ECE 738 – Advanced Digital Image Processing

Final Project Report:

MPEG2 Fine Granularity Scalability Enhancement Implementation

Deshan Yang
Department of Electrical & Computer Engineering
University of Wisconsin

dyang@cae.wisc.edu or dyangchicago@yahoo.com
# Table of Contents

1. Abstract .......................................................................................................................... 3
2. Introduction ...................................................................................................................... 3
   2.1 FGS .......................................................................................................................... 3
   2.2 Bitplane coding of DCT coefficients ........................................................................ 3
       2.2.1 Example of entropy coding: ........................................................................... 3
       2.2.2 Bitplane coding ............................................................................................... 3
       2.2.3 Example of bitplane coding ............................................................................ 3
   2.3 Overall of the project ............................................................................................... 3
   2.4 Scope and limitation of the project .......................................................................... 3
3. System design .................................................................................................................. 3
   3.1 Design of FGS Enabled MPEG2 Encoder ............................................................... 3
   3.2 Design of FGS enabled MPEG2 decoder ............................................................... 3
   3.3 Design of bitplane codec ......................................................................................... 3
   3.4 Design of FGS enabled MPEG2 player ................................................................... 3
4. Implementation details ..................................................................................................... 3
   4.1 Generate original video stream data ....................................................................... 3
   4.2 Testing the whole application .................................................................................. 3
   4.3 Decoder: How to merge enhancement layer and base layer .................................. 3
5. Result and conclusion: .................................................................................................... 3
6. Appendix .......................................................................................................................... 3
   6.1 About the source code .............................................................................................. 3
   6.2 Command options and examples .............................................................................. 3
       6.2.1 Encoder .......................................................................................................... 3
       6.2.2 Decoder .......................................................................................................... 3
       6.2.3 Player ............................................................................................................. 3
   6.3 Further information .................................................................................................... 3
7. References ......................................................................................................................... 3
1 Abstract

The project implements Fine Granularity Scalability (FGS) enhancement algorithm to the current MPEG2 encoder and decoder from MPEG Software Simulation Group (MSSG). It updates the current MPEG2 codec source code to support FGS enhancement. It implements new bitplane VL codec and streaming. It also implements a new FGS enabled MPEG2 player for Windows platform.

2 Introduction

2.1 FGS

FGS is an enhancement to standard MPEG2 specification. It likes other layered scalability enhancement techniques, such as SNR scalability, temporal scalability and spatial scalability. It encodes MPEG2 video into two layers: the base layer and the enhancement layer. Unlike other layered scalability enhancement techniques, it encodes the enhancement layer by using progressive bitplane coding. FGS enhancement layer bitstream can be truncated at any points that the decoder would like to. The effectiveness of enhancement is determined by the amount of enhancement layer data used. The more enhancement data are used, the better quality the decoded video stream has.

Figure 1: Illustration of video coding performance (Li, W., 2001)

Most other techniques work in such stepwise ways that either the full enhancement layer is used or there is no enhancement. FGS enhancement algorithm works in a progressive way. The way of FGS works is better matching the way an end user expects that the perceptual quality of the decoded video stream is getting better and better as more and more data are received and used for decoding. FGS technique is very suitable for applications such as video streaming over bandwidth-limited media.
Of course FGS technique has its own disadvantages. Its coding efficiency is lower than single layer – fixed bit rate coding method. It is difficult to apply motion prediction to the FGS enhancement layer because the enhancement layer must be ready to be truncated at any point. Motion drifting occurs if motion prediction is applied at enhancement layer. If motion prediction is not applied, the enhancement layer than has further low coding efficiency.

2.2 Bitplane coding of DCT coefficients

Entropy coding techniques are used to quantize and encode DCT coefficients in most current video coding standards, such as MPEG-1, MPEG-2, MPEG-4, H.261/263 and JPEG. A problem with such entropy coding and quantization techniques is that the statistics of the run value symbols are highly dependent on the quantization step size and the coding efficiency for all possible quantization step sized and all possible dynamic ranges of the DCT coefficients. Therefore, a single fixed entropy coding table cannot optimize coding efficiency for all possible quantization step sizes and all possible dynamic ranges of DCT coefficients (Ling, et al. 1999). Bitplane coding technique is a new coding scheme to overcome this problem. In fact, it provides a better performance than run value coding under all conditions.

2.2.1 Example of entropy coding:

DCT coefficients of a 8_by_8 DCT block:

0  -14  0  0  -3  0  0  0
-8  -2  -2  0  0  0  0  0
-7   8   3  0  0  0  0  0
-5   0   0  0  0  0  0  0
 0   3  0  0  0  0  0  0
 0   3  0  0  0  0  0  0
 0   0  0  0  0  0  0  0
 0   0  0  0  0  0  0  0

Zigzag ordering:
Values: 0  14  8  7  2  0  0  2  8  5  0  3  0  3  0  0  0  3  0  3  0  . . .
Signs:  x  1  1  1  1  x  x  1  0  1  x  x  0  x  x  x  x  0  x  x  0  x  x  . . .

Run value symbols:

3-D:  (1,14,0), (0,8,0), (0,7,0), (0,2,0), (2,2,0), (0,8,0), (0,5,0), (2,3,0)
     (1,3,0), (4,3,0), (2,3,1)
2-D:  (1,14), (0,8), (0,7), (0,2), (2,2), (0,8), (0,5), (2,3), (1,3),
     (4,3), (2,3), EOB

Coding the run value symbols and signs:

3-D:  VLC(1,14,0), 1, VLC(0,8,0), 1, VLC(0,7,0), 1, VLC(0,2,0), 1,
     VLC(2,2,0), 1, VLC(0,8,0), 0, VLC(0,5,0), 1, VLC(2,3,0), 0,
     VLC(1,3,0), 1, VLC(4,3,0), 0, VLC(2,3,1), 0
2-D:  VLC(1,14), 1, VLC(0,8), 1, VLC(0,7), 1, VLC(0,2), 1,
     VLC(2,2), 1, VLC(0,8), 0, VLC(0,5), 1, VLC(2,3), 0,
     VLC(1,3), 1, VLC(4,3), 0, VLC(2,3), 0, VLC(EOB)
2.2.2 Bitplane coding

Bitplane coding is done in the following steps (Ling, et al. 1999):
1. Zigzag order the 2-D quantized DCT array into a 1-D array
2. Take absolute values and signs of the quantized DCT coefficients
3. Find the largest absolute value and the minimum number of bits, N, needed to represent it in the binary format. N is the number of bitplanes.
4. Represent every one of the quantized DCT coefficients with N bits in the binary format and form N bit planes
5. For each bitplane, 3-D symbols are formed of three components:
   a. Number of consecutive 0’s before a 1 (run),
   b. Where there are any 1’s left on this bit plane after the current 1, i.e. End-Of-Plane (EOP),
   c. Whether there are any more bitplanes left in this DCT block, i.e. End-Of-Block (EOB).

There are only 3 (instead of 4) possible combinations for the last two components, namely, (Not EOP, Not EOB), (EOP, Not EOB) and (EOP, EOB), are formed to represent an all-zero bit plane followed or not followed by other bit planes, respectively. Note that the MSB plane does not have the all-zero case.
6. The sign bits of all non-zero coefficients are put at the end of the codes for coding the bit planes.

2.2.3 Example of bitplane coding

1. Zigzag ordering:
   Values: 0 14 8 7 2 0 0 2 8 5 0 0 3 0 3 0 0 0 3 0 0 3 0 . . . .
   Signs:  x 1 1 1 1 x x 1 0 1 x x 0 x 1 x x x 0 x x 0 x x . . . .
2. Find the largest absolute value and the number of bitplanes: 14 and N=4
3. Form 4 bitplanes
   MSB: 0110000010000000000000000000000000000000000000000000000000000000
   MSB1: 0101000010000000000000000000000000000000000000000000000000000000
   MSB2: 0101000010000000000000000000000000000000000000000000000000000000
   MSB3: 0001000010000000000000000000000000000000000000000000000000000000
4. Form 3-D symbols (run, EOP, EOB)
   MSB: (1,0,0), (0,0,0), (5,1,0)
   MSB1: (1,0,0), (1,0,0), (5,1,0)
   MSB2: (1,0,0), (1,0,0), (0,0,0), (2,0,0), (4,0,0), (1,0,0), (4,0,0), (2,1,0)
   MSB3: (3,0,0), (5,0,0), (2,0,0), (1,0,0), (4,0,0), (2,1,1)
5. Code the 3-D symbols and signs
   VLC(1,0,0), VLC(0,0,0), VLC(5,1,0), VLC(1,0,0), VLC(1,0,0), VLC(5,1,0),
   VLC(1,0,0), VLC(1,0,0), VLC(0,0,0), VLC(2,0,0), VLC(4,0,0), VLC(1,0,0),
   VLC(4,0,0), VLC(2,1,0), VLC(3,0,0), VLC(5,0,0), VLC(2,0,0), VLC(1,0,0),
   VLC(4,0,0), VLC(2,1,1), 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0

It is easy to see that output VLC codes of bitplane coding are ordered as their significance because of zigzag ordering and bitplane ordering. The most important codes are ordered in the front. Remaining codes are less important. This property is very important for FGS algorithm in which end users are willing to truncate the bit stream in the middle.
Another observation of bit planes is that the statistics properties of different bit planes are different. Different VLC code tables can be used for different bit planes. MSB, MSB1 or MSB2 planes should have its own VLC code tables. The remaining planes can share a common VLC code table.

Study shows that bitplane coding has over 20% better coding efficiency over run value coding under the same quantization. This is the reason why bitplane coding is used for JPEG2000.

Somehow, performance of bitplane coding is worse than regular MPEG run value coding. Software decoder would run slower with biplane decoding.

2.3 Overall of the project

Figure 2: Diagram of the whole FGS enabled MPEG2 encoder and decoder system.

Figure 2 shows system function blocks. Both encoder and decoder are implemented in this project. Encoded MPEG2 FGS bit stream will be transmitted via a media channel. The MPEG2 FGS decoder decodes the received bit stream by using base layer and the FGS enhancement layer. Video output from decoder will be evaluated by comparing to the original video input. PSNR, MSE and RMSE will be computed for SNR and performance analysis purpose.

Three major tasks are covered by this project:
- Updating the current MPEG2 codec source code to support FGS
- Implementing bitplane VL Codec and bit streaming
- Implementing a new FGS enabled MPEG2 player

I am using MPEG2 codec and player source code from MPEG Software Simulation Group (MSSG) at http://www.mpeg.org/MPEG/MSSG/. My MPEG2 FGS player is combined the current MPEG2 player with my new MPEG2 FGS decoder.

There are many challenges for the project:
- Understanding and using the current MSSG MPEG2 codec source code (much more difficult than it seems to be).
• Implementing a new bitplane codec
• Integrating the new bitplane codec into the current MPEG2 codec framework for FGS enhancement.
• Creating a new MPEG2 FGS player on Windows platform. The current MPEG2 player from MSSG is not very useful because its architecture is far different from the MPEG2 decoder. I have to create almost a new player with FGS supporting to play my FGS enhanced MPEG2 video stream.
• Lack of media data to play with. The current MPEG2 codec only works with certain type of source video data input. I have to find or create requirement video source data in order to run the encoder and decoder.
• Evaluation the FGS decoding results.

Regardless of all the challenges, I managed to finish the project in limited time. Encoder, decoder and player are all working. My result is promising.

2.4 Scope and limitation of the project

This project has not fully implemented FGS for MPEG2 because of the scope and time limitation. Basic FGS functionalities are finished. Current implementation only supports I-frame and B-frame. B-frame must not be used in order to have FGS working. (This could be fixed as time allows, but not in this project.) Basic bitplane encoder and decoder are implemented. The bitplane VL codec does not have good Huffman VLC tables to optimize the output bitplane coding bit stream. Again, it is easy to fix as far as there is good Huffman VLC tables to be used. Different bitplanes usually have different statistical properties, especially for bitplane MSB, MSB1 and MSB2. So we need to have multiple Huffman VLC tables for different bitplanes. (This won’t be covered by this project.)

Encoder and decoder performance is another issue that is not addressed by current implementation. Original MSSG source code was written in C language. Speed of original codec was not satisfied. After I added FGS enhancement and bitplane codec functions, execution speed gets more slow. Since the goal of this project is to demonstrate the purpose of FGS enhancement technique, performance is not a big concerning.

3 System design:

3.1 Design of FGS Enabled MPEG2 Encoder

The most important difference from the FGS enabled MPEG2 encoder to regular MPEG2 encoder is that DCT coefficient residues (DCT coefficient differences before and after quantization) are computed and sent to the bitplane encoder to generate enhancement layer bit stream.

Important issues:
• New user options to control and configure FGS encoding.
• If FGS is enabled, DCT coefficient residues are computed as the difference of DCT coefficients before quantization and DCT coefficients after quantization.
(between point 1 and point 2 in Fig. 3) for I-frame and P-frame, but not B-frame.
- DCT coefficient residues will be encoded by the bitplane encoder.
- There is almost no change with the base layer encoder form standard MPEG2 encoder. Standard MPEG2 quantization matrix is still used.
- Interface between the base layer encoder to the FGS bitplane encoder
- Only base layer DCT coefficients are used in the motion prediction loop. There is no motion prediction in the enhancement layer.
- As shown in Fig. 3, only 1 motion prediction loop is used in the encoder, not double loops. There is no motion-drifting problem because motion prediction is only done in the base layer. The enhancement layer is not used in the motion prediction loop.

![Figure 3: Block diagram of the new FGS enabled MPEG2 encoder](image.png)

**3.2 Design of FGS enabled MPEG2 decoder**

See the decoder block diagram in Figure 4. The decoder should be easier to implement then the encoder. Just like the encoder, the most important difference between the new FGS enabled decoder and the original MPEG decoder is that DCT coefficient residues from bitplane decoder are merged with base layer DCT coefficients to generate the enhanced video output.

*Important issues with the MPEG2 FGS decoder:*
- New user options to control enhancement layer FGS decoder
- Interface between the current MPEG2 decoder to the FGS bitplane decoder.
- There is no big difference from the base layer decoder to current MPEG2 decoder
- Enhancement layer DCT coefficient residue decoding.
• Merging the base layer DCT coefficients with enhancement layer DCT coefficient residues
• Output file format.
• It is possible that none, part or whole enhancement layer DCT coefficient residues are used with base layer DCT coefficients to decode MPEG2 stream, depending on the FGS scale factor selected by end users.

![Block diagram of the new FGS enabled MPEG2 decoder](image)

**Figure 4: Block diagram of the new FGS enabled MPEG2 decoder**

### 3.3 Design of bitplane codec

The bitplane encoder and decoder are implemented by following the steps in section 2.2. Since the final MPEG2 FGS encoder and MPEG2 FGS decoder are separated execution files, the bitplane encoder is embedded in the final MPEG2 FGS encoder and the bitplane decoder is embedded in the final MPEG2 FGS decoder.

**Important issues with both bitplane encoder and decoder:**
- Enhancement layer bit stream binary file format and data structure
- Frame bit stream data structure
- Block bit stream data structure
- VLC codes tables
- Zigzag ordering

**Important issues with the bitplane encoder:**
- Interface with base layer MPEG2 encoder
- Sign bits extraction
- Bitplane splitting
- Run value code generating
- Converting run value codes to variable length codes by using appropriate VLC code tables (This is not implemented because VLC code tables are not available)
- Storing encoder VLC codes in binary files as the enhancement layer bit stream output
Important issue with the bitplane decoder:
- Interface with base layer MPEG2 decoder
- Sign bits combination
- Bitplane merging
- Run value code decoding
- Converting variable length codes to run value codes by using appropriate VLC code tables (This is not implemented because VLC code tables are not available)
- Accessing VLC codes from binary files as the enhancement layer bit stream input

3.4 Design of FGS enabled MPEG2 player

The new FGS enabled MPEG2 player is a combination of the current MPEG2 player and my new FGS enabled MPEG2 decoder. Since there are big changes in the decoder and current MPEG2 player is based on an even older version decoder, existing decoder embedded in the current player is thrown out, only its GUI (Graphics User Interface) part is used for new player.

Important issues of the player:
- Decoding results from new FGS decoder are in RGB binary image format.
- RGB binary images of frames are rendered in the application GUI window.
- New menu items are added to control FGS decoding
- The new player works at Windows platform only
- New user options and command line options for FGS
- Other features are added: Replay, Stop, Stretching to window, Supporting wide screen format, FGS scale change on the fly, Turning FGS on/off, Player window resizing

4 Implementation details

4.1 Generate original video stream data

Since there is no original video stream data for me to test with, I have to generate my own original video stream data from some of my DVD movies. I ripped one of my DVD movies, decoded the DVD movie with a DVD software player, which save its video output to raw PPM image files for each frame. I saved 450 frames, 15 seconds of 30 frames per second.

4.2 Testing the whole application

The encoder, decoder and player can be tested in the following steps:
1. Use the encoder to encode original raw video data to MPEG2 base layer and FGS enhancement layer.
   Base layer is saved as a M2V file, which is one of the standard file formats of MPEG2. The M2V file itself can be played with any MPEG2 player.
   Enhancement layer is saved as multiple binary files. There is one file for each
frame. Since the binary files are in special format, only my MPEG2 FGS decoder and my new FGS enabled MPEG2 player are able to use these binary files.

2. Use the MPEG2 FGS decoder to decode the base layer M2V file with enhancement layer binary files.
   The decoder is a console command line execution file. It accepts command line options. See the appendices for examples of using the decoder from DOS command line.
   Decoder will generate PPM image files for each frame. These generated PPM image files can be compared to the original PPM image files for performance and SNR evaluation.

3. Use the FGS enabled MPEG2 player to play the FGS encoded video stream
   See the appendices for examples about how to use the player and its command line options.

4.3 Decoder: How to merge enhancement layer and base layer

   Since DCT and IDCT transformation are linear operations, DCT coefficients from enhancement layer can be added to the DCT coefficients from base layer to form the enhanced DCT coefficients. IDCT transforms the enhanced DCT coefficients to enhanced video output.

   We still need to IDCT the base layer DCT coefficients in order to get the base layer video output. Base layer video output is used in the decoder motion compensation loop. Enhanced video output must not be used to replace the base layer video output in the motion compensation loop, otherwise motion-drifting problem will occur.

5 Result and conclusion:

   FGS encoded video stream is decoded at different FGS scale factors, from 0% to 100%. FGS scale factor at 0% means FGS enhancement layer data are not used. FGS scale factor at 100% means all FGS enhancement layer data are used. FGS scale factors between 0% and 100% control the percentage of the length of FGS enhancement layer bit stream used in decoding. The higher percentage of FGS enhancement layer data are used, the better video decoding result is.

   The decoded video frames are stored into BMP image files per frames.

Figure 5: FGS scale factor = 0 (FGS)  
Figure 6: FGS scale factor = 15%
enhancement is not used)

![Figure 7: FGS Scale factor = 30%](image1)

![Figure 8: FGS Scale factor = 65%](image2)

![Figure 9: FGS Scale factor = 100% (Fully enhanced)](image3)

Table 1, 2, 3 show the FGS decoding results. Decoded frame images are compared to corresponding original frame images that were used to encode the video stream. PSNR, MSE and RMSE are computed for each FGS scale factors and are averaged across frames. Only a few FGS scale factors are used in the comparison. The result is promising. It shows that the decoding result is improved along with more FGS enhancement data are used. The higher the FGS scale factor is, the better the result is.

As we know, PSNR, MSE and RMSE do not usually represent human perceptual quality of video streams or images, neither FGS scale factors are calculated in the scale of human perceptual image quality. FGS scale factors are just the percentage of the length of the enhancement layer data used in decoding. The relationship between the FGS factors and the computed image factors are not linear proportional. As I observed, the FGS factors are actually more proportional to human perceptual scale instead of to PSNR, MSE and RMSE.
I used imgcmp.exe from Jasper project to compute PSNR, MSE and RMSE. Jasper ([http://www.ece.uvic.ca/~mdadams/jasper/](http://www.ece.uvic.ca/~mdadams/jasper/)) is an open source JPEG 2000 codec project. The utility imgcmp.exe can do a few different comparisons between one image and a reference image. All images must be in its supported format. I converted my image files to BMP files in order to use imgcmp.exe tool.

### Table 1: PSNR vs. FGS Scale

<table>
<thead>
<tr>
<th>FGS Scale</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>33.237207</td>
<td>32.973530</td>
<td>27.172984</td>
<td>31.12791</td>
</tr>
<tr>
<td>15</td>
<td>35.191471</td>
<td>36.081961</td>
<td>32.558506</td>
<td>34.61065</td>
</tr>
<tr>
<td>30</td>
<td>36.884419</td>
<td>37.973083</td>
<td>34.789846</td>
<td>36.54912</td>
</tr>
<tr>
<td>65</td>
<td>37.842335</td>
<td>39.058382</td>
<td>36.508694</td>
<td>37.80314</td>
</tr>
<tr>
<td>100</td>
<td>37.813576</td>
<td>39.025646</td>
<td>36.590201</td>
<td>37.80981</td>
</tr>
</tbody>
</table>

![Figure 10: FGS scale to PSNR. The result shows that PSNR increases along with FGS scale factor increases. Better FGS factors gives better PSNR.](image)

### Table 2: MSE vs. FGS Scale

<table>
<thead>
<tr>
<th>FGS Scale</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30.857426</td>
<td>32.788941</td>
<td>124.675742</td>
<td>62.77404</td>
</tr>
<tr>
<td>15</td>
<td>19.675838</td>
<td>16.028181</td>
<td>36.076949</td>
<td>23.92699</td>
</tr>
<tr>
<td>30</td>
<td>13.324119</td>
<td>10.369839</td>
<td>21.582205</td>
<td>15.09205</td>
</tr>
<tr>
<td>65</td>
<td>10.686780</td>
<td>8.076853</td>
<td>14.528173</td>
<td>11.09727</td>
</tr>
<tr>
<td>100</td>
<td>10.757782</td>
<td>8.137964</td>
<td>14.258056</td>
<td>11.05127</td>
</tr>
</tbody>
</table>
Figure 11: FGS scale to MSE. The result shows that MSE decreases along with FGS scale factor increases. Better FGS factors gives better MSE. Because the video stream is dominated in blue color, MSE in blue channel is much high than R and G channels.

Table 3: RMSE vs. FGS Scale

<table>
<thead>
<tr>
<th>FGS Scale</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.554946</td>
<td>5.726163</td>
<td>11.16583</td>
<td>7.482313</td>
</tr>
<tr>
<td>15</td>
<td>4.435745</td>
<td>4.003521</td>
<td>6.006409</td>
<td>4.815225</td>
</tr>
<tr>
<td>30</td>
<td>3.650222</td>
<td>3.220224</td>
<td>4.645665</td>
<td>3.838704</td>
</tr>
<tr>
<td>65</td>
<td>3.269064</td>
<td>2.841981</td>
<td>3.811584</td>
<td>3.307543</td>
</tr>
<tr>
<td>100</td>
<td>3.279906</td>
<td>2.852712</td>
<td>3.775984</td>
<td>3.302867</td>
</tr>
</tbody>
</table>

Figure 12: FGS scale to RMSE. The result shows that RMSE is decreases along with FGS scale factor increases. Better FGS factors gives better RMSE. Because the video stream is dominated in blue color, RMSE in blue channel is much high than R and G channels.
Appendix:

### 6.1 About the source code

Source codes of this project are based on MSSG open source MPEG2 codec mpeg2v12 and player mpg2w11b. Original MPEG2 codec source code can be found at [http://www.mpeg.org/MPEG/MSSG/](http://www.mpeg.org/MPEG/MSSG/). Original MPEG2 player source code can be found by searching “mpg2w11b.zip” at [www.google.com](http://www.google.com).

For the MPEG2 codec, I have modified many files in the project and added a few new files for bitplane codec. I cannot list all of my modifications here. If someone is willing to see my modification, he can compare my files with the original source files.

My new FGS enabled MPEG2 player is largely different from the original MPEG2 player mpg2w11b. My player is actually a combination of my MPEG2 FGS decoder and the enhanced GUI of original MPEG2 player mpg2w11b. Only GUI part of the original MPEG2 player source code is used and enhanced.

I wrote the bitplane encoding and decoding functions in “bitplane.c” files. There are two “bitplane.c” files. One is for the encoder; another one is for the decoder. They contain corresponding bitplane codec functions for MPEG FGS encoder and decoder.

### 6.2 Command options and examples

#### 6.2.1 Encoder

Command line usage of the encoder is simple, but you need to specify all encoder options in the parameter file. Check the ntsc.par file in the test stream directory for further information. You can also read the document of the original MPEG2 codec.

Example:

```bash
mpeg2enc ntsc.par out.m2v
```

#### 6.2.2 Decoder

Decoder has more command line options. Use the decoder execution file without options, the command will display its usage for you. You can also read the document of the original MPEG2 codec for further information.

Example:

```bash
mpeg2enc -b in.m2v -f -8 -7 fgsfilename -6 100
```

#### 6.2.3 Player

The new MPEG2 player can be used without command options to play normal MPEG2 video stream files. If you need to play FGS enabled files, you have to give FGS related parameters in the command line:

Example:

```bash
mpeg2play -b dream.m2v -8 -9 -7 dreamwork-%07d -6 100
```

---

Page 15 of 16
6.3 Further information

You can get further information about the source code in the document files that are included in the source code package.

7 References:

Li, Weiping, Fine Granularity Scalability in MPEG-4 for streaming Video, IEEE, 0-7803-5482-6/99

Li, Weiping, Overview of Fine Granularity Scalability in MPEG-4 Video Standard, IEEE, 1051-8215, 2001

Ling, F., Li, W., Sun H., Bitlevel Coding of DCT Coefficients for Image and Video Compression, SPIE Vol. 3653, P500-507, 1999

