An Industry Standard Benchmark Consortium

# DENBench™ Version 1.0

# Benchmark Name: MPEG-4 Encode

## Highlights

- **Benchmarks potential performance of an MPEG-4 encoder**
- **Five different test files stress different encoder aspects**
- **Integer implementation**
- **Based on open source XviD code base**
- **Implements PSNR to check the output quality**

## Application

**The MPEG-4 Encode benchmark provides an indication of the potential performance of a microprocessor subsystem running an MPEG-4 encoder application. MPEG-4 encode and decode are both popular in mobile devices and on the Internet, as well as in digital television and video over IP applications. We implement the XviD codec and code base with modifications for benchmarking and proprietary datasets.**

## Benchmark Description

EEMBC's XviD implementation uses simple profile/level 3 to encode the files. Since we generate the encoded files from raw YUV images, we have the option to create encoded MPEG-4 files with different profiles if needed.

The input data sets (YUV files) are the same as for EEMBC's MPEG-2 benchmarks.

Within the MPEG-4 standard, profiles and levels are defined to ensure interoperability between encoders and decoders. Profiles restrict the MPEG-4 features used, such as b-frames and quarterpel interpolation, while levels impose restriction on bit rate, memory, and complexity. These profiles are published in the MPEG-4 visual standard and available informally from the M4IF website. DivXNetworks has also defined its own profiles and levels, which are supported by hardware carrying the DivX Certified logo. XviD, the open source code base, is DivX spelled backwards.

The XviD encoder can be configured to generate any of the profiles above, although often it is configured to generate content using the complete ASP feature set. In addition, XviD presently does not perform video-buffer-verification, and as such, XviD content may not conform to the bit rates specified in any levels (ISO/IEC, DivX, or otherwise).

The M4IF definitions and discussion of profile and level are here:

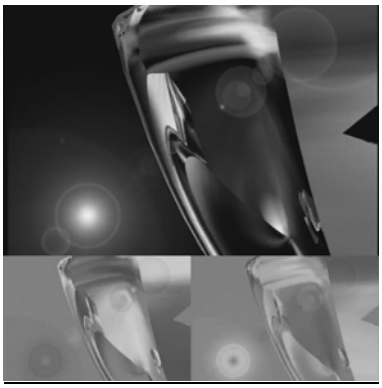
http://www.m4if.org/resources/profiles/index.php

The benchmark contains two different variations: a fixed-point integer and an optional single-precision floating point (f_32) implementation.

**The input is an a series of .PPM files along with YUV, and the outputs are mp4u files which can be decoded by XVid to help verify that the encoding was correct (assuming you use uuencode option in the Test Harness). Correctness test is also performed by measuring quality using Peak Signal to Noise Ratio analysis.**

## Description of Datasets

### Graphic



Graphic is a black background ray traced sequence with reflections, and moving light sources with coronas.

The primary elements are the reflections, a secondary halo from the first light source, and a few small artifacts on the front most graphic.

It is derived from an mpeg transport stream of encapsulated video.

### Graphic MPEG-4 Encode

A sequence of 7 frames is used for encoding. This results in a 3-second run, and keeps the ram file requirements under 4 Mb.

### Rail Grind

1. Notes
    a. 135 frame decoder source
    b. 30 frame encoder source

Rail Grind is a sequence of a skateboarder doing a grind move down a handrail and landing in an open space. The camera is centered on the skateboarder, which results in a fast moving color background.

The artifacts to watch for are tearing of the lower background at the bottom part of the rail move.
The original is an mpeg system stream with video on channel 0.

### Rail Grind MPEG-4 Encoding

A sequence of 30 frames is used for railgrind encoding.

**Sign**

1. Notes
    a. 300 frame decoder source
    c. 30 frame encoder source

Sign shows a person using sign language. There is a zoom-in effect to the speaker, with a complex color background of people.

It is derived from an mpeg system stream with video on channel 1.

The artifacts to look for are some small colorblocks appearing in the bottom lines of the picture.

The original input dimensions are 352x256, however the encoder only correctly decodes this with image size set to 352x240.

**Sign MPEG-4 Encoding**
The first 30 frames of sign are used for encoding.

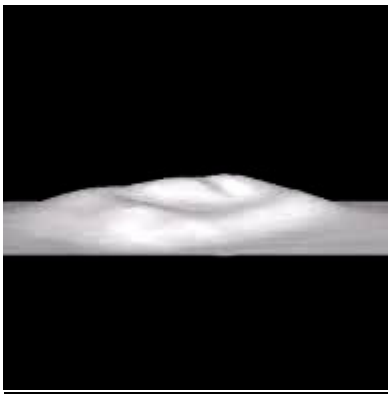
**Zoom**

1. Notes
    a. 65 frame decoder source
    b. 30 frame encoder source

Zoom is a beach scene with a rapid zoom out effect.

The original input was an AVI file, extracted to bitmaps. These bitmaps were converted to PPM files, and re-encoded at 30 fps. The final YUV files were generated from the encoded file.

**Zoom MPEG-4 Encoding**
The first 30 frames are used for encoding.



**Marsface**

1. Notes
    a. 49 frame decoder source
    d. 49 frame encoder source

Marsface is a rotating black and white radar picture of a Mars feature. The feature is three dimensional with a perspective view.

The original is a 24-fps mpeg file. This file format is kept for the deocder. For encoding, the bitrate is increased as well as the fps. An fps of 25 is the closest available setting in the encoder.

Original Attributes:
SEQUENCE PROG 192x192 chroma 96x96 fps 24 maxBps 0 vbv 32768
picture 192x192 display 192x192 pixel 1x1

**Marsface MPEG-4 Encoding**
All 49 frames are used for encoding.

**Benchmark Processing**

Processing consists of:

1. Reading the selected YUV frames.
2. Reading and interpreting the header information.
3. Read and encode frames of data
4. Process the data based on the header information
5. Output the .mpeg file into memory
6. A PSNR value is calculated

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

**Quality Measurements**

EEMBC has developed a proprietary methodology for measuring the quality of the MPEG-2 output based on peak signal-to-noise ratio (PSNR) code. PSNR is a decibel measurement of noise power used widely and consistently to measure picture and audio quality. In the EEMBC benchmarks, PSNR is measured outside the benchmark timing loop and on the host, not on the target board.

**Double-Ended Signal Quality Measurement**

The PSNR methodology implemented by EEMBC is enhanced to provide individual scores for encode and decode steps. This is still double-ended signal quality measurement; we have just introduced a second set of encoder reference files, and host processing steps, to provide additional information.

The host decoder is used to convert the output files from the embedded encoder. These result files are then used to calculate the PSNR score for the encoder. The final results for PSNR are a fundamentally different calculation from benchmark timing. The Test Harness produces a benchmark timing score represented as a single number.

**For PSNR, each benchmark is required to produce large volumes of data, i.e. on the order of several megabytes. This log file data is post-processed on the host to generate a collection of PSNR scores for each benchmark. The PSNR scores are aggregated by geometric mean. The traditional tab-delimited log file for benchmark timing plus an additional file in comma-separated value (.csv) format is produced for PSNR. Both summary files are readable by spreadsheet programs.**

**PSNR requires individual frame files from the target and a set of reference files for comparison. The YUV file format involves three separate files for each frame.**

**PSNR Utility**

A utility program called **PSNR.exe** consists of the following components:

1. Math to calculate PSNR of two comparison images, or frames using sum of squared distances method. The advantage of this method is that it can handle images of different dimensions and stride. It is also efficient for handling the volume of files we are processing in a reasonable time.
2. Math to calculate PSNR of two integer arrays used for processing PCM data. The bit depth of the PCM data is also used to support 8-, 16-, 24-, and 32-bit PCM data as required.
3. PSNR aggregation methods which include:
    a. geometric mean calculation
    b. arithmetic mean calculation
    c. sample variance calculation
    d. detection and accumulation of exact match frames (PSNR infinity)
    e. detection and accumulation of all zero frames (PSNR infinity)
4. File processing routines to locate files in the EEMBC tree paths and filename processing to generate frame file names based on file format. This minimizes the post processing steps by locating all of the files required with a few parameters.
5. Image processing to determine file types, the file formats used for reading YUV12 and AIFF formats, and to output YUV12 as PGM and AIFF data as PCM.
6. A standard method for reviewing frames by developers with three types of error images for YUV, and PCM files for MP3. Error images can be generated to identify specific problems during porting. They can also be used in certification when verification beyond basic PSNR is needed.

Knee (2000) is a relevant overview of PSNR, the error image, and its usage to evaluate quality. (www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm)

**Analysis of Computing Resources**

**There are two implementations, fixed point and floating point. The floating-point version is optional. This is a benchmark that concentrates mostly on computational processing, rather than file I/O. PSNR scores must be reported.**

**Optimizations Allowed**

**Out of the Box/Standard C**
**Full Fury/Optimized**

- The C code must not be changed for out of the box

unless it must be modified to get it to compile. All changes must be documented and must not have a performance impact.

- For Out of the Box, additional hardware can be used if it does not require code changes.
- All optimized libraries must be part of the standard compiler package, and/or available to all customers.
- The EEMBC Test Harness Regular or Test Harness Lite may be used. Test harness changes may be made for portability reasons if they do not impact performance.
- For Optimized, the basic algorithm may be changed and/or the code can be rewritten in assembler. We report PSNR scores to help you judge quality of computational processing.
- For Optimized, optimized libraries can be used if they are publicly available.
- For Optimized, hardware-assist can be used if it is on the same processor as that being benchmarked.
- For Optimized, in-lining is allowed.
- Additional data files are used by the EEMBC Technology Center (ETC) during certification to ensure the correctness of the optimized benchmark. You should not assume data patterns during optimization.