x265

Release

May 08, 2014

Contents

CHAPTER 1

Introduction

Increase in demand for high resolution video along with increase in the consumption of video can be met only with the supply of a high efficiency codec. The x265 codec from MCW aims at providing the most efficient video encoder with the highest possible quality at even the lowest given bit rate.

1.1 About HEVC

The High Efficiency Video Coding (HEVC) is the latest generation video compression standard. This standard was developed by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG), through their Joint Collaborative Team on Video Coding (JCT-VC). HEVC is also known as ISO/IEC 23008-2 MPEG-H Part 2 and ITU-T H.265. HEVC provides superior video quality and up to twice the data compression as the previous standard (H.264/MPEG-4 AVC). HEVC can support 8K Ultra High Definition video, with a picture size up to 8192x4320 pixels.

1.2 About x265

The primary objective of x265 is to become the best H.265/HEVC encoder available anywhere, offering the highest compression efficiency and the highest performance on a wide variety of hardware platforms. The x265 encoder is available as an open source library, published under the GPLv2 license. It is also available under a commercial license similar to the x264s commercial license.

x265 is developed by MulticoreWare, leaders in high performance software solutions, with backing from leading video technology providers including Telestream and Doremi Labs (and other companies who want to remain anonymous at this time), and with contributions from open source developers. x265 leverages many of the outstanding video encoding features and optimizations from the x264 AVC encoder project.

The x265 software is available for free under the GNU GPL 2 license, from https://bitbucket.org/multicoreware/x265. For commercial companies that wish to distribute x265 without being subject to the open source requirements of the GPL 2 license, commercial licenses are available with competitive terms. Contact license @ x265.com to inquire about commercial license terms.

While x265 is primarily designed as a video encoder software library, a command-line executable is provided to facilitate testing and development. We expect x265 to be utilized in many leading video hardware and software products and services in the coming months.

CHAPTER 2

Command Line Options

Note that unless an option is listed as **CLI ONLY** the option is also supported by x265_param_parse(). The CLI uses getopt to parse the command line options so the short or long versions may be used and the long options may be truncated to the shortest unambiguous abbreviation. Users of the API must pass x265_param_parse() the full option name.

Preset and tune have special implications. The API user must call x265_param_default_preset() with the preset and tune parameters they wish to use, prior to calling x265_param_parse() to set any additional fields. The CLI does this for the user implicitly, so all CLI options are applied after the user's preset and tune choices, regardless of the order of the arguments on the command line.

If there is an extra command line argument (not an option or an option value) the CLI will treat it as the input filename. This effectively makes the *--input* specifier optional for the input file. If there are two extra arguments, the second is treated as the output bitstream filename, making *--output* also optional if the input filename was implied. This makes **x265 in.y4m out.hevc** a valid command line. If there are more than two extra arguments, the CLI will consider this an error and abort.

Generally, when an option expects a string value from a list of strings the user may specify the integer ordinal of the value they desire. ie: *--log-level* 3 is equivalent to *--log-level* debug.

2.1 Standalone Executable Options

--help, -h
Display help text

CLI ONLY

--version, -V
Display version details

CLI ONLY

--asm <integer:false:string>**, --no-asm**
x265 will use all detected CPU SIMD architectures by default. You can disable all assembly by using *--no-asm* or you can specify a comma separated list of SIMD architectures to use, matching these strings: MMX2, SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2, AVX, XOP, FMA4, AVX2, FMA3

Some higher architectures imply lower ones being present, this is handled implicitly.

One may also directly supply the CPU capability bitmap as an integer.

--threads <integer>
Number of threads for thread pool. Default 0 (detected CPU core count)

--preset, -p <integer|string>
Sets parameters to preselected values, trading off compression efficiency against encoding speed. These parameters are applied before all other input parameters are applied, and so you can override any parameters that these values control.

0.ultrafast

1.superfast

2.veryfast

3.faster

4.fast

5.medium **(default)**

6.slow

7.slower

8.veryslow

9.placebo

--tune, -t <string>
Tune the settings for a particular type of source or situation. The changes will be applied after *--preset* but before all other parameters. Default none

Values: psnr, ssim, zero-latency, fast-decode.

--frame-threads, -F <integer>
Number of concurrently encoded frames. Using a single frame thread gives a slight improvement in compression, since the entire reference frames are always available for motion compensation, but it has severe performance implications. Default is an autodetected count based on the number of CPU cores and whether WPP is enabled or not.

--log-level <integer|string>
Logging level. Debug level enables per-frame QP, metric, and bitrate logging. If a CSV file is being generated, debug level makes the log be per-frame rather than per-encode. Full level enables hash and weight logging. -1 disables all logging, except certain fatal errors, and can be specified by the string "none".

0.error

1.warning

2.info **(default)**

3.debug

4.full

--csv <filename>
Writes encoding results to a comma separated value log file. Creates the file if it doesnt already exist, else adds one line per run. if *--log-level* is debug or above, it writes one line per frame. Default none

--output, -o <filename>
Bitstream output file name. If there are two extra CLI options, the first is implicitly the input filename and the second is the output filename, making the *--output* option optional.

The output file will always contain a raw HEVC bitstream, the CLI does not support any container file formats.

CLI ONLY

--no-progress
Disable CLI periodic progress reports

CLI ONLY

2.2 Quality reporting metrics

--ssim, --no-ssim
Calculate and report Structural Similarity values. It is recommended to use *--tune* ssim if you are measuring ssim, else the results should not be used for comparison purposes. Default disabled

--psnr, --no-psnr
Calculate and report Peak Signal to Noise Ratio. It is recommended to use *--tune* psnr if you are measuring PSNR, else the results should not be used for comparison purposes. Default disabled

2.3 Input Options

--input <filename>
Input filename, only raw YUV or Y4M supported. Use single dash for stdin. This option name will be implied for the first "extra" command line argument.

CLI ONLY

--y4m
Parse input stream as YUV4MPEG2 regardless of file extension, primarily intended for use with stdin (ie: *--input* - *--y4m*). This option is implied if the input filename has a ".y4m" extension

CLI ONLY

--input-depth <integer>
YUV only: Bit-depth of input file or stream

Values: any value between 8 and 16. Default is internal depth.

CLI ONLY

--dither
Enable high quality downscaling. Dithering is based on the diffusion of errors from one row of pixels to the next row of pixels in a picture. Only applicable when the input bit depth is larger than 8bits and internal bit depth is 8bits. Default disabled

CLI ONLY

--input-res <wxh>
YUV only: Source picture size [w x h]

CLI ONLY

--input-csp <integer|string>
YUV only: Source color space. Only i420, i422, and i444 are supported at this time. The internal color space is always the same as the source color space (libx265 does not support any color space conversions).

0.i400

1.i420 **(default)**

2.i422

3.i444

4.nv12

5.nv16

--fps <integer|float|numerator/denominator>
YUV only: Source frame rate

Range of values: positive int or float, or num/denom

--interlaceMode <false|tff|bff>**, --no-interlaceMode**
EXPERIMENTAL Specify interlace type of source pictures.

0.progressive pictures **(default)**

1.top field first

2.bottom field first

HEVC encodes interlaced content as fields. Fields must be provided to the encoder in the correct temporal order. The source dimensions must be field dimensions and the FPS must be in units of fields per second. The decoder must re-combine the fields in their correct orientation for display.

--seek <integer>
Number of frames to skip at start of input file. Default 0

CLI ONLY

--frames, -f <integer>
Number of frames to be encoded. Default 0 (all)

CLI ONLY

2.4 Quad-Tree analysis

--wpp, --no-wpp
Enable Wavefront Parallel Processing. The encoder may begin encoding a row as soon as the row above it is at least two CTUs ahead in the encode process. This gives a 3-5x gain in parallelism for about 1% overhead in compression efficiency. Default: Enabled

--ctu, -s <64|32|16>
Maximum CU size (width and height). The larger the maximum CU size, the more efficiently x265 can encode flat areas of the picture, giving large reductions in bitrate. However this comes at a loss of parallelism with fewer rows of CUs that can be encoded in parallel, and less frame parallelism as well. Because of this the faster presets use a CU size of 32. Default: 64

--tu-intra-depth <1..4>
The transform unit (residual) quad-tree begins with the same depth as the coding unit quad-tree, but the encoder may decide to further split the transform unit tree if it improves compression efficiency. This setting limits the number of extra recursion depth which can be attempted for intra coded units. Default: 1

--tu-inter-depth <1..4>
The transform unit (residual) quad-tree begins with the same depth as the coding unit quad-tree, but the encoder may decide to further split the transform unit tree if it improves compression efficiency. This setting limits the number of extra recursion depth which can be attempted for inter coded units. Default: 1

2.5 Temporal / motion search options

--me <integer|string>
Motion search method. Generally, the higher the number the harder the ME method will try to find an optimal match. Diamond search is the simplest. Hexagon search is a little better. Uneven Multi-Hexegon is an adaption of the search method used by x264 for slower presets. Star is a three step search adapted from the HM encoder: a star-pattern search followed by an optional radix scan followed by an optional star-search refinement. Full is an exhaustive search; an order of magnitude slower than all other searches but not much better than umh or star.

0.dia

1.hex **(default)**

2.umh

3.star

4.full

--subme, -m <0..7>
Amount of subpel refinement to perform. The higher the number the more subpel iterations and steps are performed. Default 2

-m	HPEL iters	HPEL dirs	QPEL iters	QPEL dirs	HPEL SATD
0	1	4	0	4	false
1	1	4	1	4	false
2	1	4	1	4	true
3	2	4	1	4	true
4	2	4	2	4	true
5	1	8	1	8	true
6	2	8	1	8	true
7	2	8	2	8	true

--merange <integer>
Motion search range. Default 57

The default is derived from the default CTU size (64) minus the luma interpolation half-length (4) minus maximum subpel distance (2) minus one extra pixel just in case the hex search method is used. If the search range were any larger than this, another CTU row of latency would be required for reference frames.

Range of values: an integer from 0 to 32768

--rect, --no-rect
Enable analysis of rectangular motion partitions Nx2N and 2NxN (50/50 splits, two directions). Default enabled

--amp, --no-amp
Enable analysis of asymmetric motion partitions (75/25 splits, four directions). This setting has no effect if rectangular partitions are disabled. Even though there are four possible AMP partitions, only the most likely candidate is tested, based on the results of the rectangular mode tests. Default enabled

--max-merge <1..5>
Maximum number of neighbor (spatial and temporal) candidate blocks that the encoder may consider for merging motion predictions. If a merge candidate results in no residual, it is immediately selected as a "skip". Otherwise the merge candidates are tested as part of motion estimation when searching for the least cost inter option. The max candidate number is encoded in the SPS and determines the bit cost of signaling merge CUs. Default 2

--early-skip, --no-early-skip
Measure full CU size (2Nx2N) merge candidates first; if no residual is found the analysis is short circuited. Default disabled

--fast-cbf, --no-fast-cbf
Short circuit analysis if a prediction is found that does not set the coded block flag (aka: no residual was encoded). It prevents the encoder from perhaps finding other predictions that also have no residual but require less signaling bits. Default disabled

--ref <1..16>
Max number of L0 references to be allowed. This number has a linear multiplier effect on the amount of work performed in motion search, but will generally have a beneficial affect on compression and distortion. Default 3

--weightp, -w, --no-weightp
Enable weighted prediction in P slices. This enables weighting analysis in the lookahead, which influences slice decisions, and enables weighting analysis in the main encoder which allows P reference samples to have a weight function applied to them prior to using them for motion compensation. In video which has lighting changes, it can give a large improvement in compression efficiency. Default is enabled

--weightb, --no-weightb
Enable weighted prediction in B slices. Default disabled

2.6 Spatial/intra options

--rdpenalty <0..2>
Penalty for 32x32 intra TU in non-I slices. Default 0

Values: 0:disabled 1:RD-penalty 2:maximum

--tskip, --no-tskip
Enable intra transform skipping (encode residual as coefficients) for intra coded blocks. Default disabled

--tskip-fast, --no-tskip-fast
Enable fast intra transform skip decisions. Only applicable if transform skip is enabled. Default disabled

--strong-intra-smoothing, --no-strong-intra-smoothing
Enable strong intra smoothing for 32x32 intra blocks. Default enabled

--constrained-intra, --no-constrained-intra
Constrained intra prediction. When generating intra predictions for blocks in inter slices, only intra-coded reference pixels are used. Inter-coded reference pixels are replaced with intra-coded neighbor pixels or default values. The general idea is to block the propagation of reference errors that may have resulted from lossy signals. Default disabled

2.7 Slice decision options

--open-gop, --no-open-gop
Enable open GOP, allow I-slices to be non-IDR. Default enabled

--keyint, -I <integer>
Max intra period in frames. A special case of infinite-gop (single keyframe at the beginning of the stream) can be triggered with argument -1. Use 1 to force all-intra. Default 250

--min-keyint, -i <integer>
Minimum GOP size. Scenecuts closer together than this are coded as I or P, not IDR.

Range of values: >=0 (0: auto)

--scenecut <integer>**, --no-scenecut**
How aggressively I-frames need to be inserted. The higher the threshold value, the more aggressive the I-frame placement. *--scenecut* 0 or *--no-scenecut* disables adaptive I frame placement. Default 40

--rc-lookahead <integer>
Number of frames for slice-type decision lookahead (a key determining factor for encoder latency). The longer the lookahead buffer the more accurate scenecut decisions will be, and the more effective cuTree will be at improving adaptive quant. Having a lookahead larger than the max keyframe interval is not helpful. Default 20

Range of values: Between the maximum consecutive bframe count (*--bframes*) and 250

--b-adapt <integer>
Adaptive B frame scheduling. Default 2

Values: 0:none; 1:fast; 2:full(trellis)

--bframes, -b <0..16>
Maximum number of consecutive b-frames. Use *--bframes* 0 to force all P/I low-latency encodes. Default 4. This parameter has a quadratic effect on the amount of memory allocated and the amount of work performed by the full trellis version of *--b-adapt* lookahead.

--bframe-bias <integer>
Bias towards B frames in slicetype decision. The higher the bias the more likely x265 is to use B frames. Can be any value between -20 and 100, but is typically between 10 and 30. Default 0

--b-pyramid, --no-b-pyramid
Use B-frames as references, when possible. Default enabled

2.8 Quality, rate control and rate distortion options

--bitrate <integer>
Enables single-pass ABR rate control. Specify the target bitrate in kbps. Default is 0 (CRF)

Range of values: An integer greater than 0

--crf <0..51.0>
Quality-controlled variable bitrate. CRF is the default rate control method; it does not try to reach any particular bitrate target, instead it tries to achieve a given uniform quality and the size of the bitstream is determined by the complexity of the source video. The higher the rate factor the higher the quantization and the lower the quality. Default rate factor is 28.0.

--max-crf <0..51.0>
Specify an upper limit to the rate factor which may be assigned to any given frame (ensuring a max QP). This is dangerous when CRF is used in combination with VBV as it may result in buffer underruns. Default disabled

--vbv-bufsize <integer>
Specify the size of the VBV buffer (kbits). Enables VBV in ABR mode. In CRF mode, *--vbv-maxrate* must also be specified. Default 0 (vbv disabled)

--vbv-maxrate <integer>
Maximum local bitrate (kbits/sec). Will be used only if vbv-bufsize is also non-zero. Both vbv-bufsize and vbv-maxrate are required to enable VBV in CRF mode. Default 0 (disabled)

--vbv-init <float>
Initial buffer occupancy. The portion of the decode buffer which must be full before the decoder will begin decoding. Determines absolute maximum frame size. Default 0.9

Range of values: 0 - 1.0

--qp, -q <integer>
Specify base quantization parameter for Constant QP rate control. Using this option enables Constant QP rate control. The specified QP is assigned to P slices. I and B slices are given QPs relative to P slices using param->rc.ipFactor and param->rc.pbFactor unless QP 0 is specified, in which case QP 0 is used for all slice types.

Note that QP 0 does not cause lossless encoding, it only disables quantization. A truly lossless option may be added in a later release. Default disabled (CRF)

Range of values: an integer from 0 to 51

--aq-mode <0|1|2>
Adaptive Quantization operating mode. Raise or lower per-block quantization based on complexity analysis of the source image. The more complex the block, the more quantization is used. This offsets the tendency of the encoder to spend too many bits on complex areas and not enough in flat areas.

0.disabled

1.AQ enabled **(default)**

2.AQ enabled with auto-variance

--aq-strength <float>
Adjust the strength of the adaptive quantization offsets. Setting *--aq-strength* to 0 disables AQ. Default 1.0.

Range of values: 0.0 to 3.0

--cutree, --no-cutree
Enable the use of lookahead's lowres motion vector fields to determine the amount of reuse of each block to tune adaptive quantization factors. CU blocks which are heavily reused as motion reference for later frames are given a lower QP (more bits) while CU blocks which are quickly changed and are not referenced are given less bits. This tends to improve detail in the backgrounds of video with less detail in areas of high motion. Default enabled

--cbqpoffs <integer>
Offset of Cb chroma QP from the luma QP selected by rate control. This is a general way to spend more or less bits on the chroma channel. Default 0

Range of values: -12 to 12

--crqpoffs <integer>
Offset of Cr chroma QP from the luma QP selected by rate control. This is a general way to spend more or less bits on the chroma channel. Default 0

Range of values: -12 to 12

--rd <0..6>
Level of RDO in mode decision. The higher the value, the more exhaustive the analysis and the more rate distortion optimization is used. The lower the value the faster the encode, the higher the value the smaller the bitstream (in general). Default 3

Range of values: 0: least .. 6: full RDO analysis

--signhide, --no-signhide
Hide sign bit of one coeff per TU (rdo). Default enabled

2.9 Loop filters

--lft, --no-lft
Toggle deblocking loop filter, default enabled

--sao, --no-sao
Toggle Sample Adaptive Offset loop filter, default enabled

--sao-lcu-bounds <0|1>
How to handle depencency with deblocking filter

0.right/bottom boundary areas skipped **(default)**

1.non-deblocked pixels are used

--sao-lcu-opt <0|1>
Frame level or block level optimization

0.SAO picture-based optimization (prevents frame parallelism, effectively causes *--frame-threads* 1)

1.SAO LCU-based optimization **(default)**

2.10 VUI (Video Usability Information) options

x265 emits a VUI with only the timing info by default. If the SAR is specified (or read from a Y4M header) it is also included. All other VUI fields must be manually specified.

--sar <integer|w:h>
Sample Aspect Ratio, the ratio of width to height of an individual sample (pixel). The user may supply the width and height explicitly or specify an integer from the predefined list of aspect ratios defined in the HEVC specification. Default undefined (not signaled)

1.1:1 (square)

2.12:11

3.10:11

4.16:11

5.40:33

6.24:11

7.20:11

8.32:11

9.80:33

10.18:11

11.15:11

12.64:33

13.160:99

14.4:3

15.3:2

16.2:1

--crop-rect <left,top,right,bottom>
Define the (overscan) region of the image that does not contain information because it was added to achieve certain resolution or aspect ratio. The decoder may be directed to crop away this region before displaying the images via the *--overscan* option. Default undefined (not signaled)

--overscan <show|crop>
Specify whether it is appropriate for the decoder to display or crop the overscan area. Default unspecified (not signaled)

--videoformat <integer|string>
Specify the source format of the original analog video prior to digitizing and encoding. Default undefined (not signaled)

0.component

1.pal

2.ntsc

3.secam

4.mac

5.undefined

--range <full|limited>
Specify output range of black level and range of luma and chroma signals. Default undefined (not signaled)

--colorprim <integer|string>
Specify color primitive to use when converting to RGB. Default undefined (not signaled)

1.bt709

2.undef

3.**reserved**

4.bt470m

5.bt470bg

6.smpte170m

7.smpte240m

8.film

9.bt2020

--transfer <integer|string>
Specify transfer characteristics. Default undefined (not signaled)

1.bt709

2.undef

3.**reserved**

4.bt470m

5.bt470bg

6.smpte170m

7.smpte240m

8.linear

9.log100

10.log316

11.iec61966-2-4

12.bt1361e

13.iec61966-2-1

14.bt2020-10

15.bt2020-12

--colormatrix <integer|string>
Specify color matrix setting i.e set the matrix coefficients used in deriving the luma and chroma. Default undefined (not signaled)

0.GBR

1.bt709

2.undef

3.**reserved**

4.fcc

5.bt470bg

6.smpte170m

7.smpte240m

8.YCgCo

9.bt2020nc

10.bt2020c

--chromalocs <0..5>
Specify chroma sample location for 4:2:0 inputs. Consult the HEVC specification for a description of these values. Default undefined (not signaled)

2.11 Bitstream options

--repeat-headers
If enabled, x265 will emit VPS, SPS, and PPS headers with every keyframe. This is intended for use when you do not have a container to keep the stream headers for you and you want keyframes to be random access points.

API ONLY

--aud, --no-aud
Emit an access unit delimiter NAL at the start of each slice access unit. If option:–*repeat-headers* is not enabled (indicating the user will be writing headers manually at the start of the stream) the very first AUD will be skipped since it cannot be placed at the start of the access unit, where it belongs. Default disabled

--hash <integer>
Emit decoded picture hash SEI, so the decoder may validate the reconstructed pictures and detect data loss. Also useful as a debug feature to validate the encoder state. Default None

1.MD5

2.CRC

3.Checksum

2.12 Debugging options

--recon, -r <filename>
Output file containing reconstructed images in display order. If the file extension is ".y4m" the file will contain

a YUV4MPEG2 stream header and frame headers. Otherwise it will be a raw YUV file in the encoder's internal bit depth.

CLI ONLY

--recon-depth <integer>
Bit-depth of output file. This value defaults to the internal bit depth and currently cannot to be modified.

CLI ONLY

CHAPTER 3

Threading

3.1 Thread Pool

x265 creates a pool of worker threads and shares this thread pool with all encoders within the same process (it is process global, aka a singleton). The number of threads within the thread pool is determined by the encoder which first allocates the pool, which by definition is the first encoder created within each process.

--threads specifies the number of threads the encoder will try to allocate for its thread pool. If the thread pool was already allocated this parameter is ignored. By default x265 allocated one thread per (hyperthreaded) CPU core in your system.

Work distribution is job based. Idle worker threads ask their parent pool object for jobs to perform. When no jobs are available, idle worker threads block and consume no CPU cycles.

Objects which desire to distribute work to worker threads are known as job providers (and they derive from the JobProvider class). When job providers have work they enqueue themselves into the pool's provider list (and dequeue themselves when they no longer have work). The thread pool has a method to **poke** awake a blocked idle thread, and job providers are recommended to call this method when they make new jobs available.

Worker jobs are not allowed to block except when abosultely necessary for data locking. If a job becomes blocked, the worker thread is expected to drop that job and go back to the pool and find more work.

Note: x265_cleanup() frees the process-global thread pool, allowing it to be reallocated if necessary, but only if no encoders are allocated at the time it is called.

3.2 Wavefront Parallel Processing

New with HEVC, Wavefront Parallel Processing allows each row of CTUs to be encoded in parallel, so long as each row stays at least two CTUs behind the row above it, to ensure the intra references and other data of the blocks above and above-right are available. WPP has almost no effect on the analysis and compression of each CTU and so it has a very small impact on compression efficiency relative to slices or tiles. The compression loss from WPP has been found to be less than 1% in most of our tests.

WPP has three effects which can impact efficiency. The first is the row starts must be signaled in the slice header, the second is each row must be padded to an even byte in length, and the third is the state of the entropy coder is transferred from the second CTU of each row to the first CTU of the row below it. In some conditions this transfer of state actually improves compression since the above-right state may have better locality than the end of the previous row.

Parabola Research have published an excellent HEVC animation which visualizes WPP very well. It even correctly visualizes some of WPPs key drawbacks, such as:

1. the low thread utilization at the start and end of each frame
2. a difficult block may stall the wave-front and it takes a while for the wave-front to recover.
3. 64x64 CTUs are big! there are much fewer rows than with H.264 and similar codecs

Because of these stall issues you rarely get the full parallelisation benefit one would expect from row threading. 30% to 50% of the theoretical perfect threading is typical.

In x265 WPP is enabled by default since it not only improves performance at encode but it also makes it possible for the decoder to be threaded.

If WPP is disabled by *--no-wpp* the frame will be encoded in scan order and the entropy overheads will be avoided. If frame threading is not disabled, the encoder will change the default frame thread count to be higher than if WPP was enabled. The exact formulas are described in the next section.

3.3 Frame Threading

Frame threading is the act of encoding multiple frames at the same time. It is a challenge because each frame will generally use one or more of the previously encoded frames as motion references and those frames may still be in the process of being encoded themselves.

Previous encoders such as x264 worked around this problem by limiting the motion search region within these reference frames to just one macroblock row below the coincident row being encoded. Thus a frame could be encoded at the same time as its reference frames so long as it stayed one row behind the encode progress of its references (glossing over a few details).

x265 has the same frame threading mechanism, but we generally have much less frame parallelism to exploit than x264 because of the size of our CTU rows. For instance, with 1080p video x264 has 68 16x16 macroblock rows available each frame while x265 only has 17 64x64 CTU rows.

The second extenuating circumstance is the loop filters. The pixels used for motion reference must be processed by the loop filters and the loop filters cannot run until a full row has been encoded, and it must run a full row behind the encode process so that the pixels below the row being filtered are available. When you add up all the row lags each frame ends up being 3 CTU rows behind its reference frames (the equivalent of 12 macroblock rows for x264)

The third extenuating circumstance is that when a frame being encoded becomes blocked by a reference frame row being available, that frame's wave-front becomes completely stalled and when the row becomes available again it can take quite some time for the wave to be restarted, if it ever does. This makes WPP many times less effective when frame parallelism is in use.

--merange can have a negative impact on frame parallelism. If the range is too large, more rows of CTU lag must be added to ensure those pixels are available in the reference frames. Similarly *--sao-lcu-opt* 0 will cause SAO to be performed over the entire picture at once (rather than being CTU based), which prevents any motion reference pixels from being available until the entire frame has been encoded, which prevents any real frame parallelism at all.

Note: Even though the merange is used to determine the amount of reference pixels that must be available in the reference frames, the actual motion search is not necessarily centered around the coincident block. The motion search is actually centered around the motion predictor, but the available pixel area (mvmin, mvmax) is determined by merange and the interpolation filter half-heights.

When frame threading is disabled, the entirety of all reference frames are always fully available (by definition) and thus the available pixel area is not restricted at all, and this can sometimes improve compression efficiency. Because of this, the output of encodes with frame parallelism disabled will not match the output of encodes with frame parallelism

enabled; but when enabled the number of frame threads should have no effect on the output bitstream except when using ABR or VBV rate control.

By default frame parallelism and WPP are enabled together. The number of frame threads used is auto-detected from the (hyperthreaded) CPU core count, but may be manually specified via *--frame-threads*

Cores	Frames
> 32	6
>= 16	5
>= 8	3
>= 4	2

If WPP is disabled, then the frame thread count defaults to **min(cpuCount, ctuRows / 2)**

Over-allocating frame threads can be very counter-productive. They each allocate a large amount of memory and because of the limited number of CTU rows and the reference lag, you generally get limited benefit from adding frame encoders beyond the auto-detected count, and often the extra frame encoders reduce performance.

Given these considerations, you can understand why the faster presets lower the max CTU size to 32x32 (making twice as many CTU rows available for WPP and for finer grained frame parallelism) and reduce *--merange*

Each frame encoder runs in its own thread (allocated separately from the worker pool). This frame thread has some pre-processing responsibilities and some post-processing responsibilities for each frame, but it spends the bulk of its time managing the wave-front processing by making CTU rows available to the worker threads when their dependencies are resolved. The frame encoder threads spend nearly all of their time blocked in one of 4 possible locations:

1. blocked, waiting for a frame to process
2. blocked on a reference frame, waiting for a CTU row of reconstructed and loop-filtered reference pixels to become available
3. blocked waiting for wave-front completion
4. blocked waiting for the main thread to consume an encoded frame

3.4 Lookahead

The lookahead module of x265 (the lowres pre-encode which determines scene cuts and slice types) uses the thread pool to distribute the lowres cost analysis to worker threads. It follows the same wave-front pattern as the main encoder except it works in reverse-scan order.

The function slicetypeDecide() itself may also be performed by a worker thread if your system has enough CPU cores to make this a beneficial trade-off, else it runs within the context of the thread which calls the x265_encoder_encode().