Application Description

Application: V-Nova LCEVC-XSA (FPGA software encoding accelerator)

Hyperscale video services like social and e-sports networks need to encode and serve vast numbers of streams to their users. The server infrastructure required to satisfy this demand is often the largest cost for these businesses.
V-Nova LCEVC-XSA "Xilinx Software Accelerator" running on Xilinx FPGA enables any service operating in a private/public cloud to radically transform their efficiency, reducing operating costs by up to 4x whilst improving the streaming quality-of-service for their users.
Furthermore, public or private clouds can deploy V-Nova LCEVC-XSA on Xilinx® AlveoTM Data Center accelerator cards as an ultra-dense encoding solution to offer significant quality and cost benefits to their video delivery customers.

V-Nova LCEVC-XSA employs the MPEG-5 Part 2 Low-complexity enhancement video coding standard, a unique video encoding approach that significantly enhances the quality and throughput of any standard codec such as AVC/H.264, HEVC, VP8, VP9 and AV1. When combined with a Xilinx FPGA, V-Nova's LCEVC-enhanced encoder provides the highest density encoding solution in the market enabling use cases such as live 4Kp60 encoding on a single card. Playback is supported on a broad range of devices since MPEG-5 LCEVC leverages the hardware decoding capabilities of the underlying codec already present.

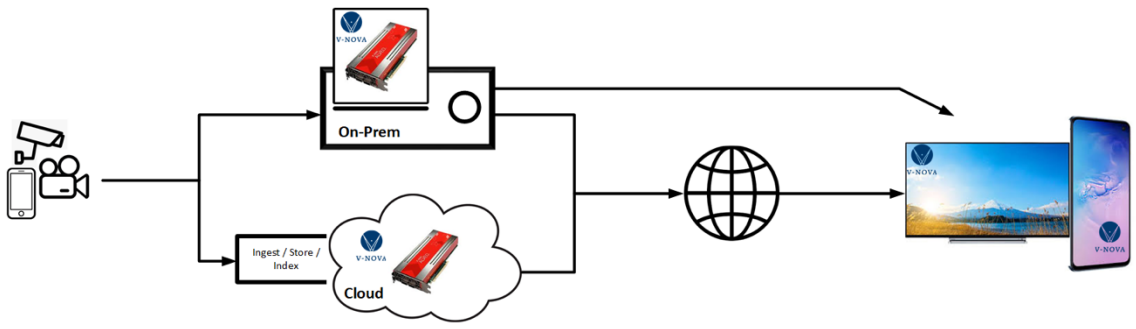


Figure 1: End–to–End system example

How does it work?

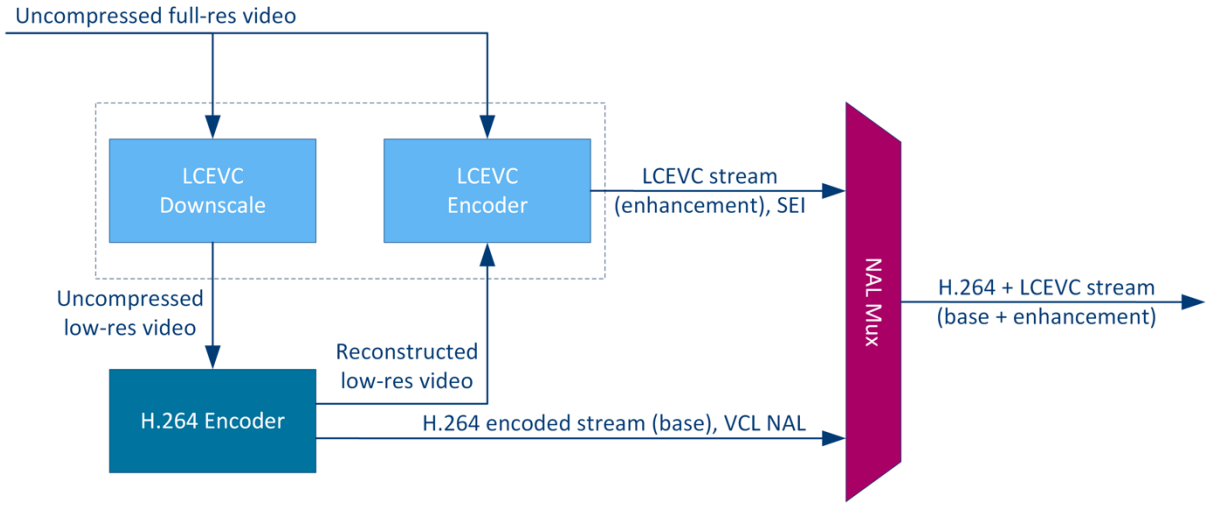


Figure 2: V-Nova LCEVC Encoder pipeline

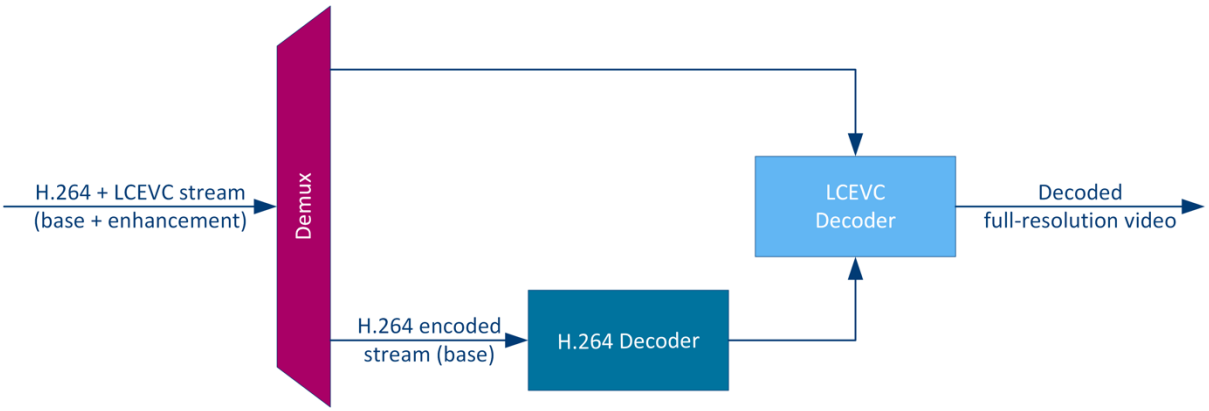


Figure 3: V-Nova LCEVC Decoder pipeline

The base encoder (e.g HEVC) is applied to a lower resolution version of the input video signal while LCEVC takes care of the additional details to reconstruct a full resolution picture.

On the consumer end, the base decoder (HEVC) can still provide a lower resolution video irrespective of whether the decoder has been LCEVC-enabled or not. The LCEVC decoder Plugin reconstructs the additional details to provide a full resolution decoded picture, thereby providing backwards compatibility with existing infrastructure.

LCEVC-XSA Container Feature Set

- Base encoder acceleration up to Live 4Kp60 (host platform dependent)
- AVC/H.264, HEVC, VP8, VP9 and AV1 base codec (V-Nova LCEVC-XSA)
- V-Nova LCEVC codec performance
- Enhanced Video Quality features
- ABR Ladder processing
- MPEG-5 LCEVC standard compliant
- FFmpeg integration

Key Target Markets & Applications

- HD to UHD migration (Backward compatible base)
- OTT, Films/VOD
- Live eSport streaming
- Video-Conferencing applications
- UHD / 4K cameras / VR
- High-Res DSNG feeds
- Security applications
- UHD, 4K, HD Asset encoding
- Playback servers
- Proxy editing feeds
- In-car entertainment systems
- UHD enabler over ADSL (IPTV Headends)

Supported platforms

Supported Xilinx Cards: Alveo U200, Alveo U250 (xdma-201830.2), Alveo U50 (xilinx_u50_gen3x16_xdma_201920_3)
Supported OS: Centos 7.x, Ubuntu 16.04 / 18.04

Supported Cloud Providers

Nimbix (TBC)
AWS (TBC)

Demo Version information

V-Nova LCEVC-XSA v3.1.0.x

Supported Resolutions

Width: 4096, 3840, 2560, 1920, 1280, 960, 640
Height: Any height divisible by 8 and greater than 64

Contact information

For more information or enquiries please contact us using the following email addresses below;

info@v-nova.com
sales@v-nova.com

Key Labels / Tag words

V-Nova, LCEVC, MPEG-5, Mpeg, Video, XSA, XDE, OTT, Transcode, Image Processing, Jpeg, h264.

Quick Start

This Docker image contains the environment and application to run the V-Nova LCEVC-XSA FPGA Encoder.

Minimum system requirements

-16GB RAM (32GB+ if input is placed in ramdisk)

-Intel Core i7-9700 or similar high-performance CPU

-CentOS 7.x

-Docker

-SSD with read speed of > 1GB/s for 4Kp60 encoding

-Xilinx XRT xrt-2.6.655-1.x86_64

-Alveo U250 (Xilinx XDMA xilinx-u250-xdma-201830.2-2580015.x86_64)

-Alveo U200 (Xilinx XDMA xilinx-u200-xdma-201830.2-2580015.x86_64)

-Alveo U50 (Xilinx XDMA xilinx_u50_gen3x16_xdma_201920_3)

Important host machine considerations:

-The performance of the base encoder is highly dependent on the processor and memory bandwidth of the host system.

-The performance of the encoder with file based inputs is highly dependent on the host disk read performance. Ensure the input file is on a ram-disk or SSD with a read speed of > 1GB/s

-Performing other CPU/Disk heavy tasks while encoding will impact performance

-Consider setting CPU scaling governor to performance mode for maximum encoder performance

-Consider disabling all tracker miners on the host system

Installation

-Sign-In / create an account on docker hub
-docker pull hubxilinx/vnova_lcevc_xsa_onprem_gpl_u50_20201:3.1.0.2 (*for the U50 platform*)
-docker pull hubxilinx/vnova_lcevc_xsa_onprem_gpl_u200_20201:3.1.0.1 (*for the U200 platform*)
-docker pull hubxilinx/vnova_lcevc_xsa_onprem_gpl_u250_20201:3.1.0.1 (*for the U250 platform*)

Licensing

A subscription license can be obtained from the [Xilinx Appstore] or going to [https://fpga.v-nova.com/]

Register and obtain a cred.json identifier. The path to this file must be passed to the encoder run command (see below).

Subscribe to a pricing plan of your choice

Running the encoder

Retrieve the xocl & xclmgmt device handles peculiar to your machine:

 # sudo xbutil scan (for xocl handle retrieval)

```
INFO: Found total 1 card(s), 1 are usable
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
System Configuration
OS name:  Linux
Release:  3.10.0-862.el7.x86_64
Version:  #1 SMP Fri Apr 20 16:44:24 UTC 2018
Machine:  x86_64
Model:          All Series
CPU cores:8
Memory:         31862 MB
Glibc:          2.17
Distribution:   CentOS Linux 7 (Core)
Now:            Thu Mar 25 20:46:20 2021
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XRT Information
Version:  2.6.655
Git Hash:  2d6bfe4ce91051d4e5b499d38fc493586dd4859a
Git Branch:     2020.1
Build Date:2020-05-22 19:05:52
```

XOCL: 2.6.655,2d6bfe4ce91051d4e5b499d38fc493586dd4859a
XCLMGMT: 2.6.655,2d6bfe4ce91051d4e5b499d38fc493586dd4859a

[0] 0000:01:00.1 xilinx_u50_gen3x16_xdma_201920_3 user(inst=129)

 # ls /dev/xclmgmt* (xclmgmt handle retrieval)

/dev/**xclmgmt256**

Encoding command line example

Two methods of command line modes are available

1. Basic mode
2. Advanced mode

Basic Mode: (Targeted for quick setup and trials with a pre-supplied input video)

"docker run ..... <codec> <width> <height> <bitrate> <use-lcevc> <loop>"
"codec options => [x264, x265, qsv_h264, qsv_hevc, vp8, vp9, av1]"
"e.g .docker run ..... x265 3840 2160 8000 1 -1" [loop input file indefinitely]
"e.g .docker run ..... x265 3840 2160 8000 1 4" [loop input file 5 times]

docker run -it --rm --device=/dev/xclmgmt256:/dev/xclmgmt256 --device=/dev/dri/renderD129:/dev/dri/renderD129 --device=/dev/dri/renderD128:/dev/dri/renderD128 -v *<path-to-ffmpeg-dir>***:/ffmpeg_dist:Z -v** *<path-to-input-output-dir>***:/io:Z -v** *<path-to-cred.json>***:/vnova_lcevc/cred.json:Z hubxilinx/vnova_lcevc_xsa_onprem_gpl_u50_20201:3.1.0.2 x265 3840 2160 8000 1 -1**

Advanced mode:

docker run -it --rm --device=/dev/xclmgmt256:/dev/xclmgmt256 --device=/dev/dri/renderD129:/dev/dri/renderD129 --device=/dev/dri/renderD128:/dev/dri/renderD128 -v *<path-to-ffmpeg-dir>***:/ffmpeg_dist:Z -v** *<path-to-input-output-dir>***:/io:Z -v** *<path-to-cred.json>***:/vnova_lcevc/cred.json:Z hubxilinx/vnova_lcevc_xsa_onprem_gpl_u50_20201:3.1.0.2 -y -s 3840x2160 -pix_fmt yuv420p -vcodec rawvideo -r 60 -stream_loop 1 -i /io/input.yuv -c:v lcevc_hevc -base_encoder x265 -eil_params "accel_type=xilinx_xma;encoding_transform_type=dd;api_mode=asynchronous;baseEncType=x265;encoding_upsample=stergy;rate_control_mode=adaptiveratio;rc_bucket_duration_ms=4000;rc_bitrate_max_base_prop=0.875;rc_bitrate_base_prop=0.875;encoding_downsample_luma=lanczos3;encoding_downsample_chroma=larea3;rc_max_pcrf=35;rc_pcrf_base_prop=0.8;rc_pcrf_gop_length=120;rc_pcrf_min_bitrate=2000 ;rc_pcrf_base_rc_mode=cbr;encoding_step_width_loq_1_min=32767;bframes=3;preset=medium;encoding_mode=enhanced;residual_mode_priority_enabled=0;temporal_sw_modifier=-2;temporal_use_priority_map=0;temporal_cq_sw_multiplier=0;temporal_use_new_cost=0;temporal_use_temporal_type=0;encoding_adaptive_deadzone_enabled=1;lcevc_tune=disabled" -b:v 8000k -vsync 0 "/io/lcevc_x265_fpga.ts"**

-Note: variables marked in **red** should **NOT** be modified

-This will encode a 4K video on the host system located at <*path-to-input-output-dir*>**/input.yuv** to generate an output bitstream located at <*path-to-input-output-dir*>**/lcevc_x265_fpga.ts**

-The above command line encodes at a bitrate of 8mbps (8000kbps) @ 60 frames per second.

Note: <*path-to-input-output-dir*> is intended to represent the location of a folder to store the input or/and output videos/bitstreams. It is recommended that this path be /dev/shm (i.e ramdisk)

Relevant encoding parameters

Docker specific:

docker run → standard command to run a Docker container
-it --rm → Run Docker in interactive mode and delete container after execution has completed
--device → Map a system device into the Docker container. These mappings give the Docker container access to the FPGA hardware.
-v host_path:container_path → Map a volume from the host filesystem into the Docker container (necessary for file input/output and providing the cred.json identifier for licensing)

FFmpeg Specific:

-y → Overwrite output files without asking
-s WxH → Resolution of input
-pix_fmt → Pixel format of input file (yuv420p)
-c:v <codec> → ffmpeg lcevc plugin used for encoding (options are, ***lcevc_h264, lcevc_hevc, lcevc_vp8, lcevc_vp9, lcevc_av1***, which specifies LCEVC with a base codec)
-base_encoder <codec> → host driver base encoder plugin (*options are* **x264, x265,qsv_h264, qsv_hevc, vpx, av1**)
-**baseEncType <codec>** → FPGA specific variable (*options are* **x264, x265, vpx, av1**) and must match codec specified above [e.g -c:v lcevc_h264 -base_encoder qsv_h264 -eil_params "baseEncType=x264"]. All three variables should be consistent in definition.
-stream_loop → Specifies the number of times to loop over the input file during encodes, set to -1 for infinite
-i → location of input file to be encoded
-b:v → Specifies the encoding bit rate, e.g 4500k will encode at 4.5mbps

LCEVC specific:

-eil_params "<params>" → Prefix option for defining LCEVC encoder specific parameters
preset → Target quality, options are veryslow, slow, medium, fast, veryfast, ultrafast [x264, x265 & QSV codecs only]
quality → Target quality, options are best, good, realtime [VP8, VP9, AV1 (only 'realtime' option is supported for AV1)]

FFMPEG distribution

FFmpeg is a collection of libraries and tools to process multimedia content such as audio, video, subtitles and related metadata. The FFmpeg distribution for LCEVC-XSA is GPL Licensed and the corresponding Linux binaries and libraries can be downloaded from <*contact V-Nova: fpga@v-nova.com*>.

Installing the decoder

Note: The sample decoder is based on Microsoft UWP (Universal Windows Player) and as such will **ONLY** run on a Windows 10 PC

1. Download and install the HEVC video extension app from the Microsoft store https://www.microsoft.com/en-us/store/p/hevc-video-extension/9n4wgh0z6vhq
2. Download and install the V-Nova decoder app from the Microsoft app store https://www.microsoft.com/store/productId/9N9C2Z21XHL9

Note: The decoder above **only** supports H.264 & HEVC based content, for VPx or AV1 LCEVC decoders please contact V-Nova on info@v-nova.com or sales@v-nova.com

LCEVC-XSA Performance characteristics

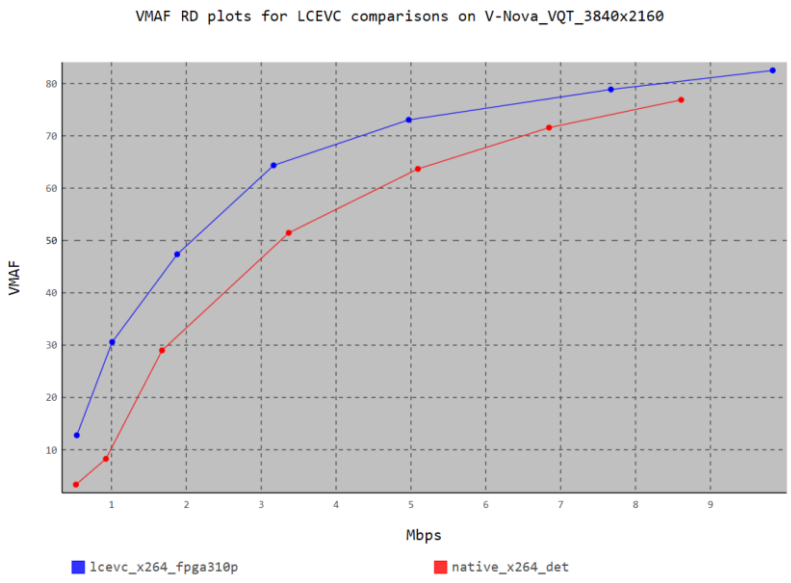
These performance results are derived on a host platform with the following configuration and can be used to verify the expected performance of the LCEVC-XSA v3.1.0.1 encoder.
Host processor => Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz
Ram => 32GB
OS => CentOS Linux release 7.5.1804 (Core)
Video => 3840 x 2160, yuv 8-bit 4:2:0 (ramdisk access)
LCEVC-XSA Version => v3.1.0.1

H.264 (libx264)

X264 native: medium preset, bitrate 8mbps, throughput 34 FPS

LCEVC-X264: medium preset, bitrate 8mbps, throughput 67 FPS (4K real-time encode)

LCEVC-X264 (pcrf mode): medium preset, bitrate 8mbps, throughput 77 FPS (4K real-time encode)

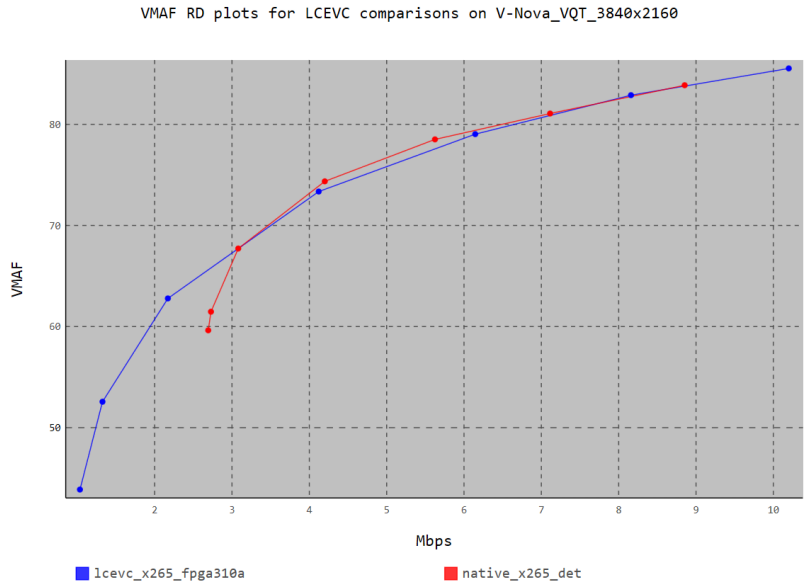


X264 V-MAF VQ comparison

HEVC (libx265)

X265 native: medium preset, bitrate 8mbps, throughput 11 FPS

LCEVC-X265: medium preset, bitrate 8mbps, throughput 34 FPS (4Kp30) [**Note**: LCEVC-XSA throughput limited by base codec throughput]

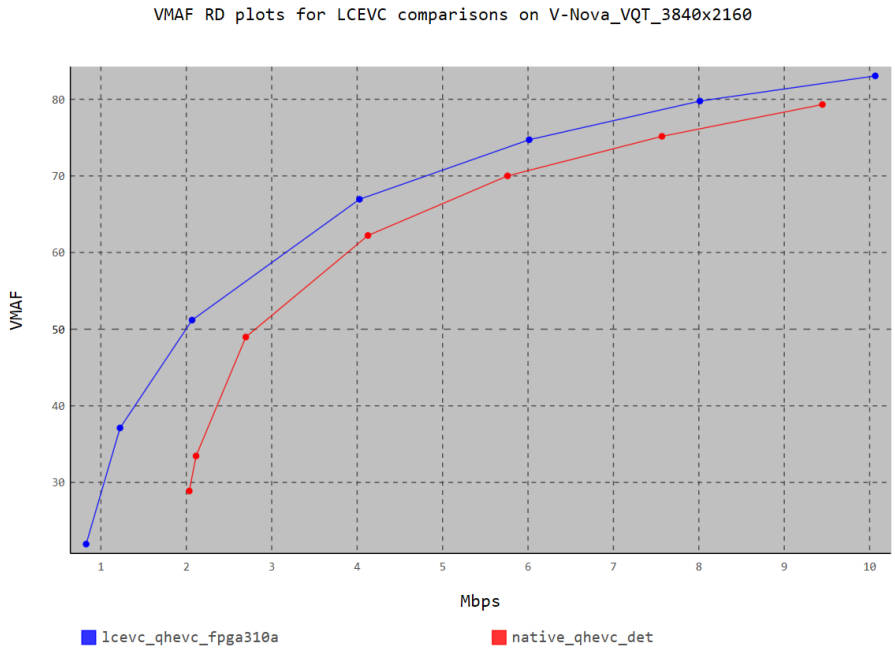


X265 V-MAF VQ comparison

QSV-HEVC

QSV HEVC native: Balanced [TargetQuality 4] preset, bitrate 8mbps, throughput 27 FPS

LCEVC-QHEVC: Balanced preset, bitrate 8mbps, throughput 77 FPS (4K real-time encode) [**Note**: LCEVC-XSA throughput limited by base codec throughput]

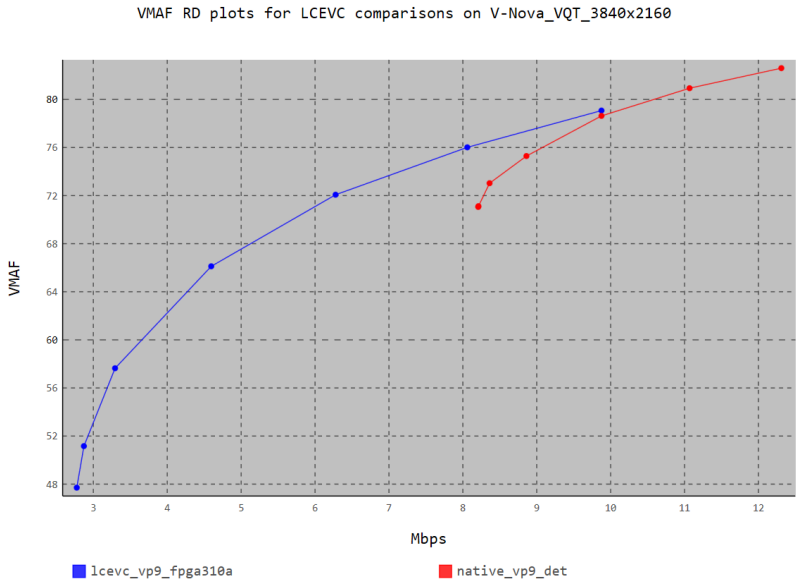


QSV-HEVC V-MAF VQ comparison

VP9 (libvpx)

VP9 native: realtime preset, bitrate 8mbps, throughput 36 FPS

LCEVC-VP9: realtime preset, bitrate 8mbps, throughput 54 FPS (4Kp50) [**Note**: LCEVC-XSA throughput limited by base codec throughput]



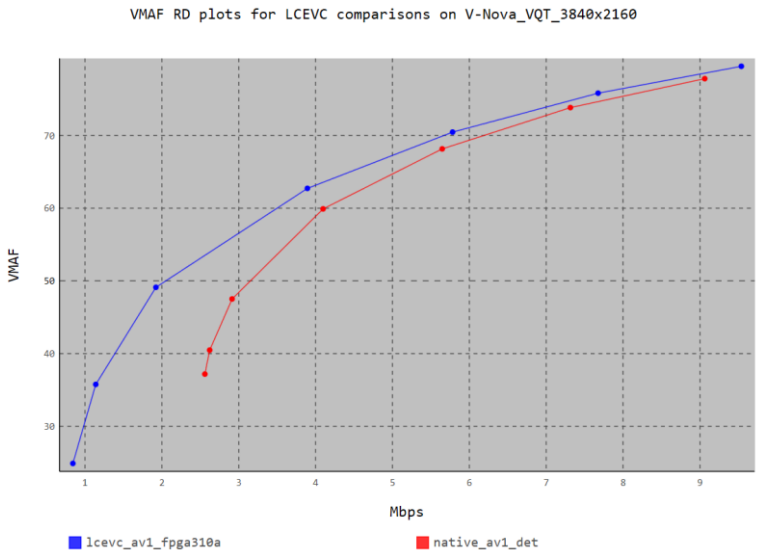
VP9 V-MAF VQ comparison

Note: Native VP9 [realtime preset] rate control unable to produce bitrates less than 8mbps
Note: ~25% bitrate savings @ VMAF=72

AV1 (libaom)

AV1 native: realtime preset, bitrate 8mbps, throughput 13 FPS

LCEVC-AV1: realtime preset, bitrate 8mbps, throughput 20 FPS [**Note**: LCEVC-XSA throughput limited by base codec throughput]



AV1 V-MAF VQ comparison