An Industry Standard Benchmark Consortium

DENBench™ Version 1.0

Benchmark Name: MPEG-2 Encode

Highlights

- **Five different test files to stress different aspect of encoders**
- **Floating point and integer implementations**
- **Implements PSNR to check output quality**

Application

The MPEG-2 Encode benchmark provides an indication of the potential performance of a microprocessor running an MPEG-2 encoder application.

Benchmark Description

The benchmark contains two different variations: an optional single-precision floating-point algorithm and a fixed-point version derived from ISO sources. MPEG-2 Encode uses a fairly standard reference implementation of the core algorithm, including Huffman decoding and modified inverse discrete cosine transform (iDCT) routines.

The fixed-point implementation base algorithm of `fdctint.c` and `jfdctint.c` is based on Loeffler et al (1989) [1].

For testing purposes, the benchmark was built and tested under Microsoft Windows using gcc and Visual C, Diab Data for PowerPC, gcc under Solaris (64-bit), Green Hills Software for ARM, and other compilers.

The input is a series of five datasets, which take the form of .PPM files along with YUV. The output is an MPEG file (.MPEG) file which can be played using Windows Media Player or Apple Quicktime to help verify that the encoding was correct (assuming you use the uuencode option in the Test Harness). Correctness is also checked by cyclical redundancy checksum (CRC checking), and we measure quality using peak signal-to-noise ratio analysis. CRC is used as a checkpoint only, not as a canonical validation. Out-of-the-box certifications, most of the time, for most compilers, will have the same CRC values.

The Datasets

Dataset #1: "Graphic"

Description

"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.

The MPEG 2 parameter file is set to the following NTSC source parameters:

SEQUENCE MPEG2 MP@ML 720x480 chroma 360x240 fps 30

maxBps 1000000 vbv 229376

Picture 720x480 display 720x480 pixel 8x9

The Encoding Process

A sequence of seven frames is used for encoding. This results in a three-second run and keeps the ram file requirements under 4 Mbytes.

The resulting MPEG file size is 232,677 bytes. On a 1.6-GHz reference platform, one iteration of the encoding process takes 3.57 seconds total run time.

Dataset #2: "Railgrind"

Description

"Railgrind" shows a skateboarder performing 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 dataset uses a 135-decoder source and a 30-frame encoder source. The original is an MPEG system stream with video on channel 0.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 25 maxBps 100000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

The Encoding Process

A sequence of 30 frames is used for railgrind encoding. The resulting MPEG-2 file size is 122,578 bytes. The total runtime for one iteration of the encoding process is 2.3 seconds on a 1.6-GHz reference platform.

Dataset #3: "Sign"

Description

"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.

Artifacts may appear as small colorblocks appearing in the bottom lines of the picture.

The dataset uses a 300-frame decoder source and a 30-frame encoder source. The original input dimensions are 352x256, but the encoder only

correctly decodes this with image size set to 352x240.

SEQUENCE MPEG2 MP@ML PROG 352x240 chroma 176x120 fps 25

maxBps 95000 vbv 32768

Picture 352x240 display 352x240 pixel 1x1

The Encoding Process

The first 30 frames of sign are used for encoding. The resulting MPEG-2 file size is 118,940 bytes. The total runtime for one iteration of the encoding process is 1.9 seconds on a 1.6-GHz reference platform.

Dataset #4: "Zoom"

Description

"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" uses a 65-frame decoder source and a 30-frame encoder source.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 30 maxBps 95000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

The Encoding Process

The first 30 frames are used for encoding. The resulting MPEG-2 file size is 96,170 bytes. The total runtime for one iteration of the encoding process is 3.3 seconds on a 1.6-GHz reference platform.

Dataset #5: "Marsface"

Description

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

Marsface uses 49-frame decoder and encoder sources.

The original is a 24-fps MPEG file. This file format is maintained for the decoder. For encoding, the bitrate is increased as well as the fps rate. An fps rate 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

Generated attributes:

SEQUENCE MPEG2 MP@ML PROG 192x192 chroma 96x96

fps 25 maxBps 95000 vbv 65536

picture 192x192 display 192x192 pixel 1x1

The Encoding Process

All 49 frames are used for encoding.

The resulting MPEG-2 file size is 70,209 bytes. The total runtime for one iteration of the encoding process is 1.4 seconds on a 1.6-GHz reference platform.

Processing consists of:

1. Reading the selected YUV frames.
2. Reading and interpreting the header information.
3. Reading and encoding frames of data
4. Processing the data based on the header information
5. Outputting the .mpeg file into memory
6. Calculating a PSNR value

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 the 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) [2] 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.

References

[1] Loeffler, C., A. Ligtenberg and G. Moschytz. "Practical Fast 1-D DCT Algorithms with 11 Multiplications," *Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP 89), pp. 988-991.

[2] Knee, Mike. "A Single-Ended Picture Quality Measure for MPEG-2," (2000); http://www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm