Getting Started with Intel® Stress Bitstreams and Encoder 2016 – VP9

Version 2.2
Updated March 28, 2016

Contents

Stress Bitstreams

Package Description

This stream set is intended to validate a decoder for VP9* format compliance. It covers both VP9 Feature Set 1 and Feature Set 2, all profiles. The stream set consists of four basic packages—one for each profile—and three additional smaller packages with streams combining different profiles—profiles 0 and 1, 0 and 2, and all four profiles, and additional package for fast smoke testing.

Package	Contents
Profile 0 (68 streams)	• 32 files of 500 frames at 432×240 resolution • 32 files of 100 frames at 1920×1080 • 5 files of 10 frames at 3840×2160 • 1 file with variable resolution from 4×4 to 128×128 • 1 file of 50 frames at 4320×240 and 1 file of 50 frames at 16384×240 for tile testing • 1 file of 1 frame at 432×240 resolution for inverse transform testing (only 8-bit package)
Profile 2 (142 streams)	71 streams per 10 and 12 bit-depth value: • 32 files of 500 frames at 432×240 resolution • 32 files of 100 frames at 1920×1080 • 5 files of 10 frames at 3840×2160 • 1 file with variable resolution from 4×4 to 128×128 • 1 file of 50 frames at 4320×240 and 1 file of 50 frames at 16384×240 for tile testing
Profile 1 (87 streams)	33 streams for each chroma subsampling configuration (4:2:2, 4:4:0 and 4:4:4): • 16 files of 500 frames at 432×240 resolution • 16 files of 50 frames at 1920×1080 • 1 file with variable resolution from 4×4 to 128×128
Profile 3 (87 streams)	33 streams for 10-bit color depth and each chroma subsampling configuration (4:2:2, 4:4:0 and 4:4:4): • 16 files of 500 frames at 432×240 resolution • 16 files of 50 frames at 1920×1080 • 1 file with variable resolution from 4×4 to 128×128 The package doesn't contain 12-bit streams, because 10 and 12 bit decoding pipelines are the same. Support of Profile 3 12 bit configuration can be checked by combination of Profile 2, Profile 3 and Mix 0–3 packages. Additional tests can be created with Random Encoder.
Mix 0–1 (8 streams)	The streams containing frames of Profile 0 and Profile 1 with random subsampling settings for Profile 1. • 4 files of 500 frames at 432×240 resolution • 4 files of 50 frames at 1920×1080
Mix 0–2 (8 streams)	The streams containing frames of Profile 0 and Profile 2 with random bit depth for Profile 2. • 4 files of 500 frames at 432×240 resolution • 4 files of 50 frames at 1920×1080
Mix 0–3 (8 streams)	The streams with fully randomized profile, bit depth and chroma subsampling. • 4 files of 500 frames at 432×240 resolution • 4 files of 50 frames at 1920×1080
Smoke test (24 streams)	Small set of streams with full code coverage for fast smoke testing. The set contains 8 streams for each bit depth (8, 10, 12) with 4:2:0 subsampling: • 1 file of 256 frames at 64×64 • 5 files of 50 frames at 432×240 • 1 file of 10 frames at 1920×1080 • 1 file of 5 frames at 16384×64

Each package contains *all_bitstreams.md5* file with checksums for all encoded files. Each bitstream is complemented with MD5 sum for decoding result by libvpx reference implementation. “Intel Stress Bitstreams and Encoder 2015 - VP9 Description.xlsx” document describes which coding tools and value ranges are covered by each bitstream.

The decoding result is assumed correct if it binary-matches the result of the reference decoder. This can be verified with the MD5 files included within the package. All the streams are VP9-compliant; the stream set doesn’t contain any invalid streams for error-resilience testing. Also the stream set is not intended for decoder-performance testing since a lot of bitstream features have distributions not typical for “real-world” video: many long motion vectors pointing out of frame, areas with random noise, highly variable QP values, etc.

There are three basic buckets—*INTRA*, *INTER* and *EXTRA*—and additional buckets *MEMORY BANDWIDTH* and *COLOR* dedicated to specific tests. *INTRA* and *INTER* buckets validate the features related to intra and inter prediction, respectively, while *EXTRA* bucket contains streams covering other VP9 format features not directly related to intra or inter prediction.

There are three types of streams: *Syntax*, *Stress* and *Smaller*. *Syntax* streams are designed to test a certain subset of features, for example, all the intra-prediction modes or all loop-filter related parameters. *Stress* streams include all the features covered by the bucket, so they are useful for smoke testing: if a decoder passes the *Stress* stream, it is likely to pass all the *Syntax* streams from the bucket. A *Smaller* stream is similar to a *Stress* stream of *INTER* bucket, its major purpose is to test the decoder’s ability to handle very small resolutions. *Syntax* streams are provided in two versions: 500 frames of 432×240 and 100 frames of 1920×1080. The only exception for *Syntax* streams is the stream dedicated to tile testing, which in addition has 4320×240 version to test vertical tiles of maximum width. *Stress* streams also have an Ultra HD version: 10 frames of 3840×2160.

File name pattern

All files are named according to the following pattern:
(Purpose)_VP9_(profile)_(resolution)_(stream_index)_(bucket)_(short_name)_(encoder_version).vp9

- (Purpose) is “Syntax”, “Stress” or “Smaller”;
- (profile) contains information about feature set, profile, and may indicate bit depth of subsampling option; for example, “FC1”, “FC2_p2b10”, “FC3_ss440”;
- (Resolution) is a pair of frame width and height joint with “x”
- (stream_index) is a 3-digit number. The first digit denotes a bucket (0 – intra, 1 – inter, 2 – extra, 3 – plum, 4 – memory bandwidth, 5 – profile mix) and the other two digits indicate the number of the stream in thebucket. Such indexing is useful for file sorting and test ordering: in most cases a stream with a smaller number is also less complex in terms of coding tools;
- (bucket): “intra”, “inter”, “extra”, “memory_bandwidth” or “color”;
- (short_name) describes which features this stream tests, for example, “tiles” or “loopfilter”;
- (encoder_version): version of encoder used for stream generation. Package may contain streams of different encoder versions.

For example, Syntax_VP9_1920x1080_106_inter_hidden_frames_2.2.vp9 is the sixth stream from *INTER* bucket and it tests support of hidden frames. Since the complexity of streams increases incrementally, the preceding streams in the bucket do not contain B-frames. Starting from this stream, some of the following streams may contain B-frames.

(stream_index) can be considered as a short ID of the stream. Pair of (bucket) and (short_name) is a short description of the content of the stream.

Video Content

Each bitstream is encoded from a synthetic video sequence with a subtitle describing *stream_index*, *bucket* and *short_name*. Encoded sequences may have strong visual artifacts, mostly due to high quantization parameter (QP) values, superblocks with random noise and randomly placed *skip_coeff* flag. For best known methods to avoid visual artifacts in output streams, see the next section.

Streams without Visual Artifacts

To get rid of visual artifacts, edit the encoder parfile:

1. Disable the following parameters that yield visual artefacts:

"frame" : "seg_skip_zeromv"

"superblock" : "skip"

"residual" : "type"

2. Lower the upper bound of the QP range using the following parameters:

"frame" : "base_qindex_range"

"frame" : "y_dc_delta_q_range"

"frame" : "uv_ac_delta_q_range",

"frame" : "uv_dc_delta_q_range",

"frame" : "seg_alt_q_delta"

Random Encoder may encode some frames as invisible, so they won't be present in decoder output. This may yield some jerking in playback. To make playback smoother, use "--target visual" command-line option. This will enable a special mode when each invisible frame is repeated with *show_existing_frame* flag. Note that disabling segmentation features and skip flag reduces the test coverage.

Stream Description

Test case (Stream name)	Description	Resolution, number of frames
Intra bucket (8 streams of 240p, 8 streams of 1080p, 1 stream of 2160p)		
001_intra_basic	key-frames only, DCPRED only, no LF, no segmentation, no tiles, no partitioning, tx_mode = ALLOW_8X8, no skip	432×240, 500 frames 1920×1080, 100 frames
002_intra_partitions	random partitioning from 64x64 to 4x4 with rectangular partitions, random tx_mode (frame level) and tx_size (sb level)	
003_intra_modes	all 10 intra modes, random skip	
004_intra_loopfilter	random values of loop filter at frame level	
005_intra_qp	random base_qindex and qp deltas	
006_intra_segmentation_basic	random segments, ALT_Q and ALT_LF features	
007_intra_segmentation_advanced	add ref_frame and skip segmentation features	
050_intra_stress	full intra-parameters randomization + random residual in some blocks	432×240, 500 frames 1920×1080, 100 frames 3840×2160, 10 frames

Inter bucket (12 streams of 240p, 12 streams of 1080p, 1 stream of 2160p)		
101_inter_basic	zeromv, only last frame ref, no skip, no LF, no segmentation, no tiles, fixed tx size, regular interpolation filter, 1/4 pel ME	432×240, 500 frames 1920×1080, 100 frames
102_inter_modes_vectors	all 4 modes + intra modes, random skip, no blocks < 8x8, allow 1/8 pel ME, vector over the frame boundary	
103_inter_sub8x8	block < 8x8 (special case of vector prediction)	
104_inter_interpolation	test all filters on frame and superblock levels	
105_inter_ref_frames	random values for lst_idx, gld_idx and alt_idx, random refresh_mask, random sign_bias	
106_inter_hidden_frames	hidden frames and intra_only hidden frames	
107_inter_compound_prediction	random selection of ref frames indices, random sign bias, single/compound prediction mode, references at SB level	
108_inter_loopfilter	random values for LF ref_deltas and mode_deltas	
109_inter_qp	random base_qindex and qp deltas	
110_inter_segmentation_advanced	full randomization of segmentation features	
111_inter_entropy_contexts	there are 4 flags responsible for error resilience and context refresh	
150_inter_stress	full parameters randomization + random residual in some blocks	432×240, 500 frames 1920×1080, 100 frames 3840×2160, 10 frames
Extra bucket (9 streams of 240p, 9 streams of 1080p, 2 streams of 2160p, 1 stream of variable resolution up to 128×128, 1 stream of 4320×240, 1 stream of 16384×240)		
201_extra_tiles	random tile settings for each frame	432×240, 500 frames 1920×1080, 100 frames 4320×240, 50 frames 16384×240, 50 frames
202_extra_residual	randomized residual after intra and inter prediction	432×240, 500 frames 1920×1080, 100 frames
203_extra_transform	random forward transform results	
204_extra_keyframe_scaling	key frames have different resolution	variable (max 432×240), 500 frames variable (max 1920×1080), 100 frames
205_extra_onfly_scaling	scaling of inter frames	
206_extra_repeat_existing	repeating a frame from ref buffer with one byte in header	432×240, 500 frames 1920×1080, 100 frames
207_extra_lossless	lossless mode and really lossless result	
208_extra_lossless_stress	lossless mode with features that not deliver lossless result	432×240, 500 frames 1920×1080, 100 frames 3840×2160, 10 frames
209_extra_smaller	test very small resolution from 4x4 to 128x128	from 4×4 to 128×128, 130 frames
210_extra_intra_only_first	specific case when stream starts with intra-only frame	432×240, 500 frames 1920×1080, 100 frames
210_extra_tokens	randomize transform coefficients by tokens	432×240, 500 frames 1920×1080, 100 frames
250_extra_stress	killer combo of all the features	variable (max 432×240), 500 frames variable (max 1920×1080), 100 frames variable (max 3840×2160), 10 frames
251_extra_stress_no_scaling	same as 250th one, but without frame scaling for easier debugging	432×240, 500 frames 1920×1080, 100 frames 3840×2160, 10 frames
264_extra_keyframe_scaling_min_size_64 265_extra_onfly_scaling_min_size_64 269_extra_stress_min_size_64	Same configurations as 204th, 205th and 250th parfiles, but with limitation of minimum frame size to 64×64.	432×240, 500 frames 1920×1080, 100 frames
Memory bandwidth bucket (1 sample stream of 1080p)		
401_memory_bandwidth	stress-test memory reading on inter-prediction	1920×1080, 60 frames

Color bucket (12 streams of 240p, 12 streams of 1080p)		
510_color_mix_intra_full 520_color_mix_intra_full 550_color_mix_intra_full	variable profile, bit depth and chroma subsampling, key frames only	432×240, 500 frames 1920×1080, 50 frames
511_color_mix_inter_no_lf_no_scaling 521_color_mix_inter_no_lf_no_scaling 551_color_mix_inter_no_lf_no_scaling	variable profile, bit depth and chroma subsampling, utilize all features except loopfilter and frame scaling	432×240, 500 frames 1920×1080, 50 frames
512_color_mix_inter_no_scaling 522_color_mix_inter_no_scaling 552_color_mix_inter_no_scaling	variable profile, bit depth and chroma subsampling, utilize all features except frame scaling	432×240, 500 frames 1920×1080, 50 frames
519_color_mix_stress 529_color_mix_stress 559_color_mix_stress	250_extra_stress with variable profile, bit depth and chroma subsampling	variable (max 432×240), 500 frames variable (max 1920×1080), 100 frames
Worst-case performance configuration (parfile-only delivery)		
402_worst_performance_19mbps 403_worst_performance_80mbps. 404_worst_performance_200mbps	streams targeted to stress performance of inter-prediction, loop filter and bool decoder	parfile-only delivery
Visually clean configuration (parfile-only delivery)		
v050_intra_stress v150_inter_stress v208_extra_lossless_stress v251_extra_stress_no_scaling	configurations for visual validation when binary match check is unavailable	parfile-only delivery

Methodology

It is easy to check if the decoder passes all the tests: MD5 checksum decoding results must be identical to those included in package. If something goes wrong, you can use the stream sets to quickly identify which feature of the VP9 format causes the problem. The complexity of streams in the set rises with *stream_index*: a decoder is unlikely to pass *002_intra_partitions* test if it has not passed *001_intra_basic*. Also there is a dependency between *INTRA* and *INTER* buckets: it is impossible to pass *110_inter_segmentation_advanced* without properly supporting all features required for passing segmentation tests from *INTRA* bucket.

If a decoder fails the *110th* test but passes all the previous tests, the problem is most likely related to segmentation features at INTER frames. Investigation of mismatch root cause will require comparison of reference and test results, debugging the decoders or using special tools for bitstream analysis and mode-decision visualization.

Each *Syntax* test case is provided in two resolutions: 1920×1080 and 432×240. Both streams have the same *stream_index* and *short_name* and cover exactly the same syntax-elements scope. In most cases a stream of smaller resolution will be more convenient for the developer's testing and debugging purposes, but not all combinations of syntax elements are guaranteed to be available at 240p resolution, so 1080p streams are required for final validation. *Stress* test cases in addition to 240p and 1080p have additional streams of 3840×2160 resolution.

Details on Reference Decoder

MD5 files were generated for plain-YUV420 results of libvpx reference decoder with all SIMD optimizations disabled. The following execution string was used:

```
vpxdec <input>.vp9 –o <output>.yuv --rawvideo -t 1
```

libvpx code snapshot from July 7, 2015
(Git revision 8565a1c99a18d29a72fb716ad64614dc8e92499f)

For high-bit-depth streams validation, we build the decoder with *--enable-vp9_highbitdepth* configure option. For FC2* streams, we also add option --output_bit_depth=16, and MD5 sums for such streams are generated for output aligned to 16-bit color depth.

Handling Streams with Variable Frame Resolution

Streams with variable frame resolution are a specific case. There is no way to view them as plain YUV files, since information about resolution is lost during decoding. Possible solutions are to use Y4M format for saving information about each frame resolution or to add an option to the decoder making it possible to dump each frame to a separate YUV file.

"Plum" Bucket – Specially Picked Streams

In the "Plum" bucket we collect streams, which test rare corner cases not covered by original package. Currently this bucket contains only one stream.

The filenames of all streams in this bucket start with "Plum_", test case index is 3XX. Streams were generated with Random Encoder for VP9 with one of parfiles used for original package creation, so instead of description of all the options for each stream we specify parfile name referring to certain line in stream-description spreadsheet.

Test case (Stream name)	Basic parfile	Resolution, # of frames	Description
301_idct_rounding	050_intra_stress	432×240, 1 frame	The stream test that dct_const_round_shift() function used for inverse transform performs proper int16_t cast. If it doesn't, the test result will mismatch the reference one for some extreme values of randomized residual.

Note that due to different definitions of *tran_low_t* in default and high-bit-depth configurations (*int16_t* vs *int32_t*), these configurations produce **different results** for the 301st stream. We calculated md5 sums using decoder with high-bit-depth option enabled. For the 301st stream we put into package additional md5 file named *Plum_VP9_432x240_301_idct_rounding_1.1_dec_hbd_disabled.md5*, so you can verify against configuration without high-bit-depth support.

Memory Bandwidth Bucket

Memory bandwidth bucket is dedicated to stress-test decoder memory throughput on inter-prediction. The bucket contains only one stream of 1080p resolution and streams of other resolutions are to be generated by the Random Encoder with *401_memory_bandwidth.json* parfile.

Color Bucket: Streams with Mixed Profiles

With this bucket, you can test how decoder handles switching between profiles, bit-depth and chroma-subsampling settings. The bucket has three sets:

- Profiles 0 and 1 (with randomized configuration of chroma subsampling for profile 1), index 51X;
- Profiles 0 and 2 (for profile 2 both 10 and 12 bit depth are allowed), index 52X;

- All profiles 0, 1, 2, 3 with arbitrary settings inside profile, index 55X.

To generate each set we used the following four parfiles:

- 5X0_color_mix_intra_full – basic test using only key-frames, derived from 050th parfile;
- 5X1_color_mix_inter_no_lf_no_scaling – this parfile is full feature configuration derived from 250th parfile with disabled loopfilter and frame scaling;
- 5X2_color_mix_inter_no_scaling – same as 5X1, but with loopfilter enabled;
- 5X9_color_mix_stress – same as 5X2, with frame scaling.

Change of profile, bit depth or subsampling may happen only on key frame.

Random Encoder

Introduction

Random Encoder is a command-line application that accepts the following parameters:

```
Usage: vp9_random_encoder -i <input.yuv> -o <output.vp9> -w <width> -h <height>
                          [-p <config.json> -s <seed> -r <reconstruct.yuv>]
                          [-f <# of frames to process>]

Options:
  -i STRING, --input=STRING
                        input file (raw yuv)
  -o STRING, --output=STRING
                        output vp9 bitstream
  -w INT, --width=INT   width of input yuv
  -h INT, --height=INT  height of input yuv
  -f INT, --frames=INT  number of frames to process
  -r STRING, --recon=STRING
                        output file for encoder's reconstruct (optional)
  -p STRING, --parfile=STRING
                        json file with distribution parameters (optional)
  -s INT, --seed=INT    seed for random engine
  --dump_hidden=BOOL    write invisible frames to reconstruct file
  --output_state=STRING
                        path where to write states of random engine at keyframes
  --input_state=STRING  path to input file with random engine state
  --bit_depth=INT       bitstream bit depth (8, 10, 12), overwrites parfile setting
  --recon_bit_depth=INT
                        encoder-reconstruct bit depth (8, 10, 12, 16),
                        for easier comparison with results of reference decoder
  --colorspace=STRING   bitstream colorspace: yuv420|yuv422|yuv440|yuv444,
                        overwrites parfile setting
  --extreme_residual=BOOL
                        use only extreme values for random residual
  --idct_overflow_hw=BOOL
                        enable Emulate-Hardware-Highbitdepth behavior for iDCT
                        calculation
  --verbose             print ref resolution for each frame (if frame scaling is
                        enabled)
  --help                output this message
```

Input and Output

Random Encoder takes uncompressed YUV stream as input and encodes it with random mode decisions. Output is VP9 bistream in IVF container. Modes to randomize and range of randomization are defined by

parfile (`-p` option). Parfile is a JSON file with comments, most of parameters are named according to VP9 reference encoder coding tools. Example vp9_renc_stress_cfg.json is set to maximum randomization of all the modes.

Parfile Editing: Weight and Range Parameters

Parfile is a .json file with comments. It has four general sections related to different groups of parameters: the "frame" and "superblock" sections allow to control syntax-element randomization scope in frame header and at superblock level. Most of members of these two sections are named according to variables in libvpx source code. The "scaling" section defines the way frame resolution will be varied in the encoded stream. The "residual" section contains settings related to residual randomization at superblock level. The "override" section allows redefining settings for specified frames.

There are two kinds of parameters: weights and range. In most cases logical or enumerated parameters are defined by relative weights. For example, setting "frame_type" : [1, 10] means one chance of key frame against 10 chances of inter frame. Numeric parameters with wide range are defined by specifying minimum and maximum values. For example, loop filter level acceptable values are from 0 to 63, and in parfile it is defined as "lf_filter_level" : [0, 63]. Actual values will be spread uniformly in this range.

To disable randomization for weights-defined parameters, weights for all except one values must be set to zero. To fix range-defined parameter you should set min and max values to the same value.

Encoder-output Validation

Random Encoder can output results of internal reconstruct to YUV file (`-r` parameter). It is useful to validate the random encoder by checking that it matches the reference decoder results. For example:

```
REM Execute random encoder on standard "foreman" sequence.
vp9_random_encoder -i foreman_352x288_300.yuv -p vp9_renc_stress_cfg.json -o
foreman_rnd.vp9 -r foreman_rnd_352x288_recon.yuv -w 352 -h 288 --dump_hidden 0

REM Decode generated bitstream with reference libvpx decoder.
vpxdec foreman_rnd.vp9 -o foreman_rnd_352x288_dec.yuv --rawvideo

REM Verify that encoder's internal reconstruct is bit-exact to result of
REM reference decoder by ensuring that their MD5 sums are identical.
md5sum -b foreman_rnd_352x288_recon.yuv foreman_rnd_352x288_dec.yuv
```

If frame scaling is enabled, it is impossible to view YUV reconstruct without metadata about frame resolution, so the random encoder also outputs .res file where it writes this information in the following format, one line per frame: <frame number> <frame width> <frame height>.

Exhaustive Decoder Validation

Seed parameter `-s` defines initial random-engine state. Changing this seed allows to produce different streams with the same parfile. This feature is helpful to create small bug-reproducers (setting frame number parameter `-f` to some small value) for the parfiles which are known to generate the streams causing failure of examined decoder.

Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting **www.intel.com/design/literature.htm**.

Intel, the Intel logo, Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804