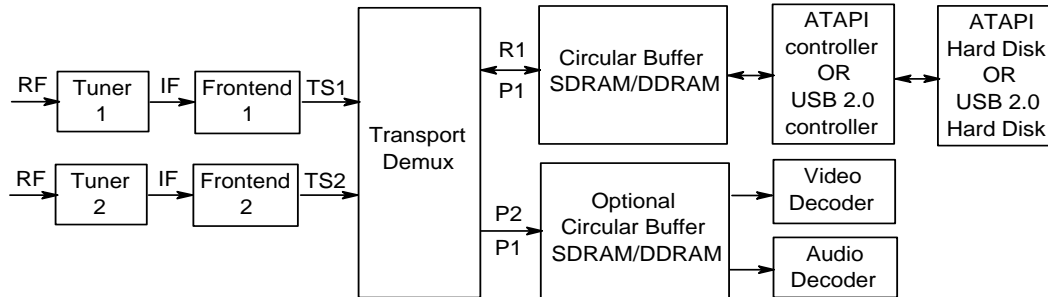
When the user hits pause, the A/V program path is as follows:
Tuner → frontend → transport demux → buffer → atapi or usb controller → Hard Disk

When the user hits resume, the A/V program path becomes as follows:
Hard Disk → atapi or usb controller → circular buffer → transport demux → optional buffer → A/V decoders

One thing to note is that the program recorded could be A/V/PCR transport packets of that program or just the A/V PES packets of that program. When a transport stream is recorded the playback is identical to the original path via the tuner → frontend. When PES packets are

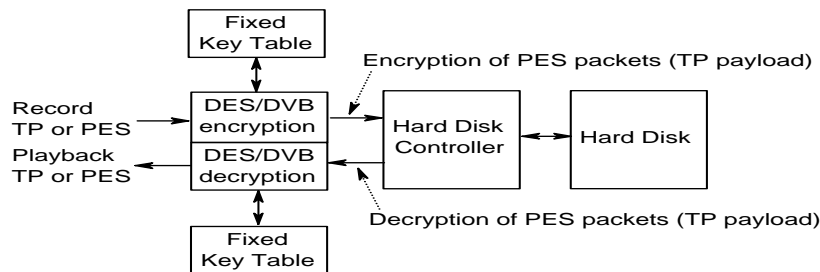
recorded the transport de-multiplexer dma's the PES packets stored on the hard disk via the optional buffer to the A/V decoders. The A/V decoders now initiate the lip-sync mechanism by writing the 1st audio or video PTS (when the 1st audio or video frame is presented) into the STC because of the un-availability of the PCR for that program.

- Recording 1st program on a hard disk while watching another (2nd) live program. The recorded program can be watched at a later time or date. This is similar to a digital VCR using the hard disk.



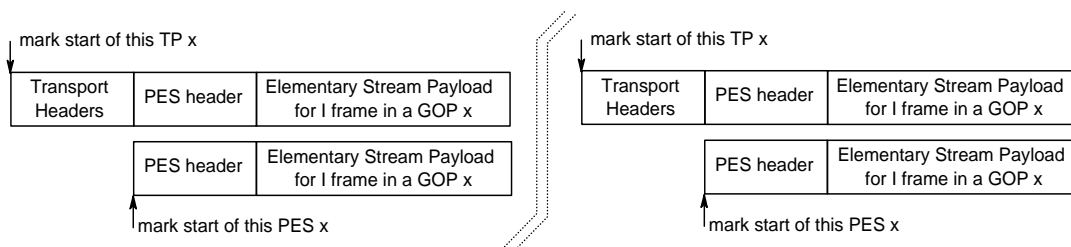
Since there are 2 tuners and 2 front-ends in the SETTOP shown, 1 program can be recorded on the hard disk (R1) while the other program (P2) is being played back. Then the program stored (recorded) on the hard disk (R1) can be played back at a later time.

To add security to this mechanism a DES or a DVB encryption/decryption block can be added in the path of the record/payback with fixed keys such that a program (pay-per-view movie) stored on the hard disk cannot be easily copied.

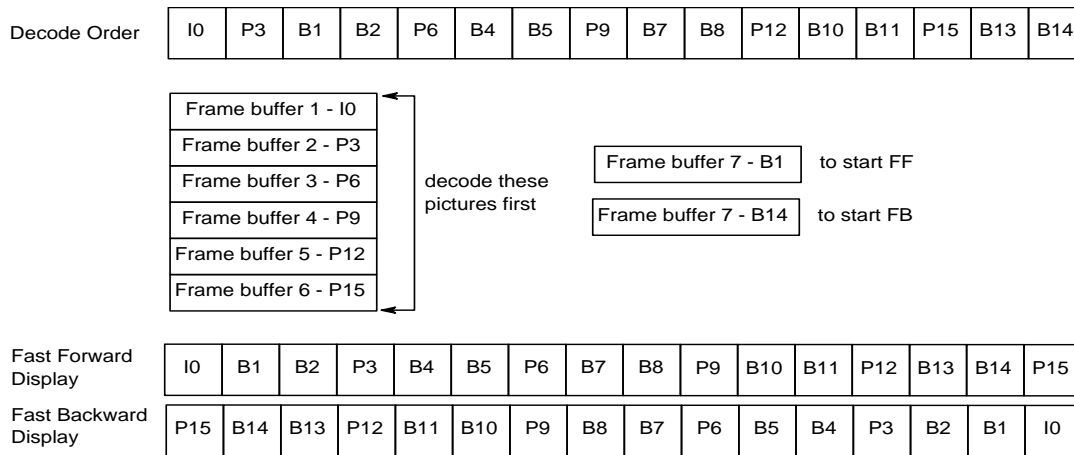


A very simple A/V file system can be used to index programs (in transport or PES format) stored on the hard disk to facilitate easy access to these programs when they are to be selectively played back.

To enable trick modes like Fast Forward and Fast Backward, a very simple indexing scheme (for transport or PES packets) can be used,



This along with the 7 video frame buffers can be used to FF or FB a closed GOP as shown next,

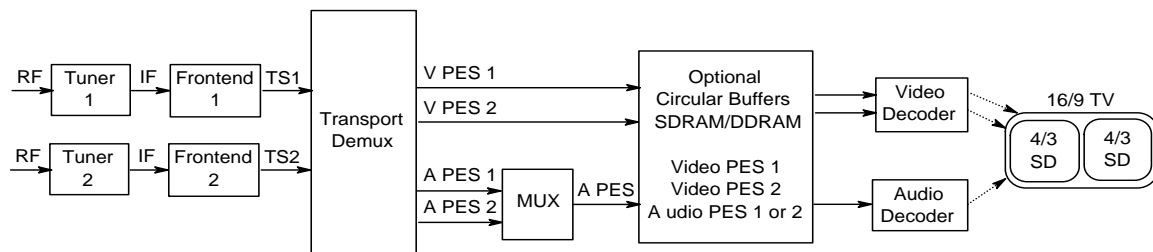


Basically to enable smooth FF and smooth FB, each transport or PES packet containing the start of the GOP (I frame within that GOP) is indexed. When the user selects a program from the hard disk, the transport or PES packet belonging to the start of that GOP is used as the starting point for the read back mechanism. The pictures in the GOP are decoded as fast as possible (by the video decoder) as shown and depending on the mode (FF or FB) the rest of the pictures are re-constructed. The B frame buffers can be freed as soon as they are displayed. The P frame buffers can be freed as soon as the P pictures in them and their corresponding B pictures are displayed. The display pointer is manipulated to pick up the pictures in the correct temporal order and also as per the FF or FB rate specified by the user (by skipping some or all the B pictures without displaying them).

This method uses additional frame buffers but a complete high end SETTOP application (HD PVR and SD DVD authoring) can be made to fit within 64 Mbytes of SDRAM/DDRDRAM and 64 Mbytes of FLASH.

Dual SD decode in the SETTOP

The SETTOP shown can decode 2 SD pictures and 1 HD picture. Dual SD decode is typically used when 2 4/3 SD programs are to be decoded and displayed on a 16/9 HD monitor, thus enabling 2 persons to watch 2 different programs side by side on the same 16/9 screen. The flow diagram for achieving this is shown next,



The 2 transport streams from 2 separate transponders (Satellite) or from 2 separate channels (Cable or Terrestrial) are de-multiplexed into 2 separate audio and video PES packets belonging to 2 separate SD programs. These PES streams can be optionally buffered before being sent to the A/V decoders.

The video decoder is capable of decoding both the video PES streams simultaneously (basically using 3 frame buffers for each SD stream it can decode 2 complete SD pictures within 2 VSYNCs). The display engine takes the 2 reconstructed SD 4:2:0 pictures, up-samples the chrominance to 4:2:2, up-samples each SD picture to its full resolution (720x480 or 720x576) and concatenates them in the final display buffer. From there the entire display buffer is outputted via the display engine to the HD 16/9 display device.

The audio decoder is capable of decoding only 1 audio PES stream. Depending on the choice made by the user, one of the audio PES streams (corresponding to video PES 1 or video PES 2) is sent to the audio decoder.

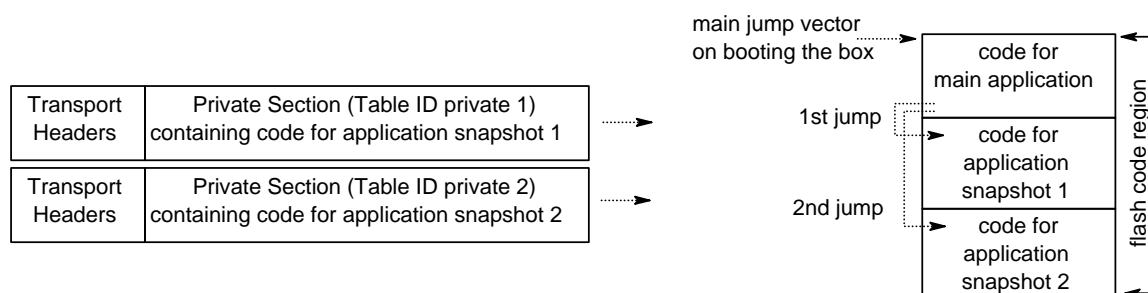
Transport Stream Output functionality in a SETTOP environment

The transport de-multiplexer in the SETTOP box chipset is capable of sending out transport packets belonging to certain PIDs selected by the user. Typically an entire descrambled program is sent out via the transport stream output to another device capable of accepting transport streams.

Sometimes a user can make his own program (by selecting a set of A/V PIDs), modify the PAT/PMT tables as per this new program and send out the transport packets belonging to this new program's PIDs along with the modified PAT/PMT transport packets. Thus a device receiving this transport stream has knowledge of the transport stream contents.

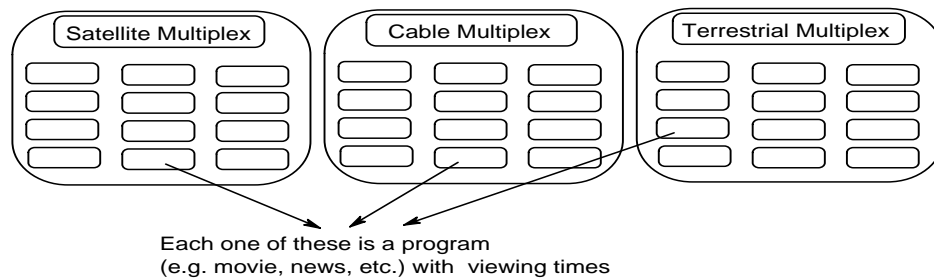
Downloading application snap-shots via Private Sections

The flash in the SETTOP box is typically partitioned into at least 3 regions, where the 1st region is for the main SETTOP box application and the next 2 regions are for adding field updates to the main application. These field updates could be to enhance functionality of the existing application, fix some known bugs, change a feature set or provide new provisioning (entitlements) to the SETTOP box. This is usually achieved by breaking and encapsulating application code snapshots into private sections. These private sections are then parsed by the transport de-multiplexer and the data put into user buffers. The CPU then takes the data from these user buffers and writes them into the appropriate flash region.



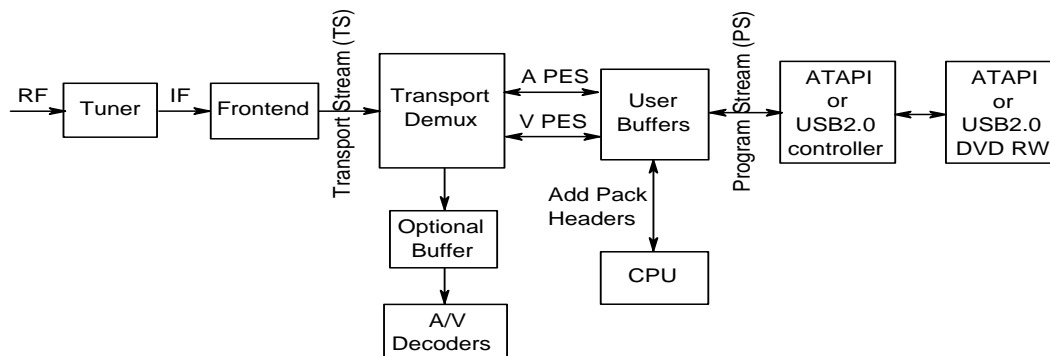
Building a Electronic Program Guide

An Electronic Program Guide is a database listing graphically all the programs (with their specific descriptors, e.g. movie content, news content, pay-per-view content) in a transport network multiplex (satellite, cable, terrestrial) with their presentation (or viewing) times. Typically all the information in the BAT, PAT, PMT, EIT, NIT, SDT and TDT sections (or tables) is used to construct an EPG. The transport de-multiplexer parses (using section sections) these sections and puts their contents into user buffers. The CPU further parses these user buffers, extracting the appropriate content to build the EPG database with the help of the graphics engine.



DVD Authoring Application on a SETTOP box using a DVD RW

Similar to the Personal Video Recorder application the SETTOP box is also capable of encapsulating A/V PES packets belonging to a certain program into packs by adding in the appropriate pack headers, thus creating a program stream with audio and video packs. This program stream can then be stored on the ATAPI or USB 2.0 DVD RW drive. If navigation packs are to be added to this program stream along with the A/V packs (to create different A/V scenes, etc.), then suitable authoring tools need to be developed which can isolate (find) and tag various scenes in the A/V program stored on the DVD RW.

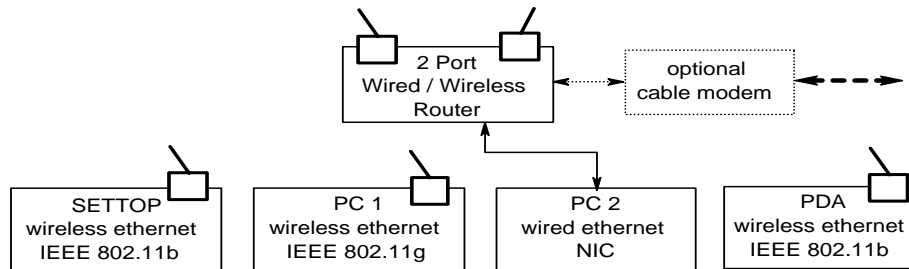


Ethernet (wired or wireless) Connectivity

If the SETTOP box is to be placed on the home 10/100 Base-T Ethernet network, then it has to have the following,

- Wired network interface card (Ethernet PHY + Ethernet MAC controller) OR,
- Wireless network interface card (supporting the wireless IEEE 802.11b/g standards).
- Software Ethernet stack to extract/add Ethernet headers from/to the payload transferred over the Ethernet network.

Thus a SETTOP box can be an integral part of the home network and can share MPEG video files, MPEG/AC-3 audio files and JPEG pictures via the hard disk or via the DVD RW.



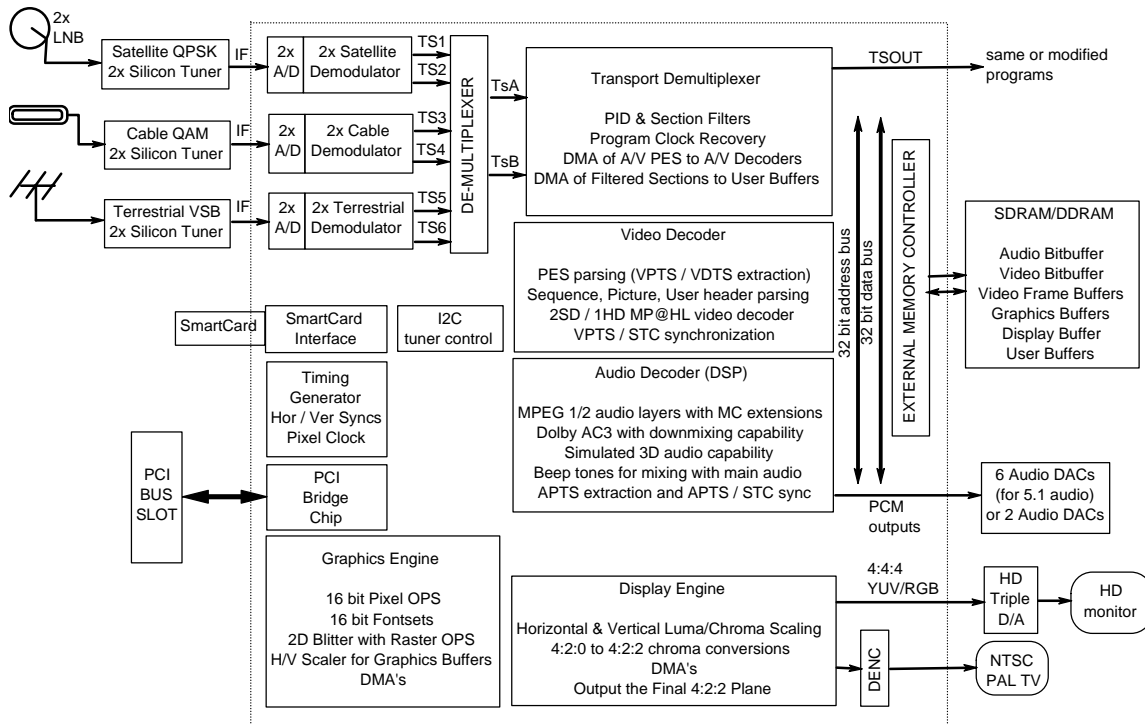
Using the PC as a SETTOP Box

A PC can be used as a SETTOP box by adding in a “PCI card with a SETTOP box chipset” in one of the PC’s PCI slots as shown below,

The following components can be omitted from a standalone SETTOP box chipset.

- The CPU is the Intel Pentium 3 (or 4) processor.
- The PCI card does not contain any FLASH since the application is booted of the PC hard disk and runs in the PC RAM. Basically the SETTOP box application is a device driver running under the windows OS and accessing the PCI card.
- Since the PC has a hard disk the PCI card does not need a hard disk controller.
- Since the PC has a DVD RW drive the PCI card does not need a DVD RW controller.
- Since the PC has an Ethernet NIC and a USB 2.0 host, both these controllers need not be present on the PCI card.
- Most peripherals like telephone modem, IR controller, UART controller, IEEE JTAG port can be omitted.
- The OS used can be windows 2000/XP and thus no third party OS is needed.
- The debugging environment can be provided by a tool like WIN-DBG, and thus there is no need for a built-in diagnostic controller.
- Software stacks & device drivers for USB and Ethernet are already provided by the Windows environment but have to be integrated (using suitable API's) with the application's device driver.

The PCI card (registers in the SETTOP box chipset plus SDRAM/DDRAM memory chipset) is memory mapped in the 4GB address space of the 32 bit Intel Processor. A standard development package like MS Visual C along with the windows 2000/XP software development kit (SDK) can be used to develop a device driver (the SETTOP application) running under windows 2000/XP.



One additional component needed for the application's device driver to access the SETTOP box chipset registers and memory on the PCI card is a PCI bridge interface chip. There are several manufacturers of these PCI bridge chipsets.

Some "software only" implementations of a SETTOP box on a PC with the PCI card containing only the tuner and a front-end chipset are possible. Here the rest of the system is implemented in software on the PC. However, real-time constraints in a transport multiplex, limit such a system from being able to de-multiplex and decode more than 1 A/V program at a time.

Notes:

1. Some of the important characteristics of a SETTOP box chipset or its application, is how well it handles error conditions in the transport/PES/ES bit-stream either (a) due to incorrect encoding (wide gamut of encoders available, including pure software based encoders) or (b) due to errors introduced by the channel (satellite, cable or terrestrial).
2. Errors due to incorrect encoding may be in the transport header, PES header, section headers, ES headers or even headers at the slice, macro-block or block level. In these cases the SETTOP box chipset and its application must be capable of performing some of error recovery and in the worst cases reset its hardware/software state machines and start processing from state 0 (default startup state). The encoding errors should never cause a SETTOP chipset or its application to fail requiring a user reboot.
3. Errors caused due to the noisy channels (rain fade causing tuner unlocks, too much jitter causing wide variations in the arrival of packets, etc.) should be handled cleanly by the application.

TYPICAL DVD PLAYBACK APPLICATION

In this chapter we will discuss all the aspects of a typical DVD playback application. A DVD chipset is very similar to a SETTOP chipset with the following differences as shown in figure 89,

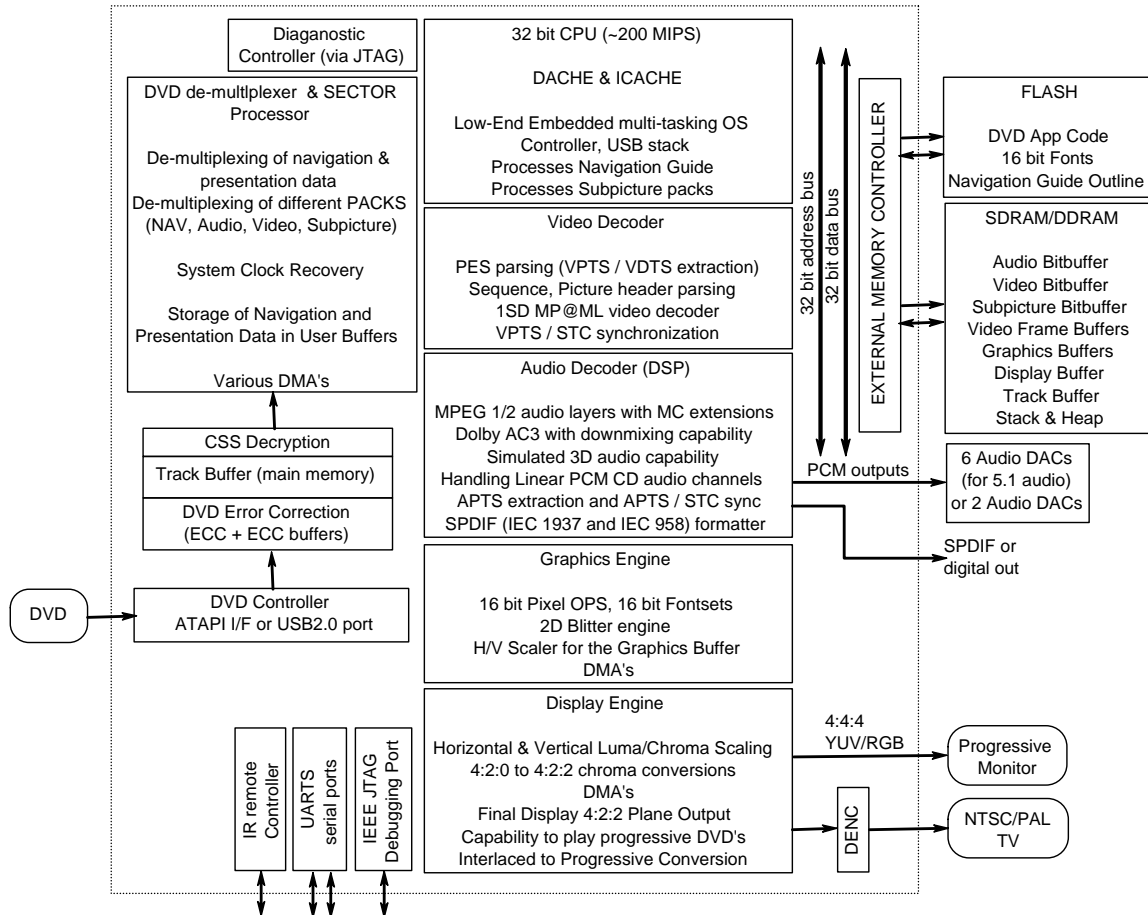
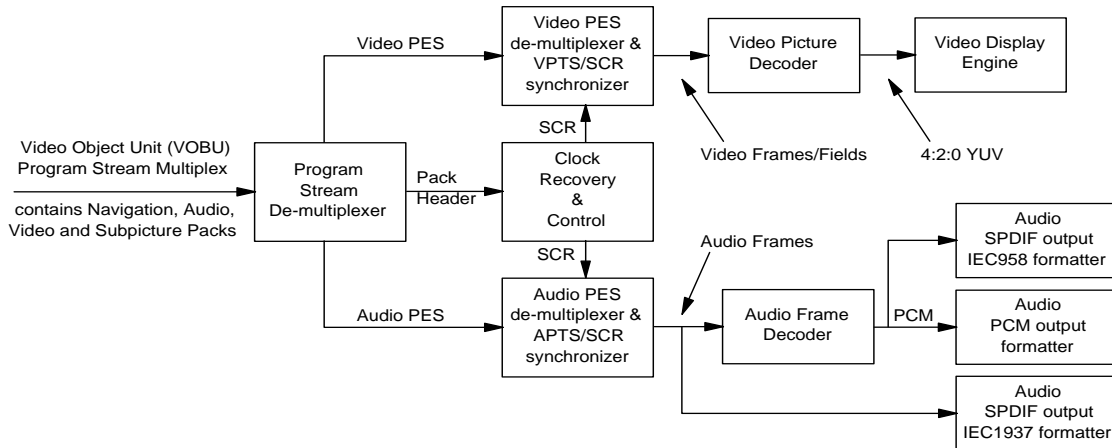


Figure 89. DVD Playback Chipset

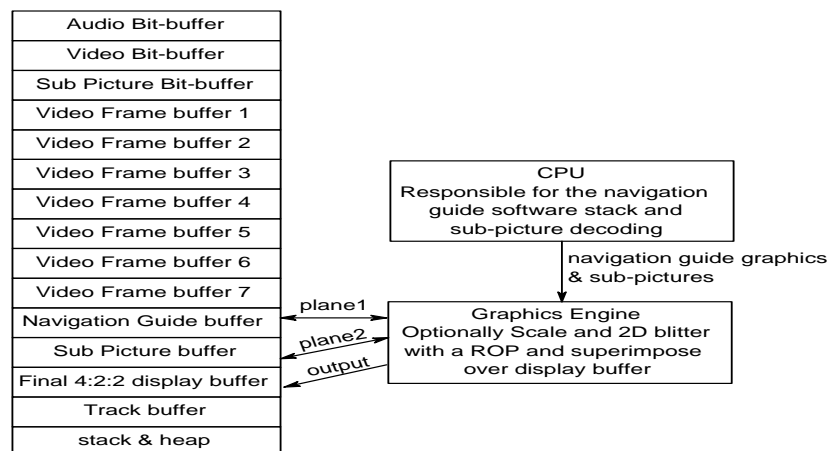
The differences between the different blocks in a DVD chipset (from a SETTOP box chipset) are described as follows,

1. The video decoder no longer has to support HD formats, since all the video streams on a DVD are MP@ML with maximum interlaced/progressive resolutions of 720x480 or 720x576.
2. The audio decoder has to support all the MPEG1 & MPEG2 layers 1, 2 and 3 along with multi-channel (MC) audio extensions. It should also be capable of supporting DOLBY AC3 5.1 audio. Some DVD discs may contain audio recorded in Digital Theatre Sound (DTS® format, in which case the audio decoder must be capable of handling the DTS (Digital Theatre Sound) format.

Also the audio decoder must be capable of handling standard linear PCM channels (CD audio channels found on a typical CD) encapsulated in MPEG2 PES packets. The audio decoder is also capable of formatting the compressed elementary streams (ES frames) into IEC1937 format or formatting the decoded PCM samples into IEC958 format. The IEC formats modulate the data samples with a clock and send out a serial bit-stream which is called SPDIF formatting or digital out. This can then be sent to a SPDIF receiver (in an external audio decoder) which first recovers the clock, and then the compressed or un-compressed audio data samples.

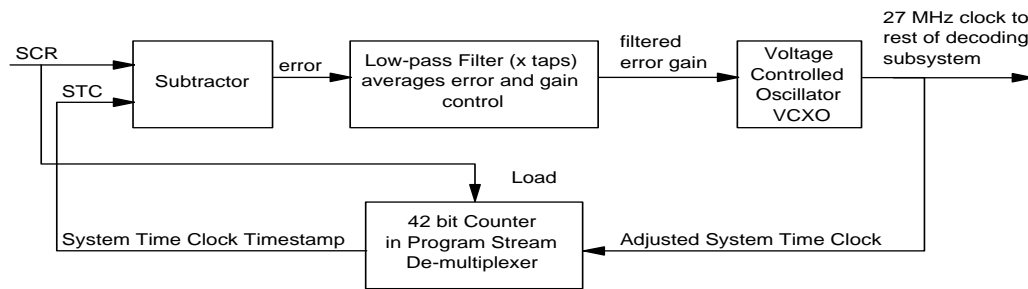


3. The graphics engine is quite the same, but there are only 2 graphics buffers (navigation guide buffer and the sub-picture buffer). Both the navigation guide and the decoded sub-pictures are rendered by the graphics engine under CPU control. Optionally H/V scaling can be applied to both buffers. Also there is a 2D blitter engine which can blitter both these graphics planes with a suitable ROP and superimpose (mix) the blitted 4:2:2 plane over the final 4:2:2 display plane using a suitable weighting factor.



4. The DVD de-multiplexer inputs and parses 2 types of data from the DVD disc (Navigation data and Presentation data). Navigation data defines how the presentation data is to be played back and consists of VMGI, VTSI, PGCI,

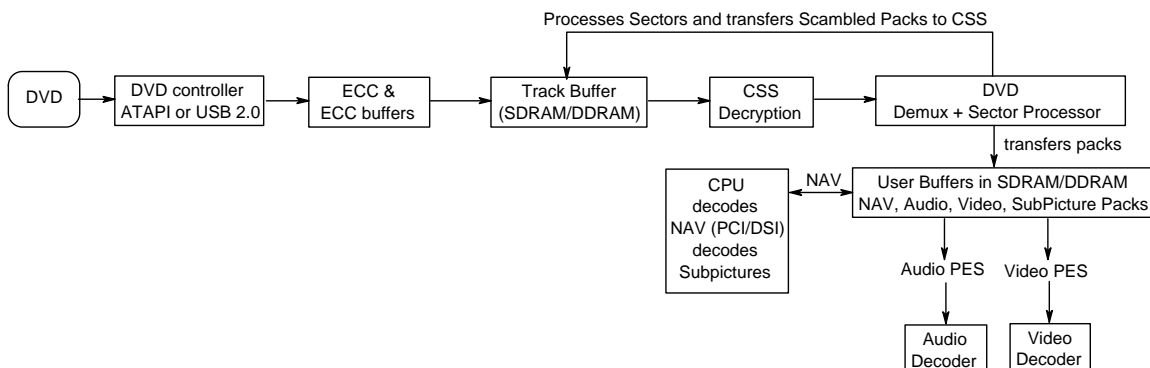
DSI and PCI information. Presentation data consists of video, audio and sub-picture packs. Both these data types are parsed and collected into user buffers which are further processed by the CPU. The CPU uses the DMA's in this block to send the processed A/V PES packets to the A/V decoders. The DVD de-multiplexer is also responsible for extracting the 42 bit SCR (base + extension) in the pack header (typically the video pack) and providing clock recovery for the system clock as shown below. This SCR recovered 27MHz clock is used for clocking all the blocks within the DVD chipset.



The DVD de-multiplexer also performs sector processing. A DVD system has to handle different sector types (sector header + sector data) depending on the type of disk used (DVD, VCD or CD) in the DVD drive. The sectors in the track buffer usually have the form,

Sector ID 1 byte Sector information 3 bytes for sector number	IEC 2 bytes of ID for the error correction code	CPR_MAI 6 bytes of Copyright Management Information	Pack Data (2048 bytes)	EDC 4 bytes of error detection code (similar to CRC)	ECC 2 bytes of Reed Solomon Error Control Code
---	---	---	---------------------------	--	--

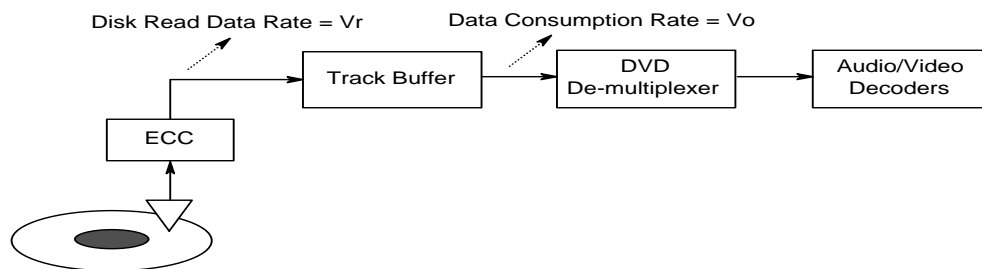
The sector processing involves accessing the sectors stored in the track buffer, stripping of the sector headers, determining sector type (Navigation, audio, video or sub-picture pack), sending the scrambled pack data to the CSS descrambler and storing the descrambled packs into user buffers. The CPU can then access these packs, process them further (Navigation and sub-picture) and DMA the A/V PES packets to the A/V decoders.



5. The 32 bit CPU in a DVD system has the following responsibilities,

- Runs the DVD application control code and is the controller for the DVD player.

- Runs the low-end pre-emptive (16/32 tasks) operating system. This could be any one of the standard embedded off the shelf OS supporting the Intel, MIPS, ARM, PowerPC or SH3/4 family of 32 bit processors.
 - Runs the USB 2.0 software stack for transferring disk sectors between the DVD and the DVD de-multiplexer if the DVD is a USB 2.0 drive.
 - If the DVD is a standard ATAPI DVD, then the CPU is responsible for the setup of the ATAPI interface registers for transferring sectors between the DVD and the DVD de-multiplexer in any of the supported ATAPI modes (PIO/Register, MWDMA or UDMA).
 - Runs the navigation manager for processing and handling the navigation data.
 - It is also responsible for parsing, decoding and displaying (via the graphics engine) the sub-pictures in the sub-picture packs that are extracted by the DVD de-multiplexer.
6. CSS is the content scrambling system (as seen before) used for storing content on a DVD. This is usually a hardware block (under CPU control) between the track buffer and the DVD de-multiplexer.
 7. ECC or error control coding is a method by which error protection bytes are added to DVD sector blocks before being stored on a DVD. This is usually done using a block based coder (Reed Solomon Encoding) and provides protection against DVD read errors (head failures, etc.). In a DVD player a Reed Solomon Decoder is used to decode and correct the read ECC blocks using these additional bytes before transferring the corrected blocks into a track buffer. The memory for the ECC buffers is a part of the ECC block.
 8. Since the DVD read mechanism is semi-synchronous (random access and not continuous like in a SETTOP box), blocks of ECC data read from the DVD have to be buffered before being transferred at a continuous rate via the de-multiplexer to the audio/video decoders (basically the bit-buffer levels feeding the audio/video decoders must be maintained as per the MPEG standard to avoid starving or over-feeding the audio/video decoders). The track buffer (part of SDRAM/DDRAM memory) is responsible for achieving this,



The track buffer size equation for a DVD player is given below,

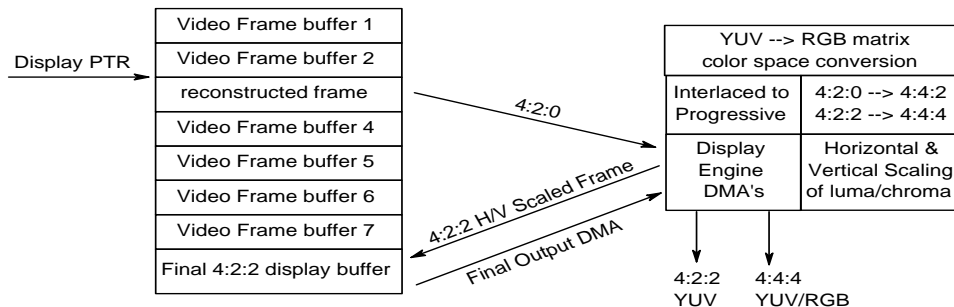
$$B_m \geq \lceil (2 \cdot T_k + t_j + 4 \cdot T_e) \cdot \text{MAX_Vo} \cdot 10^6 \rceil / \lceil 2048 \cdot 8 \rceil$$

B_m (sector)	Track Buffer Size
T_k (sec)	Time of kick-back = Latency Time for 1 Rotation of the Disk
T_e (sec)	Read in time for 1 ECC block = 24 millisecond
t_j (sec)	Track Seek Time
T_j (sec)	Jump Time = Disk Track Seek Time (t_j) + Latency or kick-back Time (T_k)
MAX_Vo	Maximum data consumption rate in Mbps

The below table shows the track buffer size in sectors (2048 bytes) for different data consumption rates (typically consumption bit-rates for a VOB) and different jump distances across sectors.

Maximum Vo (in Mbps)	8	8	7.5	7
Maximum Jump Distance (in sectors)	5000	10000	15000	20000
Maximum $2 \cdot T_k + t_j$ (in msec)	209 + 106	209 + 146	209 + 175	209 + 200
Minimum Bm (sectors)	201	221	220	216

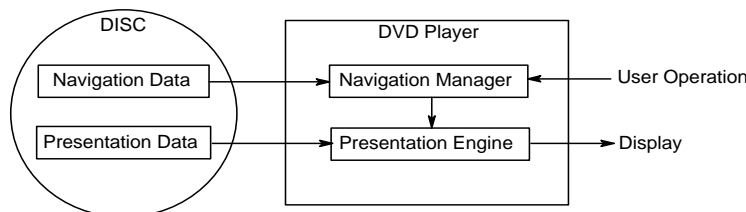
9. The display engine is also the same. It basically takes the reconstructed picture from one of the video frame buffers, scales it horizontally and vertically, converts the 4:2:0 chrominance samples to 4:2:2 and puts this picture in the final 4:2:2 display buffer. Since most DVD's can have video streams encoded as progressive frames the display engine has the capability to output a reconstructed progressive picture as 4:4:4 analog YUV or 4:4:4 analog RGB to a progressive monitor (PC monitor) bypassing the DENC. The display engine also has the capability to de-interlace reconstructed interlaced pictures using a motion estimation/detection and interpolating technique which can also be sent to a progressive monitor.



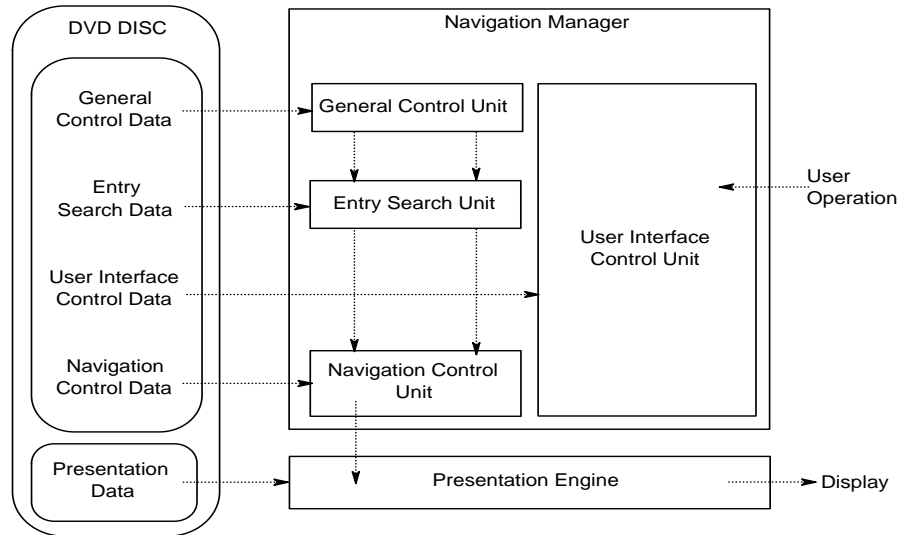
DVD Playback Model

As we have seen before, the DVD data on a disc is organized into different zones. The DVD zone consists of 2 types of data,

- Navigation Data – contains playback control data
- Presentation Data – contains playback data (video, audio, sub-pictures)



The navigation manager handles navigation data such as, VMGI–video manager information, VTSTI-video title set information, PGCI-program chain information, DSI-data search information and PCI-presentation control information. The navigation data determines how a DVD player decodes and plays back the presentation data. In addition to the navigation data, user actions also determine playback different modes.



The Navigation manager is composed of 4 units which provide the needed control for the presentation engine,

1. General Control Unit – this handles 2 kinds of information,

- Stream attributes for Video, Audio and Sub-picture streams – coding mode for each stream, language information to recognize language info in each audio and sub-picture stream and information indicating presence of selectable audio and sub-picture streams for each PGCI.
- Optional parental management information – information on various moral codes on movies and videos for each country, thus allowing for automatic selection of video, audio and sub-picture streams based on the parental level.

2. Entry Search Unit – this handles 2 kinds of information,

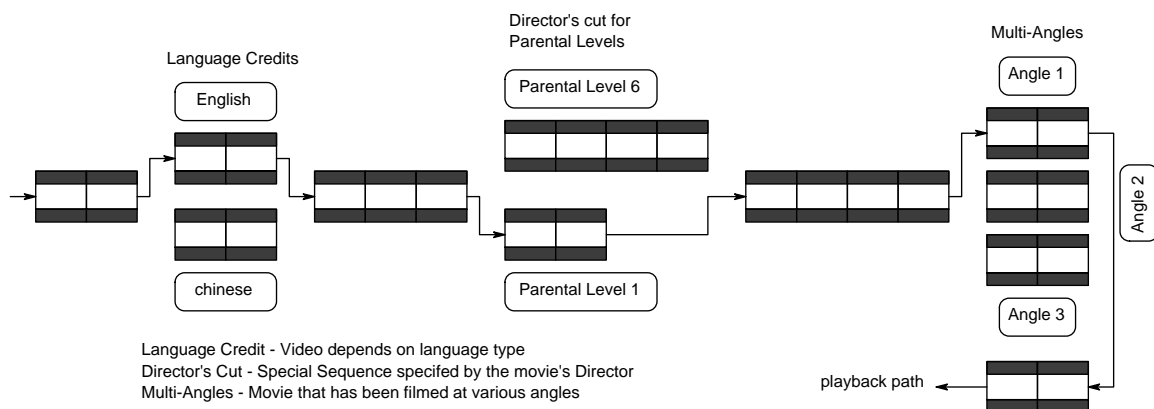
- Information for PGCI search
 1. 6 Menu IDs for Menu PGCI search (Title selection, Root menu defined by contents provider, Sub-picture selection, Audio selection, Angle selection and Part of title selection).
 2. Title search pointer table for Title PGCI search.
- Information for Presentation Data Search
 1. Part of title search pointer table for direct access to any titles or part of titles.
 2. Time Map table for time search
 3. Angle search information for changing viewing angles
 4. VOB search for trick play modes (Fast Forward, Fast Backward)

3. User Interface Control Unit – controls the permissions/prohibitions of user operations.
4. Navigation Control Unit – controls the presentation engine by handling 5 types of information,
 - Cell playback information
 - Cell playback order
 - Program playback control
 - PGC playback information
 - Navigation commands, which use 3 parameters,
 1. General parameters which are used as temporary memory by navigation commands.
 2. System parameters which reflect a DVD player's status and may be used to change player's action.
 3. Navigation timer – if this becomes 0, the DVD player terminates its current presentation and restarts at a pre-defined PGC.

The presentation engine follows presentation instructions issued by the Navigation control unit (in the Navigation Manager) to play the presentation data from the disc. Contiguous presentation data is placed in contiguous sectors on the DVD disc.

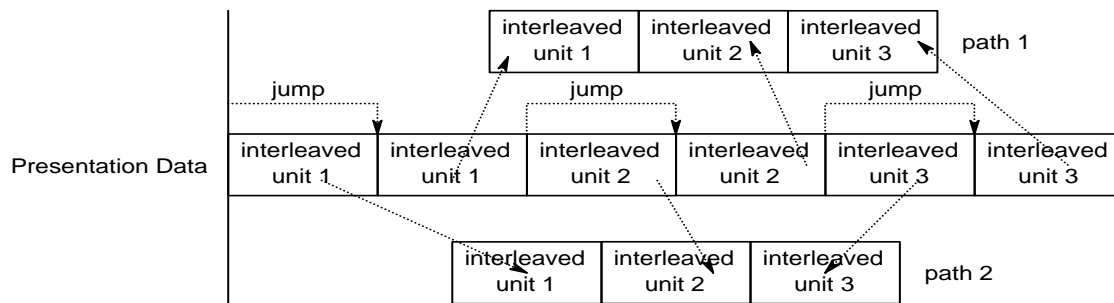
The presentation data is divided into cells. The Navigation manager issues presentation instructions for a cell. Thus the playback path of the presentation data is determined by a sequence of cells defined in a PGC. In addition to this one out of multiple cells will be selected at presentation time from an Angle Block.

Shown below are different playback paths for a movie. Different cells are presented depending on the Navigation Data and user input.



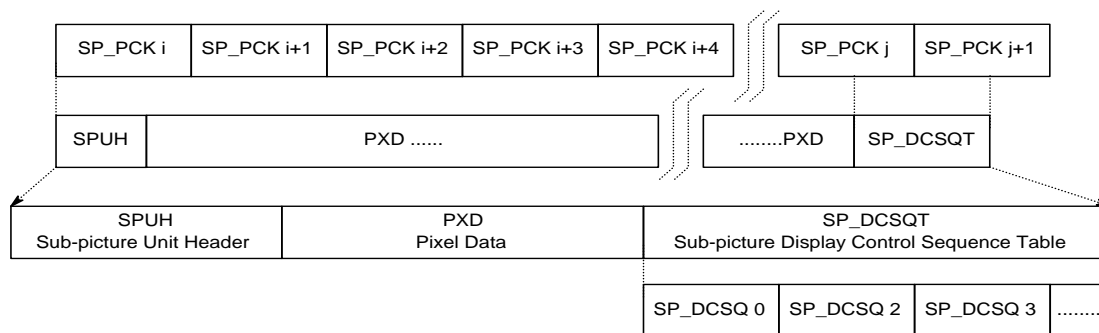
Along the playback path the data has to be presented seamlessly to the DVD demultiplexer and the Audio/Video/Sub-picture decoders. In a presentation segment where seamless play of multiple paths is carried out, the presentation data has an interleaving structure with the DVD drive head skipping blocks (or

sectors) which are not read. Smoothing out these jumps to deliver presentation data at an average pre-determined rate is handled by the **track buffer** as shown before.



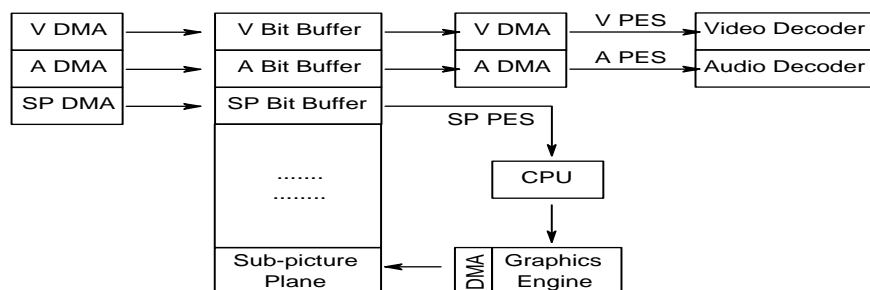
Handling and Decoding Sub-picture Packs

Sub-pictures are used to display sub-titles in a DVD environment. Sub-picture packs contain SPU (Sub-picture Units) which have the following structure,



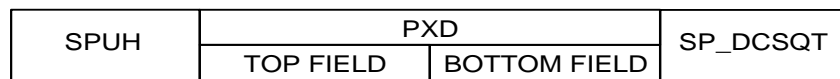
The sub-picture unit is divided into 3 parts as shown, sub-picture header, pixel data and display control sequence table – containing display control sequences). Since the SPU is contained within a sub-picture PES packet (SP_PKT), it has its own PTS which is used to synchronize it with the STC.

In the DVD environment shown the PES packets of the sub-picture packs are dma'ed into the sub-picture bit-buffer by the DVD de-multiplexer. From here they are extracted by the CPU, decoded and synced with the STC before rendering them via the graphics engine into the sub-picture buffer. Some DVD chipsets have a separate sub-picture decoder for decoding and displaying a SPU.

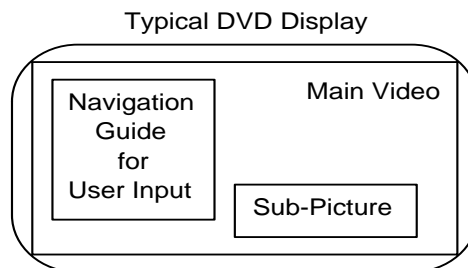


The SPU headers are shown below,

1. Sub picture unit header (SPUH) - contains the address information of the data found in the SPU.
 - SPU_SZ – size of the sub-picture unit – maximum size of a SPU is 53220 bytes.
 - SP_DCSQT_SA – start address of display control sequence table from 1st byte in a SPU.
2. Pixel Data (PXD) – this is the compressed data for each line in the actual bitmap using a specific run-length encoding method. Apart from the run-length encoding method (specific to the DVD sub-pictures), the actual bitmap data pixels have 4 types (00 – background, 01 – pattern, 10 – emphasis pixel 1, 11 – emphasis pixel 2). Also the pixel data is arranged in fields as shown below.



3. Sub-picture display control sequence table (SP_DCSQT) – this contains a list of display control sequences (SP_DCSQ #) to change the display start / stop of the SP and also the attributes (color, scrolling, blinking, display area, etc.) of the SP.



Notes:

1. **DOLBY® is a registered trademark of DOLBY LABS.**
2. **DTS® is a registered trademark of SONY Corporation.**
3. **A purely software based DVD player implementation (decode and playback of program streams) on a PC is relatively straightforward due to no real time OR error handling constraints like in a transport multiplex.**

TYPICAL DOCSIS® CABLE MODEM APPLICATION

A typical block diagram of a DOCSIS (Data Over Cable standard used in the US) cable modem chipset is shown in figure 90.

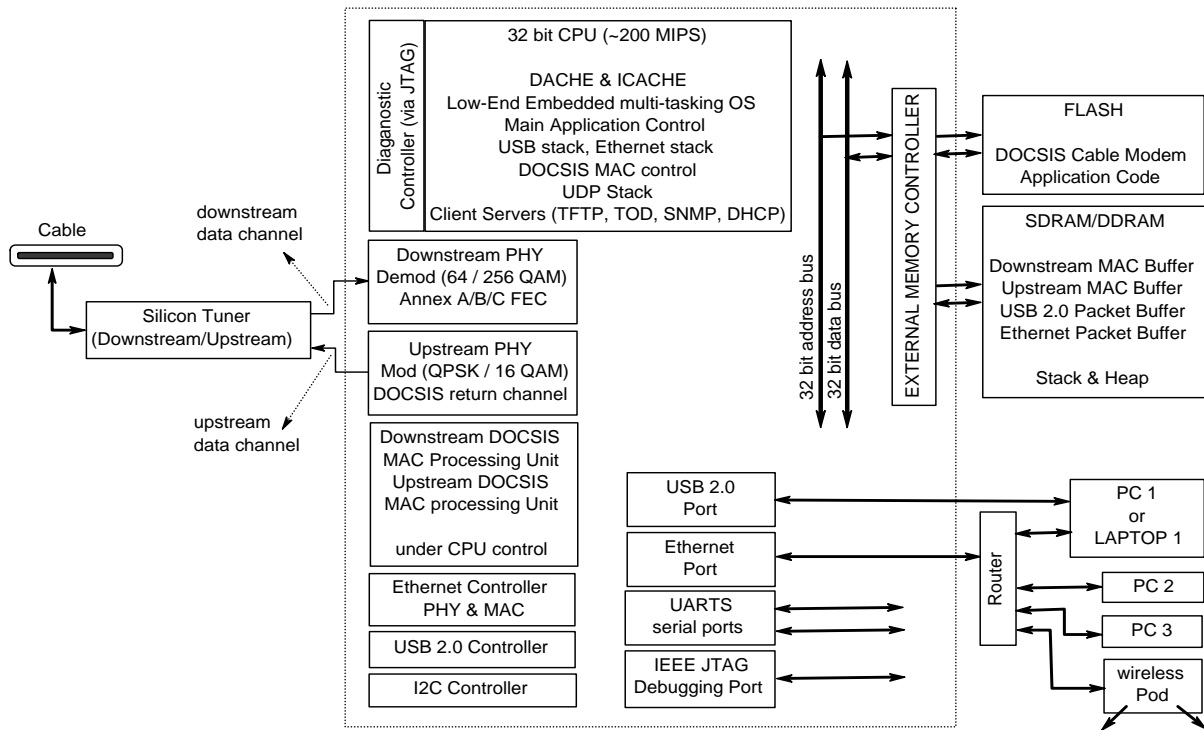


Figure 90. DOCSIS Cable Modem Chipset

A DOCSIS cable modem chipset is responsible for,

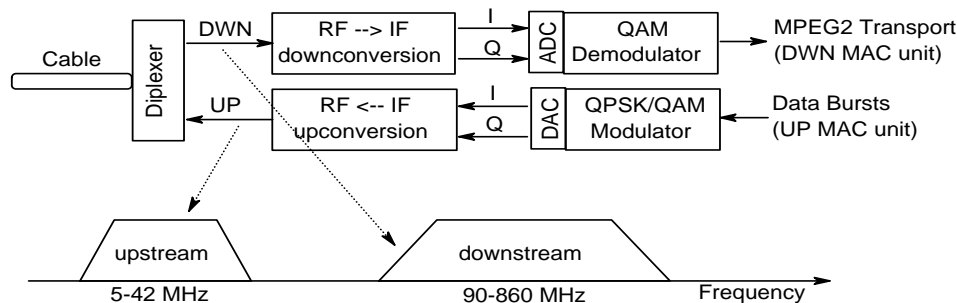
- Demodulating the downstream data channel in the cable feed, forward error correcting the demodulated signal, stripping off the MPEG headers from the packets, parsing the MAC headers, separating the MAC management messages from the user data payload (encapsulated in Ethernet packets), descrambling the MAC packet payload fields (could be timing data, Management messages or user PDUs) using the DES descrambler (in Cipher Block Chaining mode with either one of the CM's TEK keys) and forwarding Ethernet data packets via the Ethernet port to the PC or router. This is the downstream data channel (download of web pages, files, movies, etc.) which the user notices.
- Taking user responses (Ethernet upstream data packets), scrambling them using the DES algorithm and its latest TEK key, encapsulating them with MAC headers and bursting them into the upstream channel after modulating them using QPSK (or 16 QAM). Messages for communicating with the CMTS

are also DES scrambled (latest CM TEK key), encapsulated into MAC headers (as MAC management messages) and bursted into the upstream channel after modulation. This is the upstream data channel (user interaction for on-line gaming, pay-per-view movie requests, user keystrokes, etc.) or return channel the user notices.

The cable modem chipset has the following functional blocks,

1. The silicon tuner has 2 components,

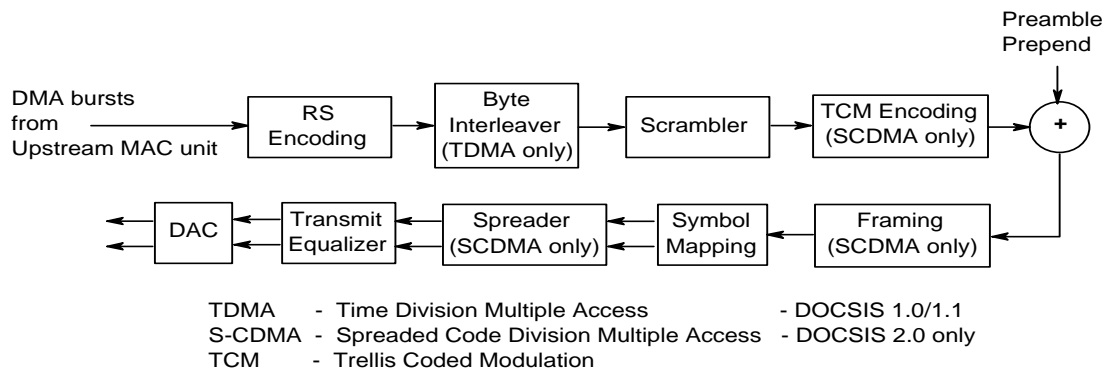
- Downstream RF → IF conversion block. It basically inputs the RF cable signal, locks to a downstream data channel (DOCSIS downstream data channels have a 6MHz BW and are in a certain RF frequency range) specified by the application, demodulates it to an IF signal and sends the IF QAM modulated I/Q components to the downstream demodulator in the cable modem chipset.
- Upstream IF → RF conversion block. It basically inputs the IF QPSK or 16 QAM modulated I/Q components from the upstream modulator in the cable modem chipset, up-converts it to an RF signal (DOCSIS upstream data channels have a 200KHz-6400KHz BW and are in a certain RF frequency range) and combines it into the cable's RF signal.



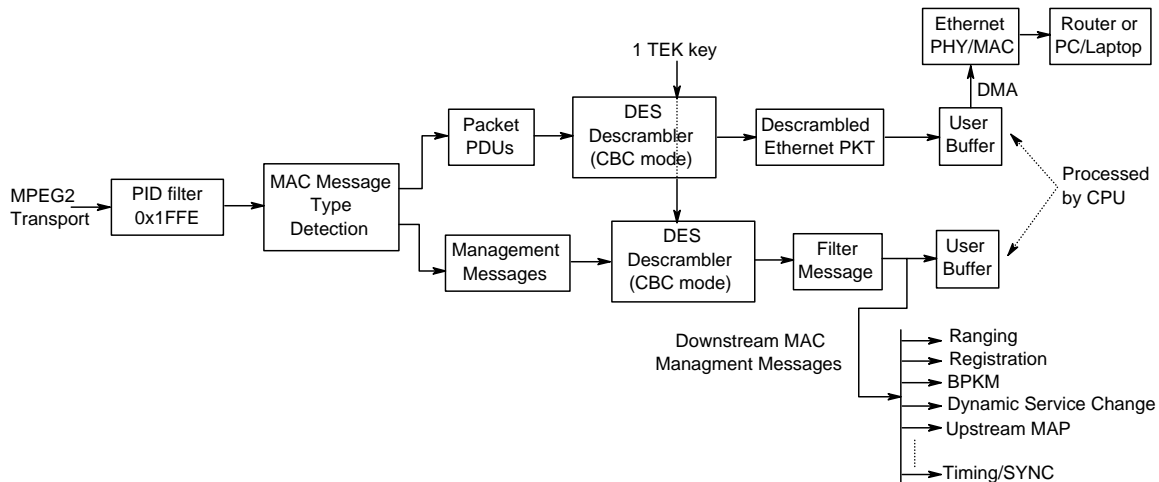
2. Downstream de-modulator inputs a 64/256 QAM modulated analog signal belonging to the DOCSIS downstream data channel, digitizes the I/Q components, equalizes the digital components to remove ISI and ghosting if any, de-rotates the I/Q components to remove residual phase errors, QAM demodulates (using trellis/convolution decoding and symbol de-mapping) the components and sends the output bit-stream to the forward error corrector (FEC). The output of the FEC is a MPEG2 transport stream which contains DOCSIS data packets with PID 0x1FFE which are passed along to the MAC downstream unit. The downstream de-modulator follows the Annex B standard.
3. Upstream modulator takes the DMA'ed bursts from the upstream MAC unit, Reed Solomon (RS) encodes the bit-stream, optionally interleaves the RS blocks (depending on the upstream multiplexing method), scrambles the blocks, optionally sends them to a trellis encoder (depending on the upstream multiplexing method, TDMA or S-CDMA), adds the physical layer preamble

(and guard interval), optionally converts the data blocks into frames (depending on the upstream multiplexing method), maps them into QPSK or 8/16/32/64/128 QAM symbols, optionally spreads these symbols (depending on the upstream multiplexing method), equalizes the I/Q components before passing them through a DAC stage and to the silicon tuner.

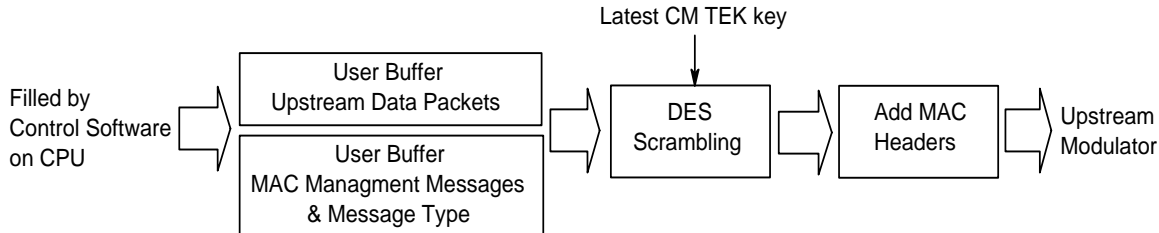
Differential encoded QPSK and 16QAM are available for TDMA channels. QPSK, 8QAM, 16QAM, 32QAM, and 64QAM are available for TDMA and S-CDMA channels. TCM encoded QPSK, 8QAM, 16QAM, 32QAM, 64QAM and 128QAM are available for S-CDMA channels.



4. The downstream DOCSIS MAC inputs a MPEG2 transport stream. It filters all the transport packets having PID 0x1FFE (DOCSIS PID). It then parses the MAC headers and optional Ethernet headers in the packet to determine the packet type (Data Packets or MAC management messages sent down by the CMTS). Then it does 2 things depending on the packet type,
 - Parses the MAC destination address to determine if the data packet belongs to this CM. If the MAC address matches, the data packet is sent to the DES descrambler (armed with either one of the cable modem's TEK keys) which descrambles the payload in cipher block chaining mode (CBC) of the DES algorithm, performs a CRC check on the entire Ethernet packet and then loads it into a downstream user buffer. The control software (CPU) periodically empties this user buffer and forwards the IEEE 802.2 Ethernet packets to the CPE (PC, Laptop, Router, PDA, etc.)
 - Parses the MAC destination address to determine if the MAC management message belongs to this CM. If the MAC address matches, the MAC management message is first descrambled using the DES descrambler (armed with one of the CM's TEK keys) and then further parsed to determine the message type. The messages and the message type notifications are put into a downstream user buffer. The control software reads this user buffer and takes appropriate action, depending on the MAC management message sent down by the CMTS (Ranging, Registration, Dynamic Service Change, BPKM, upstream bandwidth MAP, etc.).



5. The upstream DOCSIS MAC processing unit is responsible for taking the MAC management messages (and message types) and upstream Ethernet data packets from a upstream user buffer (filled by the CPU control software), scrambling them using the DES algorithm (with the latest CM TEK key), adding appropriate MAC headers to them and then bursting them via a DMA to the upstream modulator. The parsed upstream bandwidth allocation MAP is used to DMA (in TDMA or S-CDMA modes) the appropriate MAC management messages (REQ, REQ/DATA, Initial Maintenance, Periodic Maintenance, DATA, etc.) to the upstream modulator.



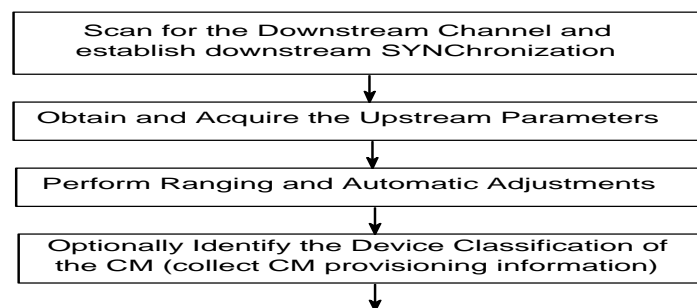
6. The CPU has the following responsibilities,
 1. Runs the control software which interacts with the state machines running on the downstream and the upstream MAC processing units.
 2. Controls the downstream de-modulator and the upstream modulator.
 3. Controls the silicon tuner via I2C for changing frequencies as per the selected downstream channel or the selected upstream channel.
 4. Runs the software state machines which use the MAC control units to perform functions and protocols like CM initialization, initial ranging, periodic ranging, CM registration, BPKM, dynamic service changes, etc with the CMTS.
 5. The CPU also takes the upstream bandwidth allocation "MAP" message transmitted by the CMTS from the downstream user buffer, parses its contents, and writes them into the upstream MAC control unit. This is then

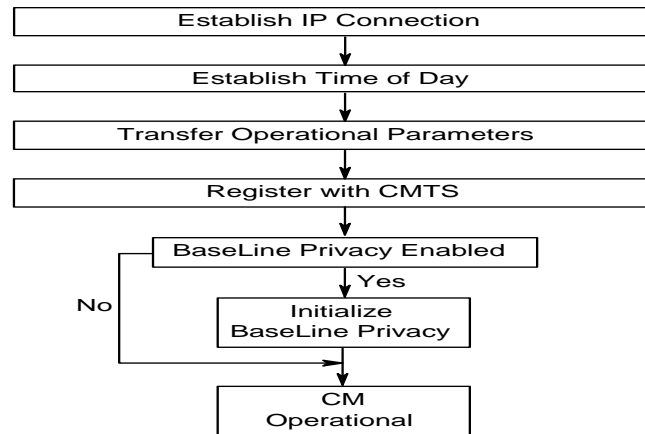
used by the upstream MAC unit to transmit MAC management messages and user data to the CMTS in TDMA or S-CDMA mode.

6. Run the high level Network “Client” software like DHCP (this is used to provide the CM a dynamic IP address), TFTP (this is used to download operational and configuration files into the CM), TOD (this is used to give the CM the correct time and date), SNMP (this is used to manage the network protocol on which the CM is attached).
7. Run the UDP software stack (Universal Datagram Protocol is a simplified version of the TCP/IP stack) which sits between the IP packets and the network layer client software. This is used to send/receive IP packets from the network layer clients.
8. Run the software stack for the Ethernet controller to transfer IEEE 802.3 data packets to/from the CPE (or router) & MAC control units using the Ethernet packet buffers.
9. If the CPE is connected to the cable modem via a USB 2.0 port, the CPU has to run the USB 2.0 software stack to input/output USB data packets to/from the CPE and the MAC control units. The USB packet buffers are used for this.

CM and CMTS Interaction

Once the cable modem is booted (started), a series of transactions (using specific protocols) are invoked between the cable modem and the CMTS to ensure proper operation of the cable modem. These transactions are done using downstream (CMTS) and upstream (CM) MAC management messages and occur on a periodic basis. The *control software* running on the CPU is responsible for all these transactions via various hardware blocks in the cable modem chipset (MAC control units, demodulators and modulators).





- On initialization or a “Re-initialize MAC” operation, the cable modem must acquire a downstream channel. The CM must have non-volatile storage in which the last operational parameters are stored and first try to re-acquire this downstream channel. If this fails, it must begin to continuously scan the 6-MHz channels of the downstream frequency band of operation until it finds a valid downstream signal.

A downstream signal is considered valid when the modem has achieved the following steps,

- Synchronization of the QAM symbol timing
 - Synchronization of the FEC framing
 - Synchronization of the MPEG packetization
 - Recognition of SYNC (applies to all upstream channels) downstream MAC messages
- After synchronization, the CM must wait for an upstream channel descriptor message (UCD) from the CMTS in order to retrieve a set of transmission parameters for a possible upstream channel.

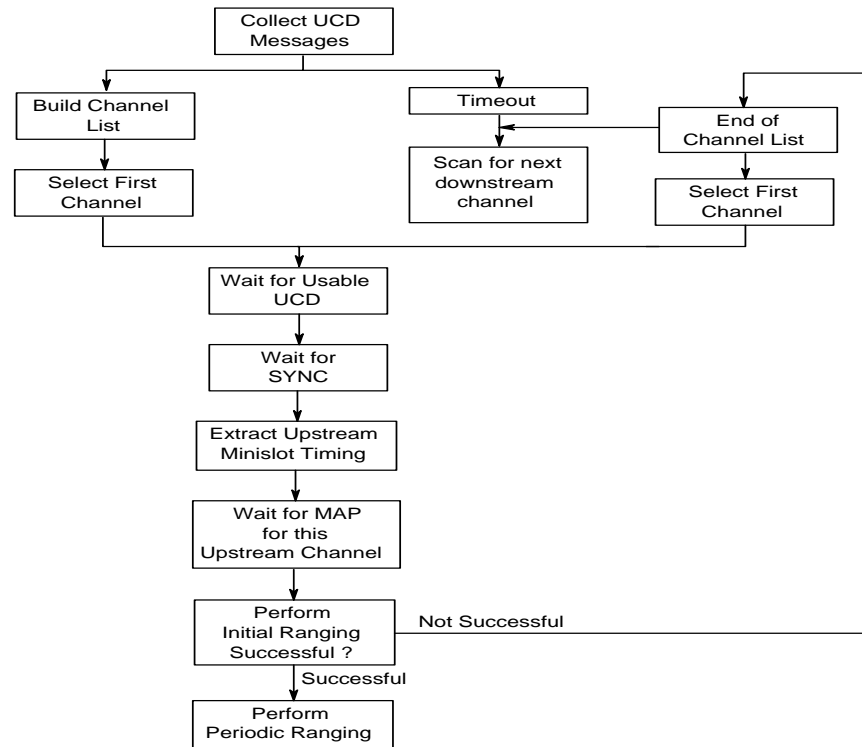
These messages are transmitted periodically from the CMTS for all available upstream channels and are addressed to the MAC broadcast address. The CM must determine whether it can use the upstream channel from the channel description parameters.

The CM must collect all UCDs with different channel ID fields to build a set of usable channel IDs. If no channel can be found after a suitable timeout period, the CM must continue scanning to find another downstream channel.

The CM must determine whether it can use the upstream channel from the channel description parameters. If the channel is not suitable, the CM must try other channels until it finds a usable channel.

If the channel is suitable, the CM must extract the parameters for this upstream from the UCD. It must then wait for the next SYNC message and extract the upstream mini-slot timestamp from this message. The CM must then wait for an upstream bandwidth allocation MAP for the selected channel. It may begin transmitting in the upstream direction in accordance with the MAC operation and the bandwidth allocation mechanism.

The CM must perform initial ranging at least once. If initial ranging is not successful, the next channel ID is selected, and the procedure restarted from UCD extraction. When there are no more channel IDs to try, the CM must continue scanning to find another downstream channel.



- Initial ranging or power adjustment of a CM's transmission parameters is done using, RNG-REQ, RNG-RSP and MAP (containing initial maintenance opportunity) MAC messages.
- Periodic ranging (by the CM) is achieved using the MAP (containing station maintenance opportunity with that CM's service ID - SID) message sent periodically by the CMTS.
- IP connectivity (or setting up of IP parameters for the CM) is achieved by invoking DHCP in order to obtain an IP address from the DHCP server. The DHCP-discover, DHCP-offer, DHCP-request and DHCP-response MAC messages achieve this. The DHCP response contains the name of a file which contains further configuration parameters.
- The CM needs to have the current date and time. This is required for time-stamping logged events which can be retrieved by the network management system. This is achieved using the time-of-day request and time-of-day response MAC messages. The request and response messages are transferred via the UDP stack and the TOD client.
- After DHCP is successful, the modem must download the configuration parameter file using TFTP.
- A CM is authorized to forward traffic into the network once it is initialized and configured. The CM is authorized to forward traffic into the network via registration. To register with a CMTS, the CM forwards its configured class of service and any other operational parameters in the configuration file to the CMTS as part of a Registration Request. Once the CM has sent a Registration Request to the CMTS it must wait for a Registration Response to authorize it to forward traffic into the Network. The Registration Response indicates that the CM is now registered with the CMTS and the network.
- Typically 1 upstream and 1 downstream Service Flows (2 Service Flow IDs - SFIDs) are setup between the CMTS and the CM after the CM is registered.

- The CMTS may change any of burst characteristics of an upstream channel by providing all the CMs using that upstream channel with the Upstream Channel Descriptor (UCD) message containing the changes.
- Also any time after registration the CMTS may request/direct the CM to change its upstream channel via a UCC-REQ message (done for traffic balancing and noise avoidance on the upstream).
- Service Flows between the CMTS and the CM may be created, changed or deleted. This is accomplished through a series of MAC management messages referred to as Dynamic Service Addition (DSA), Dynamic Service Change (DSC) and Dynamic Service Deletion (DSD). The DSA messages create a new Service Flow. The DSC messages change an existing Service Flow. The DSD messages delete a single existing Upstream and/or a single existing Downstream Service Flow.

BPKM (Baseline Privacy key Management) in the CM Environment

After the CM is registered with the CMTS and an upstream and a downstream service flow has been established, the CM must run the baseline privacy key management operations if the Baseline Privacy is enabled in the configuration file (downloaded via TFTP by the CM).

This is needed primarily to extract and de-crypt the 2 TEK keys (provided by the CMTS to the CM at regular time intervals) for,

1. Descrambling (with DES using any 1 TEK key) the data payload field in the MAC frames (could be MAC management messages or Ethernet data packets).
2. Scrambling (with DES using the latest TEK key) the data payload field in the MAC frames before bursting them via DMA to the upstream modulator.

The Baseline Key Management Protocol has the following features,

- After a CM completes DOCSIS Registration, it initiates an Authorization Key exchange sequence with the CMTS. On first receipt of an Authorization Request message from the unauthorized CM the CMTS initiates the activation of a new Authorization Key (AK), which is sent to the CM in an Authorization Reply message. This AK will remain active until it expires according to its predefined life-span.
- The Authorization key in the Authorization Reply message from the CMTS is encrypted using the RSA public-key encryption method, using the cable modem's public key.
- The HMAC-Digest written into the Authorization Key Reply message is calculated using the CM's Authorization Key.
- A set of 2 authorization keys are always maintained by the CM (old and new AK). A CM periodically refreshes its Authorization Key by re-issuing an Authorization Request to the CMTS.
- The cable modem uses the latest of its 2 most recent Authorization Keys to calculate a HMAC-Digest and attaches it to the Key Request Message for obtaining the TEK keys from the CMTS.
- The CMTS uses the CM's active Authorization keys to verify the HMAC-digest in the Key Request message received from the CM.
- The CMTS then sends to the CM a Key Reply message with has a set of 2 TEK keys (encrypted in Triple DES mode Encrypt-Decrypt-Encrypt using the CM's authorization key) and a HMAC-Digest (this is also calculated using the CM's authorization key).

- The CMTS uses the latest of the 2 active TEK keys for scrambling downstream traffic (DES in CBC mode) to the CM.
- The CM first verifies the HMAC-Digest in the TEK key Reply message (using 1 of its active authorization keys), then decrypts the 2 active TEK keys (sent by the CMTS) and uses any one of these TEK keys to descramble the downstream traffic.
- The CM also uses the latest of the 2 active TEK keys to scramble the upstream traffic that is sent to the CMTS.
- The CMTS uses any one of the 2 active TEK keys for de-scrambling upstream traffic (DES in CBC mode) sent by the CM.
- The CMTS always uses an active AK when calculating HMAC-Digests in Key Replies, Key Rejects and TEK Invalids, and when encrypting the TEK in Key Replies.
- The CM uses either of its 2 most recent AKs to authenticate Key Replies, Key Rejects or TEK Invalids, and to decrypt a Key Reply's encrypted TEK.
- The CM uses the latest of its 2 most recent Authorization Keys when calculating the HMAC-Digests it attaches to Key Requests.

The CMTS encrypts the each TEK key in the Key Reply messages it sends to a CM. The 64 bit TEK key (plain-text) is encrypted using a two-key triple DES algorithm in the encrypt-decrypt-encrypt (EDE) mode.

Encryption at the CMTS, $C = E_{K1} [D_{K2} [E_{K1} [P]]]$

Decryption at the CM, $P = D_{K1} [E_{K2} [D_{K1} [C]]]$

P Plain-text 64-bit TEK

C Cipher-text 64-bit TEK

K1 Left-most 64 bits of the 128-bit KEK

K2 Right-most 64 bits of the 128-bit KEK

E [] 56-bit DES ECB (electronic code book) mode encryption

D [] 56-bit DES ECB decryption

The KEK Key is derived from the Authorization Key. Also the HMAC-Digests (or message authentication keys) for downstream/upstream message authentication are calculated from the Authorization Key.

Network Layer Software Stacks running on the Cable Modem

Below we have shown the typical client software's running on top of the UDP software stack on a Cable Modem.

SNMP Client Software	TOD Client Software	TFTP Client Software	DHCP Client Software
UDP Software Stack Universal Datagram Protocol			
IP Packets			

UDP software stack – this is a simplified “TCP/IP like” Universal Datagram Protocol software stack which is used to route the IP packets to/from the network layer clients.

TOD client – this is the Time of Day client software which is responsible for interacting with the TOD server on the CMTS side to give the cable modem the current time and date during the CM registration process.

TFTP client – this is the Trivial File Transfer Protocol client software which is responsible for downloading the configuration parameter file from the TFTP server during the CM registration process. This configuration file provides a table of TLV (type, length, value) elements which specify all the operating and network management parameters for the CM.

The configuration file typically contains the following,

- Network Access Configuration Setting
- CM MIC Configuration Setting
- CMTS MIC Configuration Setting
- End Configuration Setting
- Upstream Service Flow Configuration Setting
- Downstream Service Flow Configuration Setting

Optionally it may also contain,

- Downstream Frequency Configuration Setting
- Upstream Channel ID Configuration Setting
- Baseline Privacy Configuration Setting
- Software Upgrade Filename Configuration Setting
- Upstream Packet Classification Setting
- Downstream Packet Classification Setting
- SNMP Write-Access Control
- SNMP MIB Object
- Software Server IP Address
- CPE Ethernet MAC Address
- Maximum Number of CPEs
- Maximum Number of Classifiers
- Privacy Enable Configuration Setting
- Payload Header Suppression
- TFTP Server Timestamp
- TFTP Server Provisioned Modem Address
- Pad Configuration Setting
- SNMPv3 Notification Receiver
- Enable DOCSIS 2.0 Mode
- Enable Test Modes

DHCP client – the Dynamic Host Control Protocol client software is responsible for providing an IP address to the Cable Modem every time it reboots.

SNMP client – the Small Networks Management Protocol (version 1/2/3) is used as the communication protocol for supporting network management of all DOCSIS services in the cable modem environment. A minimum set of managed objects are required in the Management Information Base (MIB) to support the management of CM. These MIB objects are provided in the configuration file downloaded by the CM.

Notes:

1. DOCSIS® is a registered trademark of CABLELABS.
2. Service Flow is a MAC-layer transport service which does 2 things,
 - Provides unidirectional transport of packets from the CM to the CMTS
 - Shapes, polices, and prioritizes traffic as per the Quality of Service (QoS) traffic parameters defined for the Flow.
3. Service Flow Identifier (SFID) – is an identifier assigned to a service flow by the CMTS.
4. Service Identifier (SID) – is an additional identifier assigned by the CMTS (in addition to a SFID) to an Active or Admitted Upstream Service Flow.
5. MAC address is the “built in” hardware address of the device (CM) connected to a shared network.
6. IP address is the “software” address allocated to the device (CM) connected to a shared network (can be provided dynamically via DHCP or fixed).
7. Security Association Identifier (SAID) – is a Baseline Privacy security identifier between a CM and the CMTS.
8. Trivial File Transfer Protocol (TFTP) – is an internet protocol for transferring files without the requirements for user names and passwords. Typically used for automatic downloads of data and software (configuration file is downloaded by the TFTP client running on the CM from the TFTP server at the cable head-end).
9. Message Integrity Check (MIC) – on the CM and CMTS side is calculated by performing an MD5 digest over all the bytes in the configuration file settings. This is then added to the configuration file as a CM MIC or CMTS MIC.

CABLE TV SETTOP WITH INTEGRATED DOCSIS CABLE MODEM

A Cable TV SETTOP Box with an integrated Cable Modem chipset typically does 3 things,

- QAM demodulates RF channels (containing A/V programs) on the cable feed under user control to output a transport stream (or transport multiplex) having A/V transport packets and transport packets containing the PSI tables (PAT, PMT, NIT, EIT, SDT, TDT tables). The PSI tables are then used to build the Electronic Program Guide, with which the user can interact to select and decode specific A/V programs using the A/V decoder.

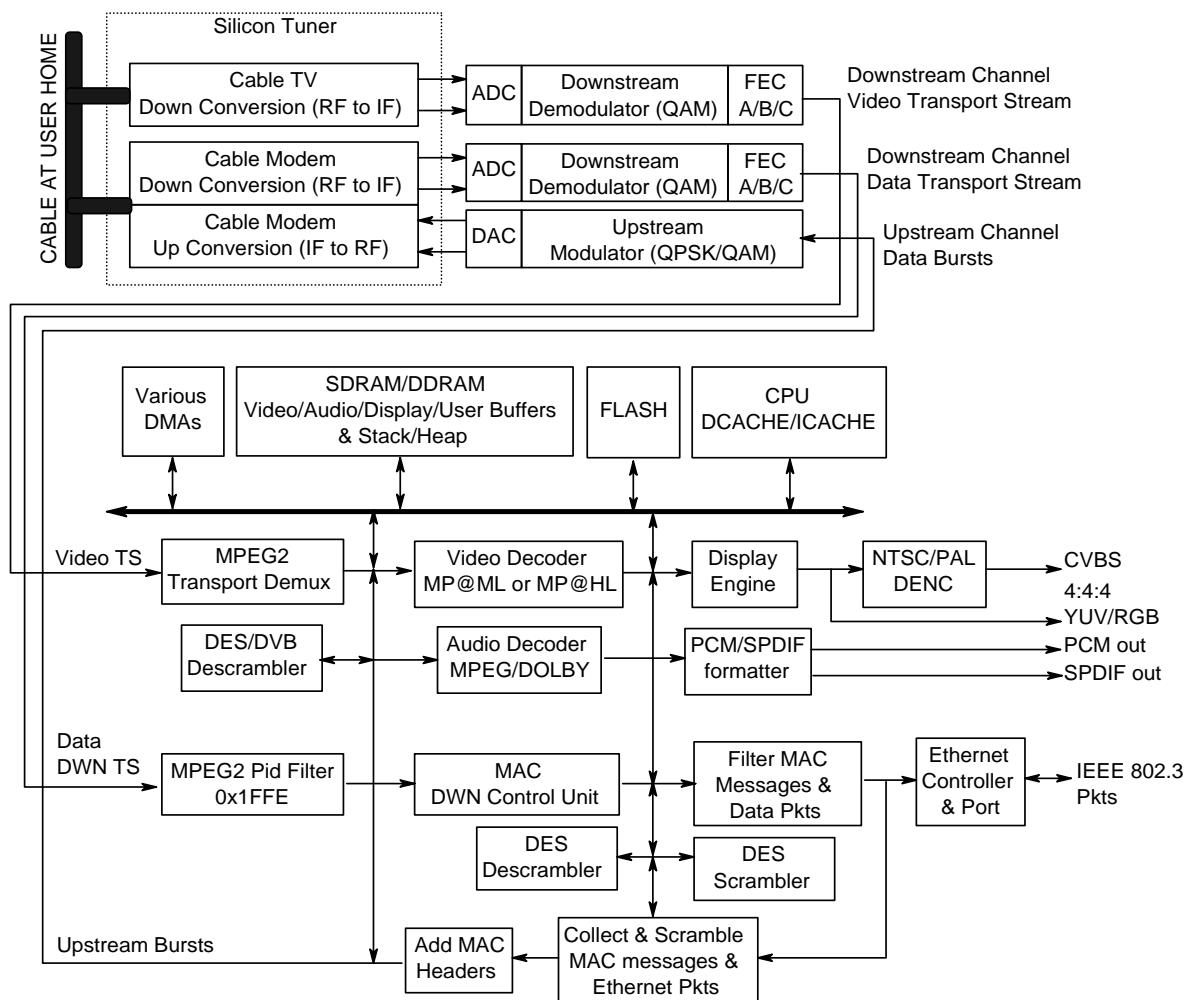
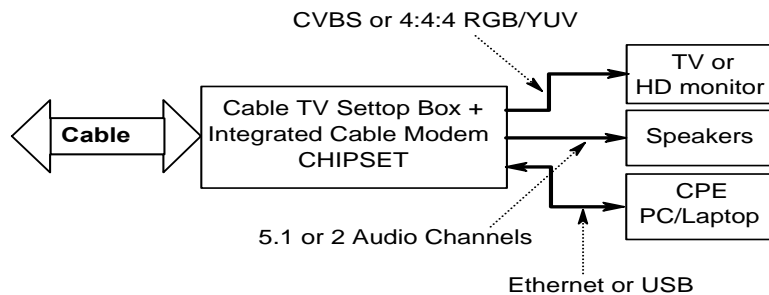


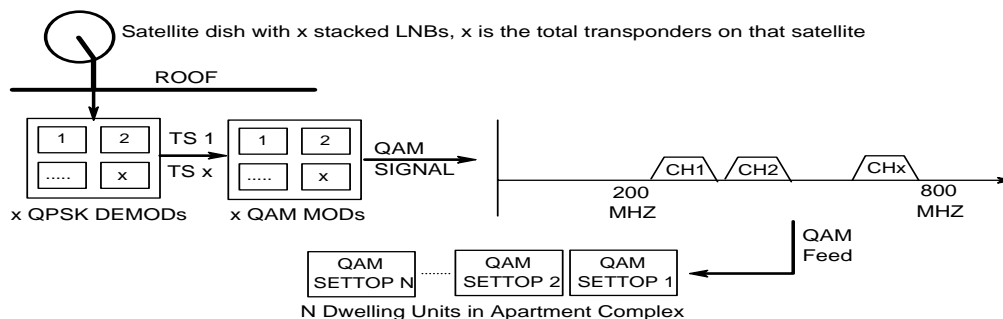
Figure 91. Cable TV + DOCSIS Cable Modem Chipset

- QAM demodulates RF downstream data channels (under CPU host software control) on the cable feed to obtain a transport stream multiplex in which the transport packets corresponding to PID 0x1FFE (DOCSIS PID) contains downstream DOCSIS data (via CMTS). The MAC messages are extracted from these transport data packets by the downstream MAC control unit. The payload in these messages consists of MAC management messages and Ethernet data packets. The control software on the CPU uses these MAC management messages to perform several control and authorization tasks with the CMTS (at head-end) in order to register the Cable Modem and start downstream/upstream service flows between the CM and the CMTS. The Ethernet packets are forwarded by the control software to the CPE.
- The control software is also responsible for bursting MAC management messages (constructed as responses to the downstream MAC management messages) and user Ethernet data packets (from the CPE) to the upstream modulator which QPSK/QAM modulates these bursts and inserts them into the upstream channel allocated to the CM. The upstream MAC management messages are used by the control software to interact with the CMTS for registering the cable modem and also to maintain the downstream/upstream service flows between the CM and the CMTS.



Notes:

1. A trans-modulation technique is used in apartment complexes wherein a QPSK satellite signal is beamed onto a stacked (x transponders) LNB satellite dish on the Roof. Each QPSK RF signal corresponding to each of the "x" transponders on that satellite is demodulated using a QPSK DEMOD, trans-modulated into 6/8 MHz QAM channel and inserted into the QAM spectrum. This cable feed is sent to each dwelling unit in the apartment complex, which have a QAM cable SETTOP box.



DESKTOP & EMBEDDED APPLICATIONS

The hardware & software used to build digital speech/Image processing and consumer electronics applications can be divided into 2 categories,

- Desktop Applications – here the hardware is a standard desktop computer (typically PC or a MAC) with an Intel/IBM processor, graphics card, sound card, Ethernet card, USB card, hard disk and a DVD ROM/RW drive. The speech or image processing software application (typically a device driver for accessing hardware components or a simple graphical applet) runs on the processor of the desktop computer and uses its associated hardware peripherals for receiving/sending input/output (graphics card, sound card, Ethernet/USB card, hard disk, DVD, etc.).

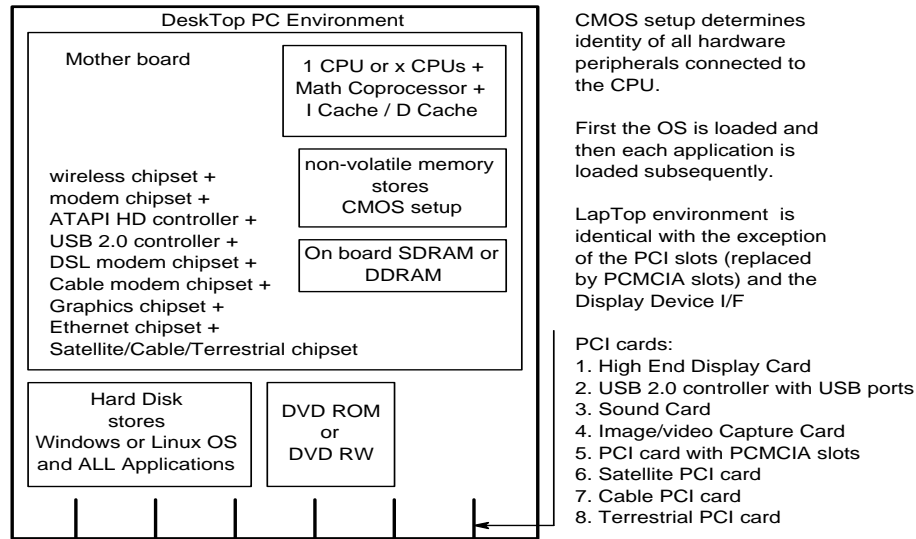
A desktop application is stored on the hard disk of the desktop computer and is loaded up and made available to the user after the multi-tasking desktop operating system (Windows or MAC OS) is operational (this is the first software component loaded on any desktop computer before any application can be invoked). Thus, most desktop applications are unsuitable for meeting real time criteria because of this slow and random boot up phase. Also in a typical desktop environment there could be several user applications which could be invoked randomly. When these applications are run along side the digital signal/image processing application which needs to operate under real time conditions, they could potentially harm/block this application due to sharing of resources (memory, processing power, etc.) on the desktop.

- Embedded Applications – here the hardware and software of the speech or image processing application is standalone. The hardware (processor and associated peripherals) in an embedded application may be very similar to those in a desktop environment, with the exception that the entire application (multi-tasking OS and the application code) is stored and booted from FLASH memory. This ensures a fast boot up and operational phase.

Also parts of the application (not needing a multi-tasking environment) could be invoked and run before the OS is loaded and made operational. Typically the OS in an embedded application does not have a lot of frills and features except those needed for running that specific user application. Thus embedded applications are ideally suited for systems which need to operate under real time conditions.

Most consumer electronics applications fall in the embedded group. Most word processing applications, database applications, any application run via an internet browser, non-real time speech/imaging/video applications fall in the desktop group.

PC/MAC Based Desktop Environments



- Hardware 1 – Intel® 32 bit Pentium® 4 processor, with 512 MB of SDRAM or DDRAM, high end 3D graphics card (from ATI® or NVIDIA®), 16/20 bit sound card (with 2 speakers and 1 microphone), Ethernet 10/100 Base-T NIC card, USB 2.0 controller card with 4 ports, Internal 40GB 7200 RPM IDE hard drive (or 10,000 RPM SCSI drive), Internal DVD ROM or DVD RW and a high end image/video processing capture card (PCI or USB 2.0).
- Hardware 2 – all the above peripherals, but the processing power being provided by dual or quadruple 32 bit Xeon® Intel processors.
- Hardware 3 – all the above peripherals, but the processing power being provided by a 64 bit Intel Itanium® or a 64 bit AMD® Athlon® processor.
- Software 1 – Microsoft® XP® operating system, Microsoft Visual C/C++ 7.0 Professional, Microsoft Device Driver Development kit (DDK) for XP, Windows Media Player® 9.0 Software Development kit (SDK), Microsoft Office® (word, excel, outlook) SDK.
- Software 2 – MathWorks® MATLAB® signal/image processing toolkit, MathWorks Simulink® toolkit and MathWorks M language to C code cross development toolkit.
- Software 3 - LeadTools® Speech/Image Processing libraries for engineering and medical applications.
- Software 4 – Video capture, editing and rendering software package from Pinnacle® Systems, still image editing and rendering software package

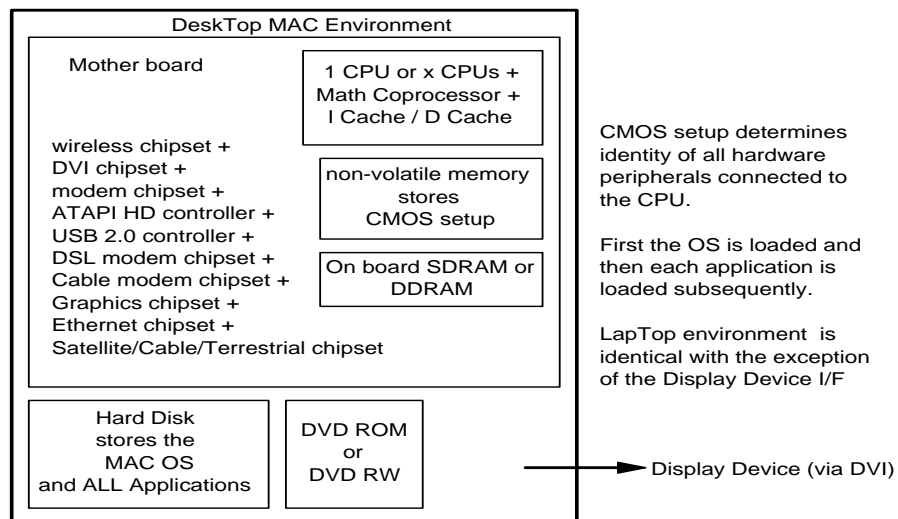
from Adobe Photoshop®, and animated 3D movie creating/editing and rendering software packages from SoftImage®.

- Software 5 – Eikona3D® 3D image processing software package for industrial or medical applications.
- Desktop applications using any of the hardware 1-2-3 and software 1-2-3-4-5 combinations,
 1. Device drivers can be developed for any peripheral connected to the PC/Laptop running windows XP. These could be camera phones off the USB port supporting USB 1.1/2.0 protocols, PCI based data modems off the telephone line (supporting V.90 protocol, etc.), any device off the serial/parallel ports (printers, etc.), PCI based sound cards (inputting speech via microphone and outputting speech via speakers), all kinds of internal storage devices (ATAPI hard disks, ATAPI SCSI hard disks) and all kinds of external storage devices connected via USB 2.0 ports (hard disks, DVD ROM/RW, tape drives).
 2. If the SCSI hard disk (fast data access time) is used instead of the standard IDE, then some engineering applications like packet injectors can be built to feed SETTOP box development platforms. Also various graphical analysis tools can be built for the streaming A/V data stored on the hard disk.
 3. Various custom databases or custom software modules integrated with existing databases.
 4. Various custom word processing packages or custom software modules integrated with existing word processing packages.
 5. Developing extensions to e-mail clients like Microsoft Outlook®.
 6. Several custom A/V streaming applications which interact (grab compressed or raw A/V data) with storage devices, A/V capture devices, or the internet to display video over a standard display device (PC monitor or laptop screen) and send audio over speakers (PC or Laptop speakers). These can be developed using the Windows Media Player SDK which allows a user to integrate the WM decoder & encoder in his custom application thus allowing decoding and encoding of several A/V compressed formats.
 7. Developing custom web-browsers and SPAM protection filters.
 8. Using the MATLAB toolkits, custom engineering applications in speech/image processing, control systems, aeronautical engineering,

medical and defense systems can be built using the M library files. These M files can be linked in with a user's C application to create standalone executables which can run on desktop or several embedded platforms. MATLAB libraries can also be used to interact with several Audio and Video capture devices (industrial strength camera's, medical microscopes, etc.)

9. C libraries developed by LeadTools form an effective way to create custom desktop speech/image processing or video applications. Also a wide array of functionality is provided for creating industrial/medical applications and also custom medical databases (with image & video processing options).
 10. Live video capturing/editing/rendering packages and development tools from Pinnacle systems offers excellent ways to capture video, modify it and store it on all kinds of storage media. Development tools from Adobe PhotoShop offers still image capture/editing and rendering capability. SoftImage provides different development tools variants (XSI/3D) for creating 3D animation movies, editing them and rendering them to a display device or a storage media.
 11. Eikona3D software development package provides complete source code for the most 3D image processing volume algorithms. One of its applications could be, developing off the shelf medical databases for MRI, Ultrasound, CT and PET images.
- Software 6 – RedHat® Linux Enterprise software variants (includes Linux kernel, compilers, software development tools, JAVA® development tools and several built in applications). Variants are WS, ES and AS.
 - Software 7 – Intel Compiler/Debugger for Linux, Intel Performance Libraries for Math and Graphics.
 - Desktop applications using any of the hardware 1-2-3 and software 6-7 combinations,
1. Using an Open Source Development Platform like Linux and the Intel Software Development Suite, one could create several applications involving digital audio and video processing or large imaging databases for industrial or medical applications.
 2. Linux being especially good for developing server and web-based applications. This aids in the creation of large industrial strength web based databases, industrial strength email servers and email clients.
 3. The “AS variant” of RedHat Enterprise Linux can be used for mission-critical, time-critical, and deterministic applications on desktops.

4. Linux offers a superior multi-tasking OS than Windows XP for multiple CPU's (more efficient in handling resources like memory and processing power distribution), so applications using numerous processes and threads can run more efficiently when the target hardware has multiple CPUs.
5. One advantage of Linux being royalty free is that a user's application can be bundled with the Linux kernel and installed on any hardware (or processor) supporting the Linux OS. This is especially advantages in embedded systems where Embedded Windows CE® and Windows XP® Operating systems have a royalty price tag.



- Hardware 4 – PowerPC® G4 32 bit or PowerPC® G5 64 bit processor, with 512 MB of SDRAM or DDRAM, high end 3D graphics card (from ATI® or NVIDIA®), 16/20 bit sound card (with 2 speakers and 1 microphone), Ethernet 10/100 Base-T NIC card, USB 2.0 controller card with 4 ports, Internal 40GB 7200 RPM IDE hard drive (or 10,000 RPM SCSI drive), Internal DVD ROM or DVD RW and a high end image/video processing capture card (PCI or USB 2.0).
- Hardware 5 – Multiple PowerPC® G5 64 bit processors with all the above peripherals.
- Software 8 – Mac® operating system X (version 10.1.3 or greater), Metrowerks CodeWarrior® Development Studio (version 8 or higher) which includes a 32/64 bit compiler, various wizards and a debugging and performance tuning environment.
- Software 9 – MathWorks® MATLAB® signal/image processing toolkit, MathWorks SimuLink® toolkit and MathWorks M language to C code cross development toolkit for the Mac OS.

- Software 10 – Alias® Maya® 3D animation movie creation, editing and rendering software package for the Mac OS.
- Desktop applications using any of the hardware 4-5 and software 8-9-10 combinations,
 1. Custom graphical (GUI) applications (involving still images, audio and video) can be built using the Metrowerks tools to run under the MAC® OS.
 2. Using the MATLAB toolkits and the M files to C code conversion utilities any standalone engineering application (speech/image processing, audio/video processing, industrial control systems, medical and defense) can be built and run on the PowerPC platform.
 3. The Maya software package for the Mac OS (running on a single or multiple G5 processors) is the best professional development package for creating/editing and rendering 3D animation movies.



Embedded Environments

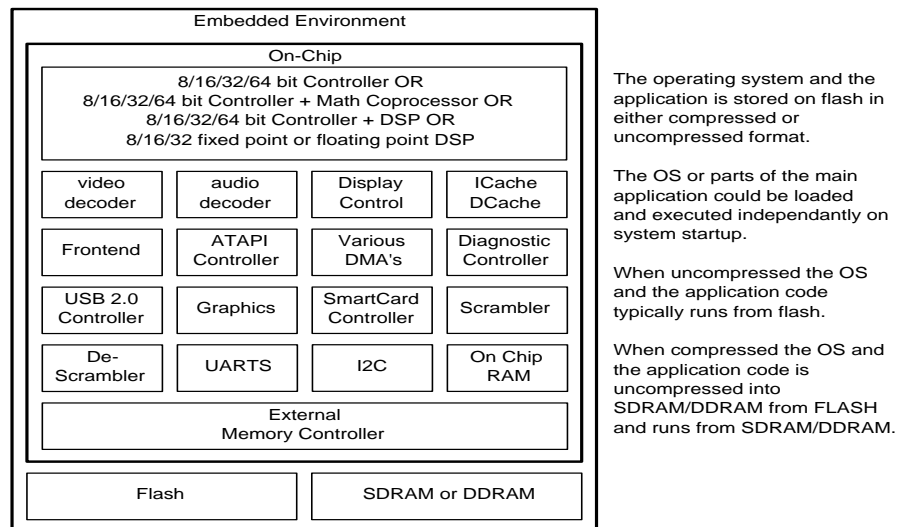
The micro-controllers used in an embedded environment can be divided into 4 classes,

- 8/16/32/64 bit processor with on-chip peripherals
- 8/16/32/64 bit processor with a math co-processor and on-chip peripherals
- 8/16/32/64 bit processor with a DSP and on-chip peripherals
- 8/16/32 bit fixed or floating point DSP with on-chip peripherals

Typical processors found in an embedded environment,

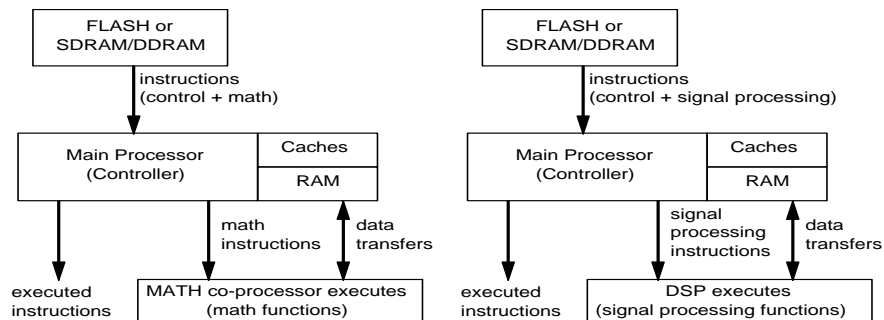
1. Motorola® 68000 family of 16/32 bit RISC/CISC processors
2. Intel 8052 8 bit CISC processors
3. ST Microelectronics® ST20® 32 bit RISC processors
4. Advanced RISC Machines (ARM®) 32 bit processors
5. MIPS® 32/64 bit RISC processors
6. Intel XSCALE® 32 bit scalable processors
7. Intel Pentium 2/3/4 32 bit CISC processors with math co-processor

8. Intel Xeon 32 bit CISC processors with math co-processor
9. Intel Itanium 64 bit CISC processors with math co-processor
10. Hitachi® SH3/SH4/SH4 processors with or without DSP cores
11. Motorola PowerPC 32 bit processor family with math co-processor
12. Apple/IBM PowerPC 32/64 bit processor family with math co-processor
13. TI TMS® DSP family of 16/32 bit fixed and floating point processors
14. Motorola DSP family of 16/32 bit fixed and floating point processors
15. Analog Devices DSP family of 16/32 bit fixed and floating point processors



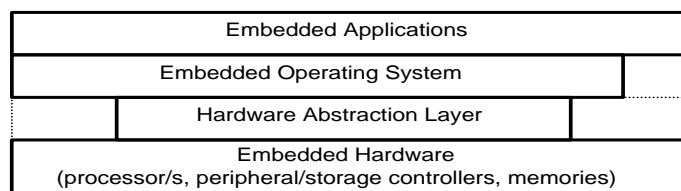
Processors which have a math co-processor typically parse/split the application code into 2 parts, control and math intensive functions. The control functions are handled by the processor, whereas the math intensive functions are sent to the math co-processor which performs the requisite math operations on the data and sends back the results to the main processor.

Processors (SH family) which have a DSP core for handling signal processing functions typically parse/split the application code into 2 parts, control and signal processing functions. The signal processing functions are sent to the DSP core which performs the requisite signal processing operations on the data and sends the results back to the main processor.



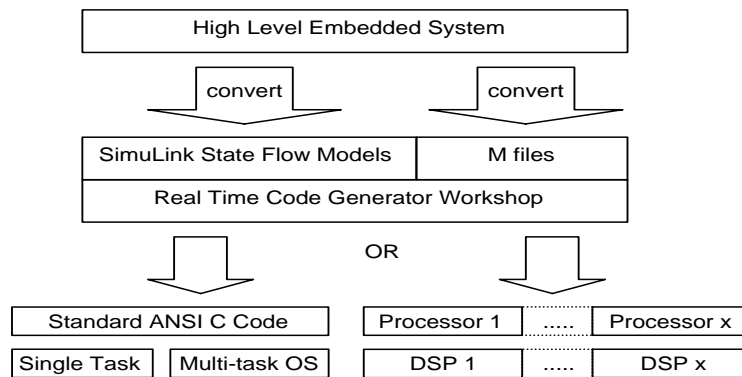
The 3 widely used embedded operating systems for 32/64 bit processors are,

1. Embedded MS Windows CE® and Embedded MS Windows XP®.
 - Windows CE and its development tools (DDK – device driver development toolkit for the embedded arena) are offered by Microsoft. This OS is widely used in consumer electronic devices like Personal Digital Assistants (PDA), Global Positioning Systems (GPS), Still/Video Phones, SETTOP boxes, Cable Modems and some high end digital TV's. Windows XP (and its DDK) is widely used in developing standalone test equipment like logic analyzers, digital oscilloscopes and some monitoring systems.
 - Windows CE & XP in a real time embedded system is always used with hardware assist (some critical time dependant functions in the system are implemented in hardware). This is due to the fact that the context switch time (time for the OS to switch between 2 same or different priority tasks/threads, or between 2 same or different priority interrupts, or between a task/thread and an interrupt) is large and mostly non-deterministic. Thus in a real-time system only the control and graphics layer is provided by WinCE & WinXP.
 - The advantage of using WinCE and WinXP is that it supports almost every 32 bit embedded processor family. Also the development tools (e.g. Platform Builder for CE and XP) offered by Microsoft contains drivers for almost every possible peripheral that goes with a supported processor (e.g. Wired or Wireless Ethernet, Bluetooth, telephone modems, sound and display devices, mouse and pointing devices, etc.). Thus an embedded application using any one of the CE/XP supported processors can be developed within a short period of time.
2. WindRiver® VxWorks® real time operating system and its associated Tornado® development tools for most 32 bit embedded processors are widely used for mission-critical, time-critical, safety-critical and deterministic embedded applications in the defense, automotive and industrial automation industries.
3. FSMLabs® embedded Linux is a complete development system with the Linux kernel and tool-chain, a fully embeddable/network-bootable file-system, and utilities for building efficient flash file systems. The development tools come with the Linux kernel with file-systems, full TCP/IP networking, X-Windows Graphical User Interface, and development tools.



Using MathWorks toolkits for developing an embedded application

MathWorks offers various toolkits like Matlab M file libraries, SimuLink block based state flow models, a Real Time Code Generator Workshop which can take a high level embedded state flow model and convert it either into processor independent 8/16/32 bit fixed or floating point ANSI C code and/or generate assembly language code for (many) supported embedded or digital signal processors. The Real-Time Workshop supports single-tasking or multitasking operating systems and "bare board" (no operating system) environments. It also enables you to generate inline code for custom blocks, including signal and image manipulation algorithms and device drivers.



Different Processor Types in Desktop/Embedded Environments

The performance of a processor is determined by the time taken to run a program on that processor,

$$\text{Time/Program} = \text{Time/Cycle} * \text{Cycles/Instruction} * \text{Instructions/Program}$$

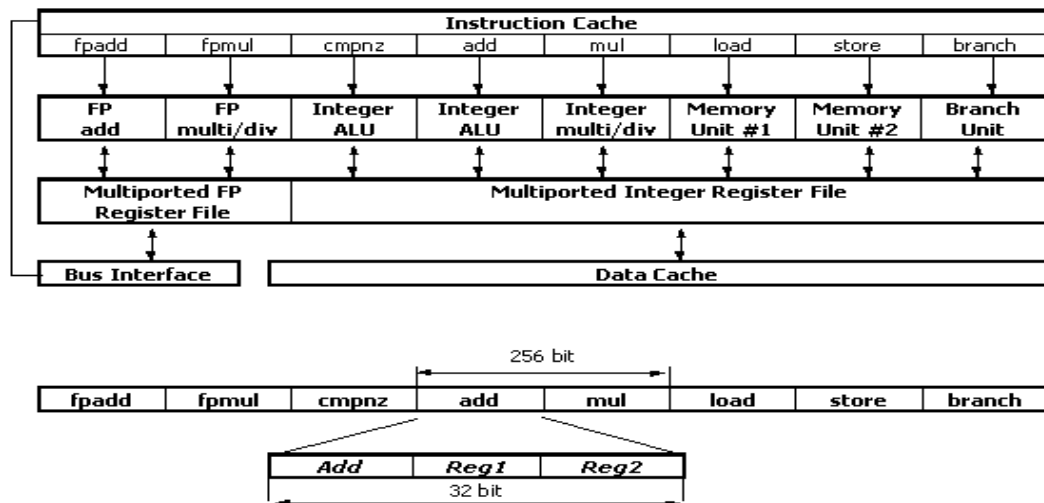
The 4 main processor types are,

CISC – Complex Instruction Set Processor - the CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. In CISC the emphasis is on hardware and includes multi-clock complex instructions. CISC instructions result in small code sizes but lead to a larger hardware footprint. The primary goal of CISC architecture is to complete a task in as few lines of assembly code as possible. This is achieved by building processor hardware that is capable of understanding complex instructions (e.g. multiply or divide) and executing a series of operations to complete complex instructions.

RISC – Reduced Instruction Set Processor – the RISC approach aims at reducing the cycles per instruction at the cost of the number of instructions per program. In RISC the emphasis is on software and includes single-clock, reduced instructions only. The RISC instruction set results in large code sizes but

a smaller hardware footprint. Since RISC processors only use simple instructions that can be executed within one clock cycle, a complex instruction like “multiply or divide” is divided by the software (RISC compiler) or the user writing the assembly language for the RISC processor into several smaller instructions which are then executed by the RISC hardware.

VLIW – Very Long Instruction Word Processor - for the VLIW architecture to deliver the requisite performance as a super-scaling architecture, the VLIW compiler has to be written specifically for that VLIW architecture. The VLIW compiler creates a strictly defined Plan of Execution (POE) during compiling a user program to be executed on that VLIW architecture. The code determines when each operation is to be executed, which functional units in the VLIW architecture are to work and which registers are to contain operands. The compiler delivers the POE (via the architecture of the VLIW instruction set) to the VLIW hardware which implements it and gives back a Record of Execution (ROE), which is a sequence of events which really take place when the program is executed on the hardware. Shown below, a VLIW architecture capable of executing 8 instructions per clock cycle.

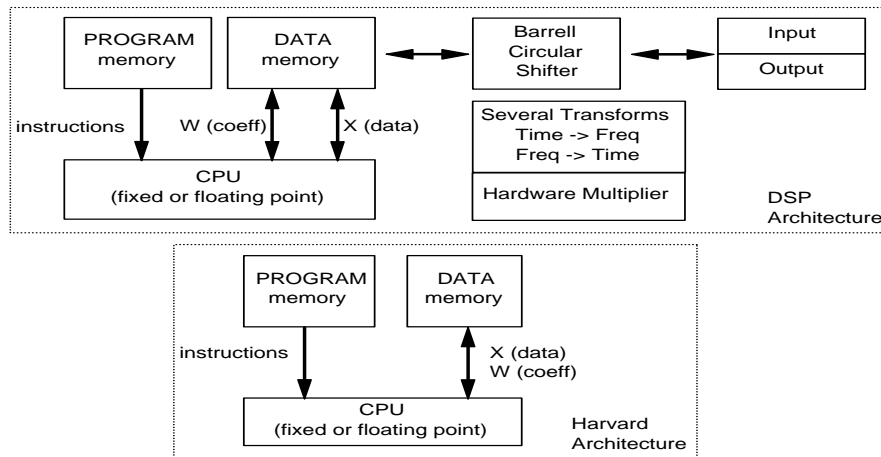


DSP – Digital Signal Processor – this approach aims to implement digital signal processing algorithms as fast as possible. Typically digital signal processing algorithms have 3 main parts,

- circular shift operations for scaling input/output data – barrel shifters
- multiply accumulate operations for IIR/FIR filters – separate x and y data buses for implementing the following equation in 1 cycle,

$$Y(n) = \sum_{n=0}^{N-1} [W_i(n) * X_i(n)] \text{ – where } W_i \text{ and } X_i \text{ are vectors}$$

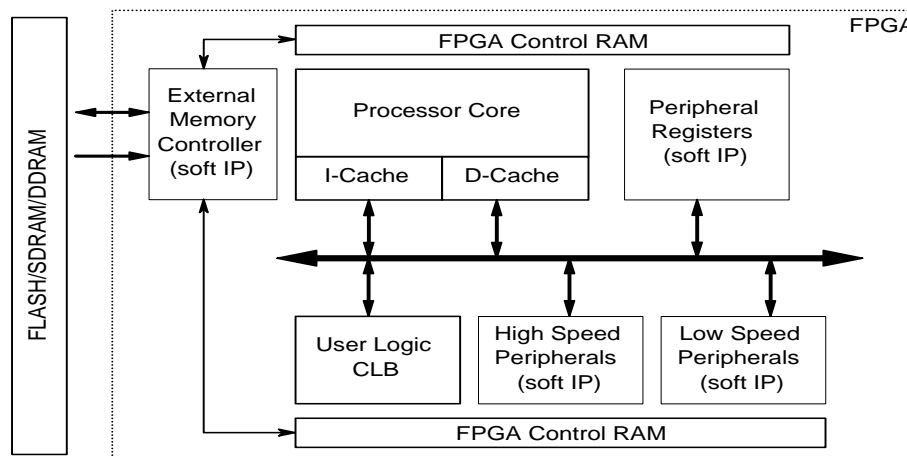
- dedicated hardware blocks for different transforms



Different Hardware Types in Desktop/Embedded Environments

FPGA – Field Programmable Gate Array – these are devices which typically contain combinational logic blocks, memories, external memory interfaces, some 8/16/32 bit fixed or floating point processor core, hardwired multipliers, circular shifters, programmable IO's, and specialized math cells. This entire hardware can be custom programmed multiple (thousands) times in the field (customer's location). Every time the FPGA is programmed a new functional & routing plan is loaded into the chip which modifies its functionality (adding new or removing previous features) and/or its various IOs.

This custom programmability is achieved by allocated a part of the FPGA's internal memory (control RAM) for loading the control program. The control program can be stored into an attached ROM (FLASH), and be loaded into internal control RAM at boot time. The control program describes the functional and routing details in an FPGA. After every boot up stage, the FPGA becomes a standalone embedded device.

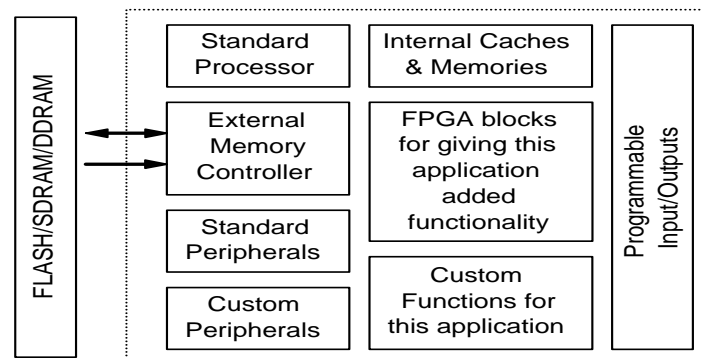


The control program is typically developed on a host (PC or MAC) using the following tools,

- Design entry tool, such as a schematic capture tool for graphically entering the design or a hardware descriptive language (VHDL) tool for describing the design.
- Functional simulator for simulating the functionality of the captured or written design.
- Functional to timing converter which converts the functional blocks into library cells specific to the FPGA vendor.
- Routing generator which adds routing details to the various blocks within an FPGA and also programs the I/O cells.
- Graphical timing simulator for simulating the entire routed design.
- Converting the entire design after simulation into a file (typically HEX or Binary) which is compatible with the control language of the FPGA. This can then be stored into FLASH which can be loaded into the FPGA control RAM at boot time.

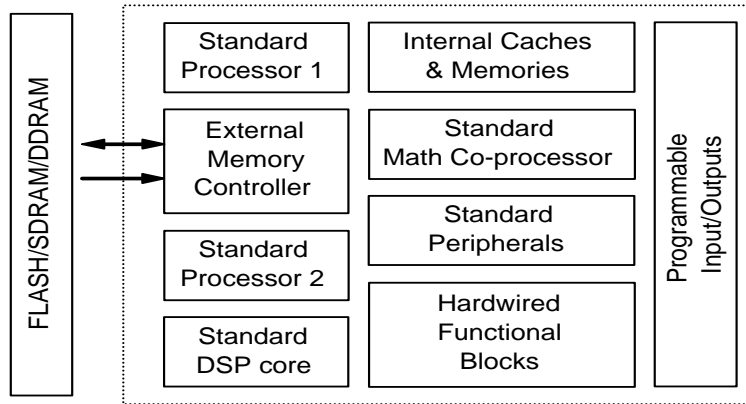
Thus building an FPGA application is very similar to building an SOC or ASIC application, the only difference being this process can be done without involving the expensive fabrication process typical to an ASIC or a SOC. Because of this programmability the FPGA is typically an order of magnitude larger than its ASIC or SOC equivalents. The advantage being, once a design is complete it practically takes no time to use it in an application.

ASIC – Application Specific Integrated Circuit – this type of hardware typically is custom designed for a specific application. It can contain a processor, custom hardware blocks implementing features specific to an application, standard peripherals and also some FPGA blocks which the user can program to give some added functionality to the ASIC at the last minute. An ASIC is typically made for a single customer for his specific application, thus the design and fabrication cost of an ASIC is typically quite high.



SOC – System on a Chip – this type of hardware is typically generic to several applications (e.g. Desktop/Laptop chipsets or SETTOP and DVD chipsets), thus making this chipset extremely complex and expensive from a device integration

(millions of gates), design and fabrication standpoint. Absolutely no change (modification) is possible in the functionality of this chipset once its design and fabrication process is completed. However, since this chipset targets a multitude of high volume applications, its design and fabrication costs are generally averaged out over these applications.



MICROSOFT WINDOWS BASED EDUCATIONAL APPLETS

All the 5 windows applets on the web site "<http://applets.0catch.com/applets.zip>", were built with Microsoft® Visual C/C++ 7.0 (.net) using the Win32 API (no MFC). They are for educational purposes only and should provide some additional information on the subjects listed in this book.

The applications were built using the multi-threading functions found in the Win32 application language interface (API). Each applet is designed using 4 threads (from 1 single windowing task) as follows,

- Thread for reading/buffering a file from the hard disk
- Thread for reading/storing all user inputs via the menu interface
- Thread for processing or decoding the input data samples

www.eyepuzzles.net

Jpeg Applet – this allows a user to de-compress and display a JPEG compressed image OR perform conversions between the most common image formats.

Mpeg Applet – this allows a user to de-compress and display an MPEG1/MPEG2 elementary stream or an MPEG2 program stream. The decode/display supports only MP@ML interlaced/progressive frame resolutions.

Audio Applet – this allows a user to de-compress and display an MPEG1/MPEG2 layer1/layer2 elementary streams without multi-channel extensions.

Imaging Applet – this allows a user to de-compress and display JPEG images and MPEG1/MPEG2 elementary/program streams and also perform several image processing functions in the display window.

PSI Applet – this allows a user to extract and display the PSI tables (PAT, PMT, NIT, SDT and EIT) in a MPEG2 transport stream. Once the tables are displayed the user can extract a MPEG elementary stream (using an Audio/Video PID) and store it on his hard disk.

REFERENCES & WEBSITES

Digitized Signals

Understanding Digital Signal Processing by Richard G. Lyons

<http://electronics.howstuffworks.com/question7.htm>

Digital Signal Processing

Discrete-Time Signal Processing (Prentice Hall Signal Processing Series) by Alan V. Oppenheim, Ronald W. Schaffer

Digital Signal Processing: Principles, Algorithms and Applications by John Proakis, Dimitris Manolakis

Understanding Digital Transmission and Recording by Irwin Lebow

Digital Signal Processing Using MATLAB by Vinay K. Ingle, John G. Proakis

<http://www.calc101.com/>

Digital Filters

Digital Filters by R. W. Hamming

Digital Filters: Basics and Design by Dietrich Schlichtharle

Numerical Methods for Scientists and Engineers by Richard Hamming

<http://ccrma-www.stanford.edu/~jos/filters/>

Adaptive Filters

Adaptive Filter Theory by Simon Haykin

Fundamentals of Statistical Signal Processing (Volume 1 & 2 – Estimation and Detection) by Simon Haykin

<http://www.dspalgorithms.com/>

Frequency Transforms

Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications by Alfred Mertins

http://en.wikipedia.org/wiki/Frequency_transform

Applications in Speech Processing

Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition by Daniel Jurafsky, James H. Martin

Speech Coding Algorithms : Foundation and Evolution of Standardized Coders by Wai C. Chu

Statistical Methods for Speech Recognition (Language, Speech, and Communication) by Frederick Jelinek

<http://www.mathworks.com/products/signal/>

Applications in Image Processing

Fundamentals of Digital Image Processing by Anil K. Jain

Digital Image Processing (2nd Edition) by Rafael C. Gonzalez, Richard E. Woods

Algorithms for Image Processing and Computer Vision by J. R. Parker

<http://www.evepuzzles.net/>

<http://www.evepuzzles.net/applets.zip>

<http://www.mathworks.com/products/imaq/>

<http://www.mathworks.com/products/image/>
<http://www.machinevisiononline.org/>
<http://www.roboticonline.com/>
<http://www.3D-Doctor.com/>
<http://www.intuitivesurgical.com/products/>
<http://www.phenom-world.com/>
<http://www.adastrarocket.com/aarc/Technology>
<http://www.economist.com/node/15543683>
<http://www.visiblebody.com/>
<http://www.fusfoundation.org/MRgFUS-Overview/about-focused-ultrasound-surgery>

Modulation Techniques

Digital Communications by John G. Proakis

Wireless Crash Course by Paul Bedell

The Essential Guide to Wireless Communications Applications, From Cellular Systems to WAP and M-Commerce by Andy Dornan

3G Wireless Demystified by Roman Kitka, Richard Levine, Roman Kikta, Lawrence J. Harte, Romm Kikta

GPRS and 3G Wireless Applications: Professional Developer's Guide by Christoffer Andersson

W-CDMA and cdma2000 for 3G Mobile Networks by M. R. Karim, Mohsen Sarraf, Victor B. Lawrence

Understanding SONET/SDH and ATM: Communications Networks for the Next Millennium by Stamatios V. Kartalopoulos

<http://www.mathworks.com/products/communications/>

Audio Compression Techniques

Digital Video and Audio Compression by Stephen J. Solari

Techniques and Standards for Image, Video, and Audio Coding by K. R. Rao, J.J. Hwang

Introduction to Digital Audio Coding and Standards by Marina Bosi, Richard E. Goldberg, Leonardo Chiariglione

<http://www.mpeg.org/MPEG/audio/>

<http://www.videolan.org/vlc/download-sources.html>

Video Compression Techniques

Mpeg 2 by John Watkinson

Video Compression Demystified by Peter D. Symes

Video Codec Design: Developing Image and Video Compression Systems by Iain Richardson

Video Demystified, 3rd Edition by Keith Jack

The MPEG-4 Book by Touradj Ebrahimi, Fernando Pereira

<http://www.mpeg.org/MPEG/video/>

<http://www.mpeg.tv/>

<http://www.doom9.org/index.html?software.htm>

<http://www.videolan.org/vlc/download-sources.html>

Packetization of Audio, Video, & Data for Delivery over Networks

Data Broadcasting: Understanding the ATSC Data Broadcast Standard by Richard S. Chernock, Regis Crinon, Regis J. Crinon, Michael A. Dolan, Richard Chernock, Regis Crinon

Digital Video Broadcasting : The International Standard for Digital Television by Ulrich Reimers

Packet Video Communications over ATM Networks by Z. S. Bojkovic, Kamisetty Ramamohan Rao

Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers by H. Jonathan Chao, Cheuk H. Lam, Eiji Oki

The Protocols (TCP/IP Illustrated, Volume 1) by W. Richard Stevens

Network Processor Design : Issues and Practices, Volume 1 by Patrick Crowley, Mark A. Franklin, Haldun Hadimioglu, Peter Z. Onufryk

<http://www.mpeg.org/MPEG/mpeg-systems-resources-and-software/>

<http://www.videolan.org/vlc/download-sources.html>

<http://www.hulu.com/movies>
<http://www.hulu.com/documentaries>
<http://edition.cnn.com/video/>

Conditional Access (Security) Systems

Conditional access for digital TV: Opportunities and challenges in Europe and the US by Datamonitor
Satellite and Cable TV: Scrambling and Descrambling by Frank Baylin, Brent Gale

Symmetric and Asymmetric Encryption/Decryption Algorithms

Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition by Bruce Schneier
Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone
Cryptography: Theory and Practice (Discrete Mathematics and Its Applications) by Douglas R. Stinson
The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography) by Joan Daemen, Vincent Rijmen

<http://www.pgpi.org/>

Typical DVB SETTOP Box Application

Digital Television: MPEG-1, MPEG-2 and Principles of the DVB System, Second Edition by Herve Benoit
Digital Video and HDTV Algorithms and Interfaces by Charles Poynton
MPEG Handbook by John Watkinson
Digital Video Broadcasting : The International Standard for Digital Television by Ulrich Reimers

<http://www.tek.com/mpeg-video-test-solutions>

Typical DVD Playback Application

Desktop DVD Authoring by Douglas Dixon
DVD Demystified by Jim Taylor

<http://electronics.howstuffworks.com/dvd1.htm>
<http://www.storagereview.com/guide2000/ref/hdd/ifs/scsi/protPacket.html>
<http://www.doom9.org/index.html?software.htm>
<http://www.divx.com/>
<http://www.videolan.org/vlc/download-sources.html>

Typical DOCSIS® Cable Modem Application

Cable Modems Current Technology by Venketa C. Majeti, John Fijoleck, Michelle Kuska, Kotikalapudi Sriram

<http://www.cablemodem.com/>
<http://www.cablemodem.com/specifications/specifications20.html>

Cable TV SETTOP with Integrated DOCSIS® Cable Modem

<http://www.cablelabs.com/opencable/>
<http://www.cablelabs.com/packetcable/>

Desktop & Embedded Applications

DSP Processor Fundamentals : Architectures and Features by Phil Lapsley, Jeff Bier, Amit Shoham, Edward A. Lee
Microprocessor Architectures RISC, CISC and DSP by Steve Heath
Microprocessor Architectures : From VLIW to TTA by Henk Corporaal

<http://www.xilinx.com/>
<http://www.analog.com/processors/>

<http://www.st.com/>
<http://www.magnumsemi.com/>
<http://www.intel.com/content/www/us/en/homepage.html#>
<http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-4000/hd-4870X2/Pages/hd-4870X2.aspx>
http://www.nvidia.com/object/GeForce_3D_Vision_Main.html
<http://www.broadcom.com/>
<http://www.vanu.com/>
<http://www.mathworks.com/products/xpctarget/>
<http://www.mathworks.com/products/ccslink/>
<http://www.mathworks.com/products/rtw/>
<http://www.metrowerks.com/mw/default.htm>
<http://www.fsmlabs.com/>
<http://www.iniche.com/datasheets.php>
<http://fedoraproject.org/>
<http://www.windriver.com/>
<http://www.anandtech.com/>
<http://www.aaai.org/>

Miscellaneous Websites

<http://ocw.mit.edu/index.html>
http://en.wikipedia.org/wiki/University_of_California,_San_Diego
<http://participatorymedicine.org/>
<http://www.howstuffworks.com/>
<http://www.ted.com/talks/>
<http://www.kurzweilai.net/>
<http://www.burzynskiclinic.com/>
<http://www.gatesfoundation.org/>
<http://consumerguideauto.howstuffworks.com/2012-tata-nano-america.htm>
<http://www.eetimes.com/>
<http://www.edn.com/>
<http://www.embedded.com/>
<http://www.rdmag.com/>
<http://www.scimag.com/>
<http://www.mitre.org/tech/nanotech/index.html>
<http://www.advancedmanufacturing.com/>
<http://www.medicaldevices.org/>
<http://www.classmatepc.com/>
<http://www.cpubenchmark.net/>
<http://www.datacenterknowledge.com/the-top-10-supercomputers-illustrated-nov-2011/>
<http://www.intel.com/content/www/us/en/silicon-innovations/intel-22nm-technology.html>
<http://www.advamed.org/>
<http://www.aami.org/>
<http://www.popsoci.com/popsoci/>
<http://popularmechanics.com/>
<http://www.sciam.com/>
<http://discovermagazine.com/>
<http://www.newscientist.com/>
<http://www.economist.com/science-technology>
http://www.google.com/language_tools?hl=en
<http://maps.google.com/>
<http://www.keyhole.com/keyhole.php>
<http://www.seashepherd.org/campaigns/>
<http://mapsofindia.com/>
<http://www.isro.org/launchvehicles/launchvehicles.aspx>
<http://www.chinatoday.com/>
http://en.wikipedia.org/wiki/Fifth-generation_jet_fighter
<http://www.world-nuclear.org/info/inf34.html>

INDEX

2D FFT, *page 35-36*
2D LMS (TDLMS), *page 60*
3D Audio, *page 113, 205, 210, 219, 221*
3D Imaging, *page 10-11, 13, 131, 246-248*

A

AC3 Audio, *page 93, 107, 158, 210*
AC3 Header, *page 108-109*
Acoustic Echo Cancellation, *page 7, 43, 52, 54-55*
Adaptive Line Enhancer (ALE), *page 29, 55, 59*
ADC, *page 1, 3, 48, 55, 77, 80, 85, 89, 232-233, 243*
Adaptive Digital Pulse Code Modulation (ADPCM), *page 44-45, 47-48, 141*
Adaptive Transversal Filter, *page 23-25, 52, 55, 62*
Adaptive Symmetric Filter, *page 23-25*
Adaptive Asymmetric Filter, *page 23-24*
Adaptive FIR Filter, *page 23-25, 52, 62*
Adaptive IIR Filter, *page 23, 25, 51-52, 62*
Adaptive Lattice Filter, *page 23, 25-26, 45, 47, 52, 55*
Adaptive Sub-band Filter, *page 23, 27, 94-97, 100-101, 103*
Adaptive Frequency Domain Filter, *page 23, 28, 56*
Advanced Encryption Standard (AES), *page 147, 179, 191, 198-202*
AES Key Expansion & Scheduling, *page 198*
AES Encryption, *page 199*
AES Decryption, *page 200*
Aliasing, *page 1-2, 27, 39*
ASIC, *page 256*
Aspect Ratio Conversion, *page 63, 67, 125, 157*
Asymmetric Encryption & Decryption Algorithms, *page 179, 191, 201*
ATM Encapsulation, *page 141-142, 148-149, 163, 172*
ATSC, *page 31, 60-61, 63-64, 82, 87, 107, 109, 119*
ATAPI, *page 144, 205, 213-214, 217, 221, 223-224, 246-247, 249, 251*
Audio Data Interchange Format (ADIF), *page 106*
Audio Data Transport Stream Frame (ADTS), *page 106*
Audio Decoder, *page 94, 99, 110, 113, 179, 205, 210-214, 216, 219-223, 228, 243*
Auto-Correlation, *page 17-20, 33-34, 50*
Auto-Correlation in LPC, *page 45-47*
Automated Voice Response System, *page 48*
Averaging, *page 20, 62, 73, 80*

B

B Frame, *page 121, 127-128, 130, 215*
Baseline Privacy Plus (BPI+), *page 184-189*
Baseline Privacy Key Management (BPKM), *page 174, 185-188, 233-234, 238*
Bi-cubic Interpolation, *page 20-21, 64*
Blitter, *page 11, 205, 210-211, 213, 219-222*
Blackman, *page 15-16*
Blind Equalization Mode, *page 62, 80, 88*
BPI+ Packet Data Encryption, *page 185, 187*
BPI+ Key Management Protocol, *page 185-186, 238*
BPI+ Security Association Identifier (SAID), *page 187-189, 241*
Butterworth, *page 18*

C

Cascaded Filters, *page 15, 18-19, 25*
Cable Modem, *page 3, 82, 167, 169, 172, 175, 179, 184-187, 218, 231-233, 235-244*

Cable TV, *page 82, 151, 169, 243-244*
 Cell Phone, *page 3*
 Central Processing Unit (CPU), *page 58, 205-212, 216-218, 221-224, 228, 231, 233-235, 243-255*
 Chebyshev Filter, *page 18*
 Chirp Z Transform (CZT), *page 6-7*
 CISC, *page 250, 253*
 Clock Recovery, *page 62, 82, 85, 148, 152, 205, 207, 219-221, 223*
 CM and CMTS Interaction, *page 235-238*
 CM Network Layer Software Stacks, *page 235, 239*
 Covariance LPC, *page 45*
 Convolution, *page 5, 11, 16, 40-41, 59-60*
 Contrast Stretching, *page 11, 70, 73*
 Content Scrambling System (CSS), *page 182, 184, 221, 223-224*
 Color Space Conversion (CSC), *page 67*
 Concatenated Coding, *page 75, 90-91, 141, 206*
 Convolutional Encoding, *page 76, 78-84, 88-91*
 Coded Orthogonal Frequency Division Multiplexing (COFDM), *page 3, 75, 87-90, 141*
 Cross-Correlation, *page 15, 17, 20, 33-34*
 CRC, *page 94, 97-99, 106, 109, 112, 143, 149-150, 153, 172-175, 223, 233*

D

DAC, *page 1, 48, 64, 76, 79, 89, 113, 205, 209-210, 219-221, 231-233, 243*
 Data Cache (DCACHE), *page 212, 243, 251*
 Data Directed Mode in Equalization, *page 62, 80*
 Data Encryption Standard (DES), *page 147, 179, 185-188, 191-194, 198, 201-202, 214, 231-234, 238-239, 243*
 DDRAM, *page 205, 209, 212-215, 218-221, 223-224, 231, 243, 246, 249, 251, 256-257*
 De-rotator, *page 77, 80*
 De-modulator, *page 77, 88, 232, 234*
 Decimation, *page 12, 15, 17, 27, 39-41, 47, 63-65, 107*
 Decoding Sub-picture Packs, *page 159, 221-223*
 De-Interlacing (Interlaced to Progressive), *page 57, 63, 66, 68, 128, 236*
 Decoding Time Stamp (DTS), *page 143-144, 157-158, 166, 205, 207-208, 219*
 Desktop Application, *page 178, 245-250, 253, 255-256*
 DHCP, *page 167, 176, 231, 235, 237, 239-241*
 Digitization, *page 1-2*
 Discrete Cosine Transform (DCT), *page 28, 36-38, 93-94, 100-101, 105, 113, 116-123, 126-128, 130, 133-136, 138-139, 209*
 Digital Encoder (DENC), *page 205, 209-211, 219-221, 225, 243*
 Display Engine, *page 67, 127-128, 205-213, 216, 219-221, 225, 243*
 Digital Operator, *page 5, 11-13*
 Digital Signing of Documents, *page 202*
 Digital Video Broadcasting (DVB), *page 76, 79, 87, 142, 145-147, 149-150, 179, 181, 191, 195, 201-202, 205, 214, 243*
 Digital Video Disk (DVD), *page 7, 93, 107, 111, 119-120, 141-142, 152, 155, 157-158, 159-161, 177, 179, 182-184, 205, 208, 213, 215, 217-218, 221-229, 245-247, 249, 256*
 Digital Video Interface (DVI), *page 128, 178, 249*
 DOCSIS Downstream Channel, *page 142, 167, 232*
 DOCSIS Downstream MAC Unit, *page 231, 234, 236, 244*
 DOCSIS MAC Downstream Packetization, *page 142, 167-171*
 DOCSIS MAC Headers, *page 171-174*
 DOCSIS MAC Management Messages, *page 173-174*
 DOCSIS MAC Upstream Packetization, *page 170-171*
 DOCSIS Service Identifier (SID), *page 169-171, 173, 187-189, 241*
 DOCSIS Upstream Channel, *page 142, 169-171, 173, 177, 231-232, 234, 236-238, 240, 243-244*
 DOCSIS Upstream Modulator, *page 232, 234, 238, 244*
 DOCSIS Upstream MAC Unit, *page 169-171, 231-234, 243-244*
 Double Talk Detection (DTD), *page 53-55*
 Downloading Applications in the Field, *page 216*
 DSP, *page 3, 205, 210-212, 219-221, 250-251, 253-255, 257*
 Dual SD Decode, *page 215*
 DVB Encryption/Decryption using Super Scrambling, *page 191, 195-198*
 DVB SETTOP Box, *page 3, 142, 177-180, 205, 207, 210-213, 216-221, 224, 243-244, 247, 252*

DVD Audio Pack, *page 157, 159*
DVD Authoring in a SETTOP environment, *page 205, 215, 217*
DVD Data Search Information (DSI), *page 157-160, 162-163, 177, 183, 223, 226*
DVD Navigation Guide, *page 142, 159, 177, 221-222, 229*
DVD Navigation Pack, *page 157-158, 162, 217*
DVD Playback Model, *page 225*
DVD Playback Paths, *page 227*
DVD Program Chain Structure (PGC), *page 156-157, 160-162, 222, 226-227*
DVD Program Chain Information Structure (PGCI), *page 156, 160-162, 222, 226*
DVD Presentation Control Information (PCI), *page 157-160, 162, 177*
DVD Sub-Stream ID, *page 158-159*
DVD Sub-Picture Pack, *page 159, 221-223*
DVD Video Pack, *page 157-158, 217, 223*
DVD Video Manager (VMG), *page 155-157, 159-161, 177, 222, 226*
DVD Video Manager Information (VMGI), *page 155, 159-160, 177, 222, 226*
DVD Video Title Set (VTS), *page 155-157, 160-162, 177, 222, 226*
DVD Video Title Set Information (VTSI), *page 155-156, 159-161, 177, 222, 226*
DVD Video Object Set (VOBS), *page 155-157, 160-162*
DVD Video Object (VOB), *page 155-158, 161-163*
DVD Video Object Unit (VOBU), *page 157-158, 160-163, 222, 225-226*

E

Electronic Control Messages (ECM), *page 143-144, 147, 179-181, 202, 206, 212*
Electronic Management Messages (EMM), *page 143-144, 146-147, 150, 179-181, 206, 212*
Electronic Program Guide (EPG), *page 145, 179-180, 205-207, 210, 217*
Elliptical, *page 18*
Embossing, *page 11, 20-22*
Embedded Application, *page 68, 245, 252-253*
Embedded Sync, *page 67*
Encryption of Session Keys, *page 191, 201-202*
Endpoint Algorithm, *page 50-51*
End of Active Video (EAV), *page 67*
Equalization, *page 11, 52, 60, 62, 77, 80-81, 83, 88-89*
Error Control Coding (ECC) in DVD, *page 221, 223-224*
Ethernet, *page 141-142, 148, 168, 171-172, 175, 205, 211-212, 217-218, 231, 233-235, 238, 240, 243-246, 249, 252*
Event Information Table (EIT), *page 146, 149-151, 177, 179-180, 217, 243, 259*
External Conditional Access, *page 180*

F

Fast Fourier Transform (FFT), *page 6-8, 28, 32-36, 88-90, 97*
Fast Hartley Transform (FHT), *page 28, 33-36*
Feature Extraction, *page 73*
Finite Impulse Response (FIR), *page 15, 17-19, 22-25, 31, 39, 52, 62, 139, 207, 254*
Fingerprint Recognition, *page 70*
File System, *page 214, 252*
Filter-Bank, *page 95-96, 100, 102, 104-105, 113*
Flash Memory, *page 205, 212, 215-216, 218, 221, 231, 243, 245, 251-252, 255-257*
Format Conversion, *page 63-64, 68*
Forward Error Correction (FEC), *page 61, 75, 83, 86, 89-91, 148, 170, 206, 231-232, 236, 243*
FPGA, *page 255-256*
Frame Sync, *page 61-62, 83-86, 97, 109-110*
Frequency Domain, *page 1, 5-6, 15, 23, 28, 31-32, 56, 88, 104, 107, 110*

G

Galois Field (GF), *page 76, 79, 84, 92, 198-200*
Galois Field Polynomial, *page 92, 198*
Geometric Processes, *page 12*
Ghost Cancellation, *page 60, 62*
Global Positioning Satellite System (GPS), *page 56*
Graphics Engine, *page 205, 208, 210, 212, 217, 219-222, 224, 228*

H

Hanning Filter, *page 7, 15-16*
Handwriting Recognition, *page 72-73*
Hamming Filter, *page 7, 15-16*
High Pass Filter, *page 7, 11, 20-22, 39-37, 59, 96*
High Definition Multi-Media Interface (HDMI), *page 178*
Hilbert Transform, *page 31, 33, 35, 86-87*
Huffman Encoding, *page 58-59, 96-97, 99-102, 104, 115-118, 120-121, 123, 140*

I

I Frame, *page 121, 128, 214-215*
IEEE 802.3 (Ethernet) Packetization, *page 142, 175, 218, 233, 235, 243*
IIR Filter, *page 15, 18-19, 23, 25, 52, 62, 254*
Image Coding, *page 36-37, 57*
Image Enhancement, *page 57, 59, 70*
Image Processing, *page 10-12, 37, 57, 68, 245-250, 259*
Impulse Response, *page 15, 24, 31, 39, 54, 56, 59, 78, 82*
Inner Coding, *page 75-76, 79-81, 90-91*
Instruction Cache (ICACHE), *page 205, 221, 231, 243, 251*
Interleaving, *page 76-82, 84-85, 87, 89-92*
Internal Conditional Access, *page 179*
Internet Protocol (IP), *page 142, 168, 175, 241*
Interlaced, *page 10-11, 64-66, 68, 119, 121, 123-124, 128, 130, 140, 207, 211, 221, 225, 259*
Interpolation, *page 11-12, 15, 17, 20-21, 27, 47, 63-66, 68, 77, 120-121, 128, 139, 209, 225*
Inter Symbol Interference (ISI), *page 77, 86, 206, 232*

J

JPEG Encoding/Decoding, *page 36, 58, 115-118, 209-210, 213, 218, 259*

K

Kaiser, *page 15-16*

L

Least Mean Square (LMS), *page 23, 29-30, 52, 54-56, 59-60, 63*
Line Doubling, *page 63, 66*
Line Echo Cancellation, *page 7, 43, 52-54*
Linear PCM, *page 93, 101, 113, 157-159, 221-222*
Linear Predictive Coding (LPC), *page 45-46, 113*
Logical Structure of a DVD, *page 155*
Low Frequency Channel (Subwoofer), *page 93-94, 98-99, 106, 109, 210*
Low Pass Filter (LPF), *page 3, 27, 38, 40-41, 47, 49, 52, 77, 85*

M

Maximum Eye Opening, *page 62, 75, 78, 80-81, 206*
Matlab for Embedded Applications, *page 246-250, 253*
Median Filter, *page 20, 59*
Modulation, *page 1, 3, 9, 31, 44, 75-80, 82, 84-85, 87-89, 92, 141, 168, 170, 232-233*
Motion Detection, *page 11, 66, 128, 209*
MPEG Audio Layer 1, *page 93-100, 205, 219-221*
MPEG Audio Layer 2, *page 93-100, 205, 219-221*
MPEG Audio Layer 3, *page 93-101, 205, 219-221*
MPEG Sync, *page 79-82, 147-148*
MPEG Video Sequence, *page 119-120, 122, 124-127, 129, 131-132, 134, 137, 207-208*
MPEG Video GOP, *page 120-121, 124-126, 128, 132, 158, 208, 214-215*
MPEG Video Picture, *page 118-127, 132, 135, 139-140, 144, 158, 207-209, 212-213, 215-216, 222, 225*
MPEG Video Slice, *page 123-124, 126, 130, 208-209, 219*
MPEG Video Macro-block, *page 68, 115, 118-124, 126, 128-130, 133-135, 139, 209*
MPEG Video Block, *page 121, 123, 126, 128, 130, 133-135, 139, 208-209, 219*
MPEG1 Audio, *page 93-94, 97-98, 153*

MPEG1 Video, *page 120, 153*
MPEG2 Audio, *page 93, 98, 153, 168*
MPEG2 Audio Layer 4, *page 93, 102, 106*
MPEG2 Packetization, *page 142, 150*
MPEG2 Packetized Elementary Stream (PES), *page 142-145, 147-148, 152-154, 157-158, 163, 179-181, 191, 195-196, 201*
MPEG2 Program Streams, *page 142, 144, 152, 229, 259*
MPEG2 Transport Streams, *page 142, 145-146, 148, 151, 166, 205-206, 215-216*
MPEG2 Video, *page 119-120, 122, 124, 126-127, 153, 168*
MPEG4 Access Unit, *page 142, 163*
MPEG4 Delivery Multi-Media Integration Framework (DMIF), *page 164-165*
MPEG4 Error Resiliency, *page 119, 129, 136-137*
MPEG4 FlexMux Layer, *page 142, 163-165*
MPEG4 TransMux Layer, *page 142, 163-165*
MPEG4 Video, *page 129-134, 136*
MPEG4 Video Object Plane (VOP), *page 129-139*
MPEG4 Group of Video Object Planes (GOV), *page 132, 139*
MPEG4 Video Object Layer (VOL), *page 131-134, 136-137, 139*
MPEG4 Video Object (VO), *page 131-133*
MPEG4 Video Object Sequence (VS), *page 132, 139*
Multi-Channel (MC) Audio Extensions, *page 47, 93-94, 97-99, 102, 107, 210, 221, 259*

N

Navigation Guide, *page 142, 159, 177, 221-222, 229*
NTSC, *page 64, 82, 84-86, 128, 205, 210-211, 219-221, 243*
NTSC Rejection Filter, *page 86*

O

Optical Character Recognition (OCR), *page 68*
Outer Coding, *page 76, 90-92*

P

P Frame, *page 121, 128, 215*
Pack Header, *page 143-144, 152-153, 158-159, 183, 217, 223*
PAL, *page 64, 128, 205, 210-211, 219-221, 243*
Pattern Recognition, *page 48, 50-51, 57, 68-69*
PC as a SETTOP Box, *page 218*
Peripheral Component Interface (PCI) Express, *page 178*
Personal Video Recorder (PVR), *page 205, 208, 212-213, 215*
Pitch Period, *page 46-47*
Predictor Coefficients, *page 45-47*
Presentation Time Stamp (PTS), *page 143-145, 153-154, 166, 205, 207-208, 210-211, 214, 219-222, 228*
Program Association Section (PAT), *page 146-147, 150, 177, 179-180, 206, 216-217, 243, 259*
Program Clock Reference (PCR), *page 146-149, 166, 207, 213-214*
Program Map Section (PMT), *page 146, 150-151, 177, 179-180, 206, 216-217, 243, 259*
Program Specific Information (PSI), *page 146-150, 169, 243, 259*
Program Stream MAP, *page 143, 153-154*
Progressive, *page 10-11, 64-66, 68, 118-119, 123-126, 128-129, 140, 207, 211, 221, 225, 259*
Psuedo-Coloring, *page 11*
Public Key Encryption using the RSA Algorithm, *page 179, 186-187, 191, 201-203, 238*

Q

Quadrature Amplitude Modulation (QAM), *page 3, 62-63, 75, 78-82, 87-89, 141, 167-168, 205, 219, 231-233, 236, 243-244*
Quadrature Mirror Filters (QMF), *page 27, 38-42*
Quadrature Phase Shift Keying (QPSK), *page 3, 75-78, 81-82, 90, 141, 167, 205, 219, 231-233, 243-244*
Quantization, *page 44, 93, 95, 101-103, 107, 110, 115-116, 118, 128, 138*

R

Recursive Least Square (RLS), *page 23, 30*

Reed Solomon Encoding/Decoding, *page 76, 78-81, 84, 87, 89, 92, 206, 223-224, 232*
RGB, *page 10, 22, 35, 58, 67-68, 115, 117, 128, 205, 210, 219-221, 225, 243-244*
RISC, *page 250, 253-254*
Root Raised Cosine, *page 76-77, 79, 84*
Run Length Encoding (RLE), *page 57-58, 117, 121-123, 130, 140-141, 229*
Running Status Table (RST), *page 146, 152*

S

Scaling, *page 12, 39, 41, 63-64, 66, 104, 137, 139, 154, 205, 209, 219-222, 225, 254*
SDRAM memory, *page 68, 205-206, 208-209, 212-215, 218-221, 223-224, 231, 243, 246, 249, 251, 255-257*
Security in DOCSIS, *page 168, 175-176, 184, 187-188*
Sending Secure Messages, *page 185-187, 202*
Service Description Table (SDT), *page 146, 151, 177, 179-180, 206, 217, 243, 259*
Service Delivery Model in DVB, *page 145*
SIFT, *page 47*
Silicon Tuner, *page 205, 219, 231-234, 243*
Simulcast Transmission, *page 84*
Skeletonization, *page 70-71*
Small Networks Management Protocol (SNMP), *page 167, 231, 235, 239-240*
Sobel Edge Detection, *page 11, 20*
Sound Card, *page 3, 245-247, 249, 259*
Spatial, *page 11, 36, 119-121, 124, 129-130, 136-137*
Spatial Filters, *page 11*
Spatial Scalability, *page 120, 124, 130, 136-137*
SPDIF Format, *page 113, 141, 207, 211, 221-222, 243*
Speaker Dependant Speech Recognition, *page 43, 51*
Speaker Identification System, *page 43, 48-49*
Speaker Independent Speech Recognition, *page 43, 49*
Spectrogram, *page 7-9*
Speech Coding, *page 44, 47*
Speech Enhancement, *page 43, 51-52*
Speech Processing, *page 7, 43, 48*
Speech Recognition, *page 43, 49-51*
Speech Synthesis, *page 43*
Start of Active Video (SAV), *page 67, 112*
Stuffing Table (ST), *page 146, 152*
Surface Acoustic Wave (SAW) filter, *page 77, 80*
Symbol Averaging, *page 80*
Symmetric Encryption/Decryption, *page 147, 179, 191, 198, 201*
System on a Chip (SOC), *page 256*
System Time Clock (STC), *page 144-146, 152, 205, 207-208, 210-211, 214, 219-221, 223, 228*

T

T Spaced Equalizer, *page 62*
TCP/IP, *page 141, 176, 235, 239, 252*
Temporal Filtering, *page 102, 104-105, 120-121, 124, 126, 129-130, 135-137, 215*
Terrestrial, *page 1, 31, 43, 61, 75, 82-85, 87-90, 111, 120, 141-142, 145-146, 177, 201, 205-206, 212, 215, 217, 219, 246, 249*
Time Domain, *page 5-6, 19, 28, 31-33, 38, 47, 50, 58, 95, 104-105, 110, 112-113, 128*
Time Offset Table (TOT), *page 146, 151*
Time of Day (TOD), *page 167, 231, 235, 237, 239-240*
Timing, *page 84-85, 111, 128, 146, 152, 163, 165-166, 169-170, 172-174, 205, 211, 219, 231, 234, 236-237, 256*
Training Mode in Equalizer, *page 52, 62, 72, 83*
Trellis, *page 79-85, 87, 90-91, 232-233*
Triple DES, *page 147, 179, 186, 194, 238-239*
Track Buffer in DVD, *page 221-225, 228*
Transport De-multiplexer, *page 146-148, 150, 205-207, 211-214, 216-217, 219, 243*
Transport Header, *page 148, 206, 214, 219*
Transport Stream Output, *page 205, 216, 219*

U

Universal Datagram Protocol (UDP), *page* 141, 163, 167, 176, 231, 235, 237, 239

USB Packetization, *page* 141-142, 144, 176-177, 205, 211-214, 217-218, 221, 223-224, 231, 235, 244-247, 249, 251

V

VBV Buffer, *page* 119, 127

Vestigial Sideband Modulation (VSB), *page* 3, 61-63, 75, 82-87, 141, 205, 219

VLIW, *page* 254

Voiced & Unvoiced Decisions, *page* 46-47, 50-51

Volumetric Process, *page* 13, 248

Voxel, *page* 10-11, 13

W

Walsh Transform, *page* 68-69, 73

Wavelet Transform, *page* 38, 40-41, 58, 140

Y

YUV pixels, *page* 10, 22, 35, 58, 64, 67-68, 115, 117, 128-129, 132, 142, 144, 205, 207, 210-211, 219-222, 225, 243-244

Z

Z Transform, *page* 6-7

Zero Crossing Rate, *page* 50, 86

Zig-Zag Scan, *page* 116-118, 138-139

About the Author

Manish Pradhan has twelve years of experience developing hardware & software in the areas of digital audio-video signal processing, multi-media and pattern recognition using desktop/embedded systems. He holds a Master of Science degree in Electrical Engineering with specialization in Digital Signal Processing.

<http://www.mathworks.com/support/books/book48498.html?category=11&language=1&view=category>

http://www.jaicobooks.com/i/j_searchtry.asp?selcat=title&keyword=Digital+Signals