WAVE420L HEVC Encoder IP

Verification Guide

Version 1.8.14

WAVE420L HEVC Encoder IP: Verification Guide

Version 1.8.14

Revision History

Date	Revision	Change
2021/7/30	1.8.14	Release version

Proprietary Notice

Address and Phone Number

Chips&Media
NC Tower1, 7~8th FL, 509, Teheran-ro, Gangnam-gu, 125-880
Seoul, Korea
Tel: +82-2-568-3767
Fax: +82-2-568-3768

Homepage: *http://www.chipsnmedia.com*

Table of Contents

List of Figures

List of Tables

Preface

This preface introduces the WAVE420L Verification Guide and its reference documentation. It contains the following sections:

1. About this document

This document is the verification guide for WAVE420L Video Codec IP .

1.1. Intended Audience

This document has been written for experienced hardware engineers who want to test and verify functionalities of this IP product at an RTL level or software level.

1.2. Document Scope

This document describes IP simulation environment including test bench and script files for running the tests. It covers the directory structure of the comprehensive RTL simulation environment, testbench structure, test flow, and how to simulate with a simulation script called *runit*. Also, chapter 3 introduces the reference C-model of decoder and/or encoder which generates golden data to be compared with output data from RTL. Contents on the provided software test suite using the API are presented in chapter 4.

Note This is a standard document for WAVE4 Video IP Core. So parts of the document may include standard information irrelevant to your IP.

1.3. Typographical Conventions

The following typographical conventions are used in this document:

bold Highlights signal names within text, and interface elements such as menu names. May also be used for emphasis in descriptive lists where appropriate.

italic Highlights cross-references in blue, file names, and citations.

`typewriter` Denotes example source codes and dumped character or text.

2. Further Reading

This section lists the documents that are related to this IP product.

2.1. Other documents

- *WAVE420L Datasheet*
- *WAVE420L Programmer's guide*

Glossary

APB	Advanced Peripheral Bus, the ARM open standard for peripheral buses. APB is used for host processor to communication with VPU as their interface protocol.
AVC	Advanced Video Coding video standard known as H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC)
AXI	Advanced eXtensible Interface. The ARM open standard for high-performance
BPU	BIT Processor Unit composing of a highly optimized 16-bit fixed-point DSP for handling bitstream and controlling V-CORE to accelerate video codec processing.
BW	Bandwidth
BWB	It stands for Burst Write Back. BWB is a hardware module that collects write data and send 8 bursts of data to the external memory. It aims to increase bus efficiency by reducing a lot of short burst accesses that happen from VPU due to the nature of CTU-based processing.
WTL	It represents Write To Linear. It is an optional hardware module that allows a reconstructed output to be written in linear frame as well as in compressed frame (if WTL is disabled, VPU only writes compressed frame for less bandwidth). WTL writes linear frame data by using the BWB module.
C-model	It is an executable file of software codec written by C language to verify VPU's behavior and comparing the result between VPU hardware and C-model to see if they are bit-exact matched.
C&M	Chips&Media
DPB	Decoded Picture Buffer. A buffer holding decoded pictures for reference, output reordering, or output delay
FBC	Frame Buffer Compressor hardware block
FPGA	Field Programmable Gate Array
HEVC	The name of video standard which is a shorten form of abbreviates High Efficiency Video Coding
HOST PROCESSOR	Host CPU which gives video commands to VPU such as picture decoding or picture encoding through API or host interface registers
IP	Intellectual Property
LSB	Least Significant Bit
MSB	Most Significant Bit
RTL	Register Transfer Level
runit	The name of simulation script for NC-verilog
SoC	System on Chip
Testrunner	A script file listing encode/decode test commands with other inputs such as YUV/bitstream and parameters

VDI	VPU Device driver Interface
VLD	Variable Length Decoder
VPU	Video Processing Unit. It stands for the Chips&Media's video IP in this document.
VPUAPI	VPU API source and header files
V-CPU	V-CPU is a 32-bit processor that is the top layer of VPU hierarchical architecture. It is responsible for parsing bitstream syntax from sequence to slice header unit, controlling the underlying video hardware blocks called V-CORE.
VP9	An open and royalty free video coding format developed by Google.

Chapter 1
RTL SIMULATION ENVIRONMENT

Chips&Media provides customer with comprehensive simulation environment for VPU (Video Processing Unit) called a test bench as a belonging of deliverable package. Once you receive the package, you can check whether VPU successfully works simply by executing a testbench script file `runit` with bitstreams that you want to decode or with YUVs to encode. You can also use this testbench to see if the signals are connected rightly.

This chapter describes the RTL simulation environment using the bit-exact C model, brief verification flow, and test cases.

1.1. Verification Flow

Figure 1.1, "Verification Flow" is an illustration of how RTL simulation is performed.

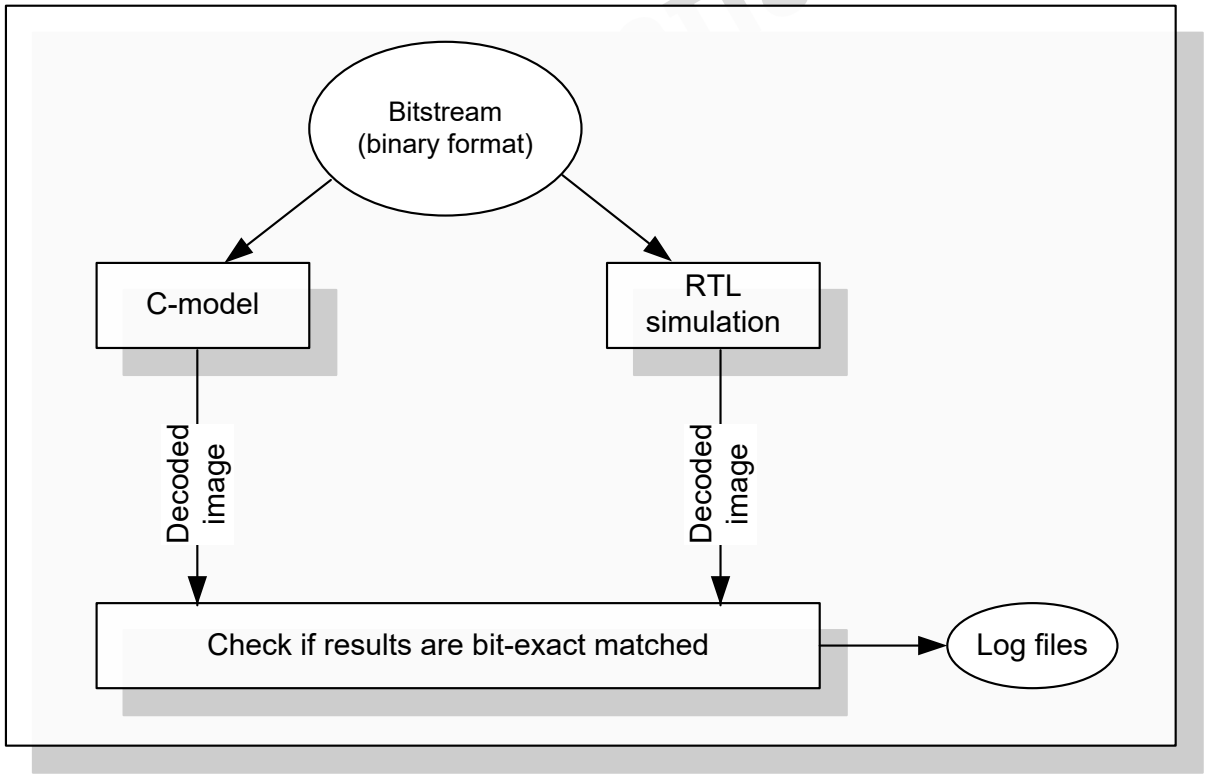

Figure 1.1. Verification Flow

In the decoding verification flow, the same bitstream file is used for both the bit-exact C model and RTL simulation environment as their input. After decoding user-specified number of frames, the decoded images from RTL are compared to the ones from C model in pixel-by-pixel.

In the same manner, a same YUV image and a configuration file are used for both the bit-exact C model and RTL simulation environment in the encoding verification flow. After encoding user-specified number of frames, the encoded stream from RTL simulation and bit-exact C model are compared in bit-by-bit.

Optionally, RTL simulation environment can generate trace files as intermediate data from the VPU internal blocks such as entropy coder, intra prediction, motion-compensative prediction, and so forth.

The formats of those trace files are the same as that from the bit-exact model. Thus, those files can also be used to check if the results of each decoding/encoding phase from both RTL and C-model sides are exactly matched. You can disable this function if you want to save simulation time.

1.2. Directory Structure

Figure 1.2, "Directory Structure" describes the directory structure for the simulation environment.

```
<install_dir>/
|
| - design/
|     | - firmware/                 ; Firmware
|     | - ref_c/
|     |     | - bin/                ; Executable model
|     |
|     | - rtl_v/                    ; Verilog RTL
|     |
| - sim/
|     | - listfile/                 ; List of verilog files and their paths for simulation
|     | - test_case_v/              ; Host processor behavioral model and verilog tasks
|     | - testbench_v/              ; Directory for simulation model
|     | - vectors/                  ; Simulation run directories
|     |
| - syn/                            ; Synthesis script
```

Figure 1.2. Directory Structure

1.3. Structure of Test Environment

Figure 1.3, "Testbench Structure" shows the structure of Testbench, the RTL simulation environment.

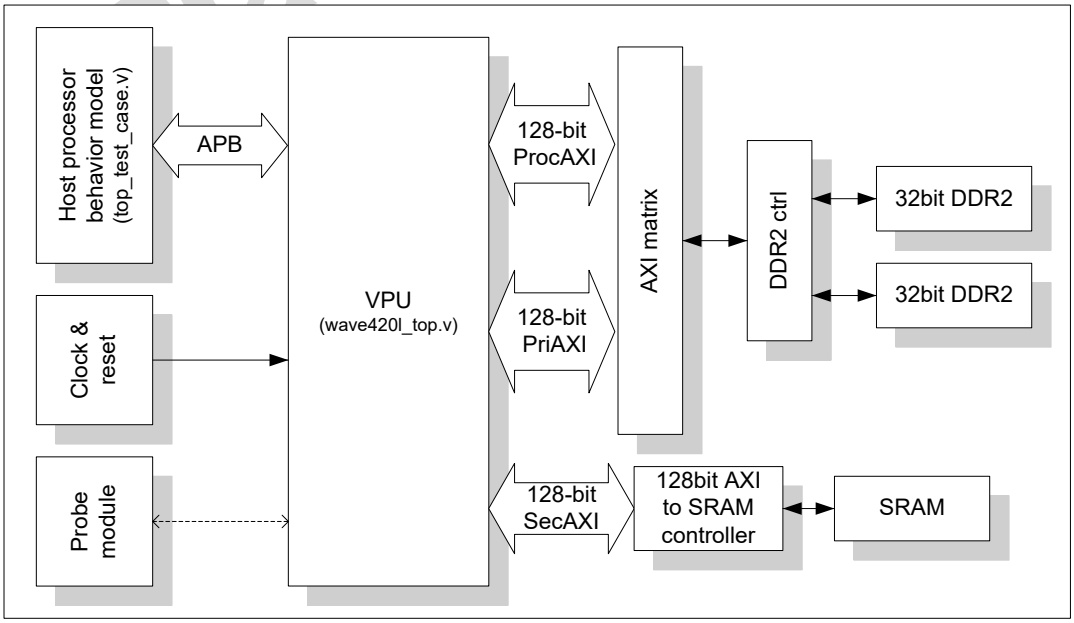

Figure 1.3. Testbench Structure

Testbench is mainly composed of three parts: VPU, host behavioral model, and memory controller model. In the testbech, VPU is connected to the host model through APB, DDR2 controllers through AXI, and some other modules such as Clock & Reset generator model and probe module. VPU runs with commands and parameters that come from host model for simulation. While decoding or encoding VPU accesses two 32-bit DDR2s via the C&M 128-bit AXI DDR2 controller model in order to read or write temporal or output data. The probe module catches intermediate data during video processing and generates trace files.

1.4. Host Behavioral Model

The behavioral model of host processor (top_test_case.v) is responsible for the followings.

- Downloading the firmware image for V-CPU and BPU embedded in VPU
- Setting global static parameters to VPU such as code, work, param buffers, use of secondary AXI, etc.
- Commanding VPU to initialize
- Commanding VPU to decode/encode a frame
- Getting result from VPU through access to host interface register

Figure 1.4, "Basic Encoding Flow" and *Figure 1.5, "Basic Decoding Flow"* illustrate what host behavioral model does for encoding and decoding in sequential way.

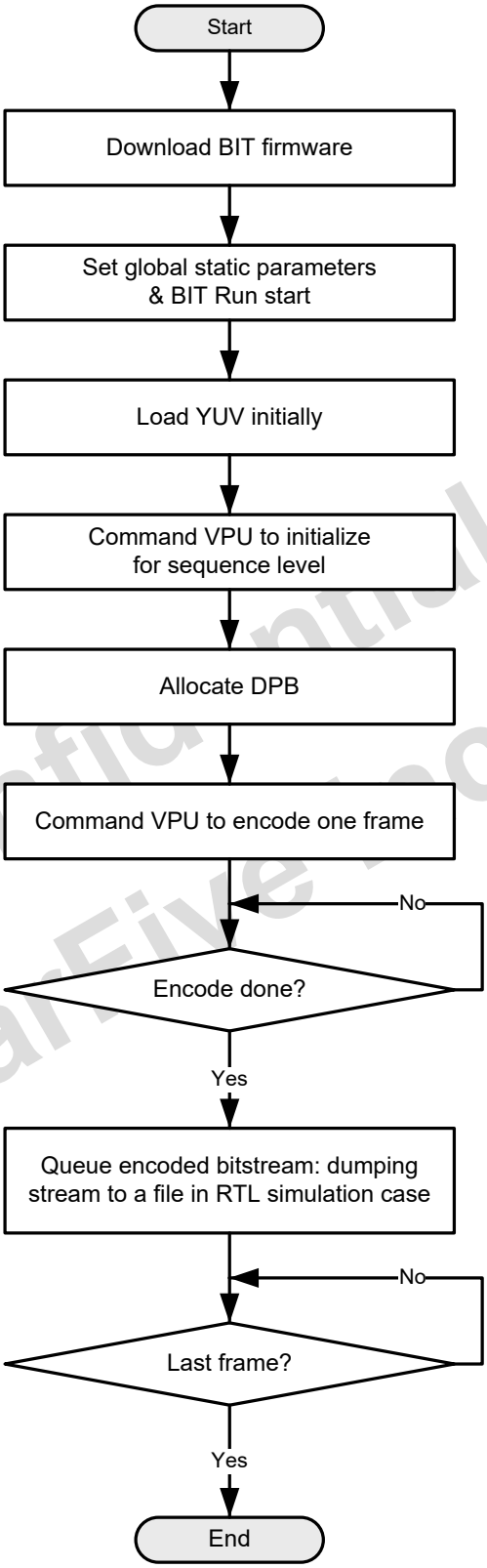

Figure 1.4. Basic Encoding Flow

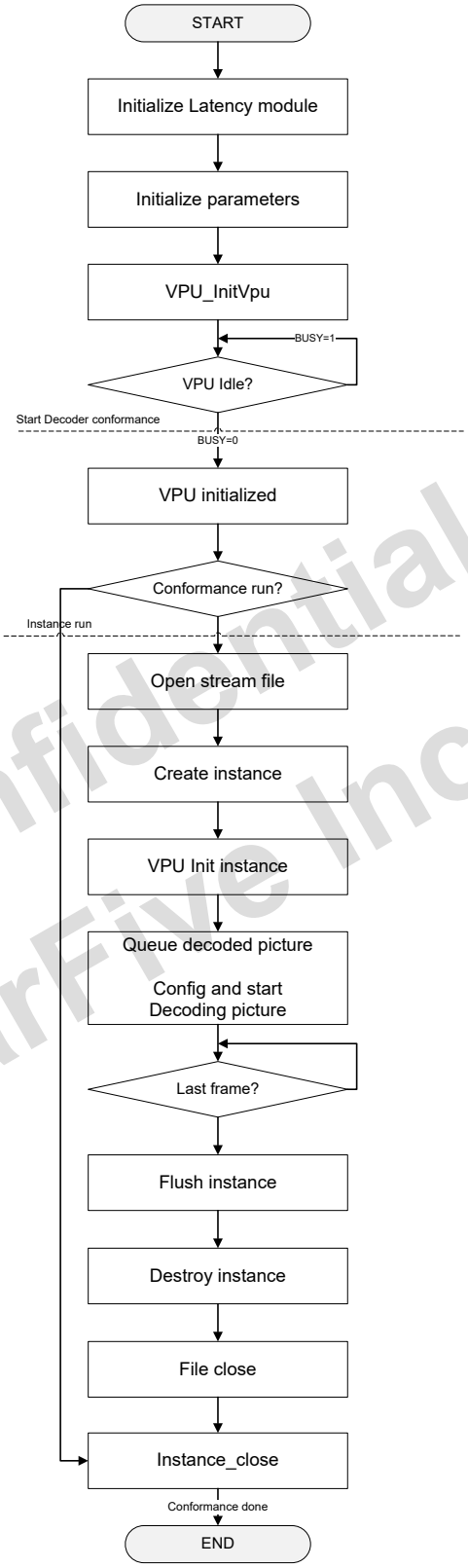

Figure 1.5. Basic Decoding Flow

Chapter 2
HOW TO RUN RTL SIMULATION

Before integrating VPU into target SoC, you need to check if VPU works well. With the simulation environment we provided in the deliverable package, you can simply execute a *runit_enc.pl* script file for RTL simulation with useful options. This chapter explains how to run the script with some run options.

2.1. Encoder

2.1.1. Specifying cfg

In the sim/vectors/codec directory, there is a *cfg_tt.f* that lists file names of cfgs (configuration files) to be tested. Once you put your test cfg into the <install_dir>/stream/ or somewhere else you want, you need to specify a right path for the cfg in the cfg_tt.f as below.

```
../../../cfg/hevc/aimain.cfg 1
../../../cfg/hevc/ldmain.cfg 1
```

You can create another cfg list file like *cfg_hevc.f* with readiness of the cfgs and their path designations according to your flavor.

2.1.2. runit Script

Figure 2.1, "Runit" shows the execution flow of runit script for RTL simulation.

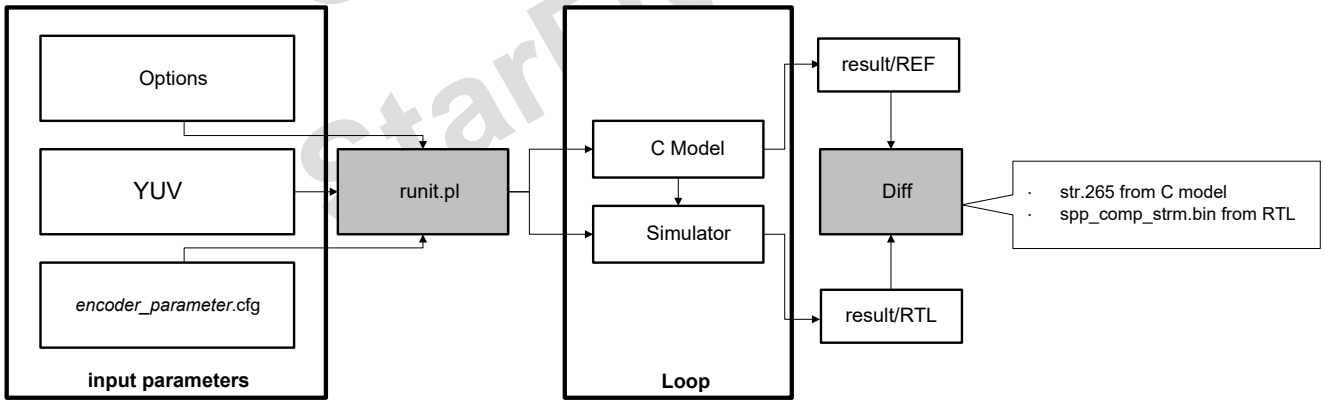

Figure 2.1. Runit

The simulation script for VCS, runit, is found in the <install_dir>/sim/vectors/codec` directory. You can enter `runit_enc` with the following available options for the script:

2.1.3. runit Options

- -f <filename>

  The script file reads the given .f file following -f option and simulates encoding with bitstream/cfg files listed inside. Please refer to the cfg_basic.f existing in the directory for a reference. If a line begins with #, the line

is ignored. So, users can write a comment by adding the character at the beginning of the line in the list file. There could be spaces between lines in the file.

```
Example:
   > cat cfg_basic.f
         #aimain.cfg 1
         ipp.cfg 1
         #ldmain.cfg 1
         #lpmain.cfg 1
         #ramain.cfg 1
   > runit_enc.pl -f cfg_basic.f
```

Note 1 after the .cfg file indicates an encoder test while 0 after test.265 is a decoder test.

In the above example, the script executes the simulation only for ipp.cfg.

- -m <frame number>

  The testbench encodes the specified number of frames with each cfg in the list file.

```
Example:
   > runit_enc.pl -f cfg_basic.f -m 5
```

The scripts encodes 5 frames for each cfg in the list file

- -w <frame number>

  The testbench dumps from the specified frame number and makes a waveform file in the FSDB format.

```
Example:
   > runit_enc.pl -f cfg_basic.f -m 5 -w 4
```

The testbench writes a waveform file after encoding 4 frames. If there are multiple files in the cfg_basic.f, waveform file for the last cfg file is stored. The scope for dumping waveform is the entire design. So, if users want to adjust the scope, they have to modify the <install_dir>/sim/test_case_v/top_test_case.v

- - progendian <endianness mode>

  This option sets an endianness mode of firmware code. VPU supports 16 endianness modes for frame buffer as the below.
  – 0 : f-e-d-c-b-a-9-8-7-6-5-4-3-2-1-0 (128-bit little endian)
  – 1 : e-f-d-c-a-b-8-9-6-7-4-5-2-3-0-1
  – 2 : d-c-f-e-9-8-b-a-5-4-7-6-1-0-3-2
  – 3 : c-d-e-f-8-9-a-b-4-5-6-7-0-1-2-3
  – 4 : b-a-9-8-f-e-d-c-3-2-1-0-7-6-5-4
  – 5 : a-b-8-9-e-f-c-d-2-3-0-1-6-7-4-5
  – 6 : 9-8-b-a-d-c-f-e-1-0-3-2-5-4-7-6
  – 7 : 8-9-a-b-c-d-e-f-0-1-2-3-4-5-6-7
  – 8 : 7-6-5-4-3-2-1-0-f-e-d-c-b-a-9-8
  – 9 : 6-7-4-5-2-3-0-1-e-f-c-d-a-b-8-9
  – 10 : 4-5-7-6-1-0-3-2-d-c-f-e-9-8-b-a
  – 11 : 5-4-6-7-0-1-2-3-c-d-e-f-8-9-a-b
  – 12 : 3-2-1-0-7-6-5-4-b-a-9-8-f-e-d-c
  – 13 : 2-3-0-1-6-7-4-5-a-b-8-9-e-f-c-d
  – 14 : 1-0-3-2-5-4-7-6-9-8-b-a-d-c-f-e
  – 15 : 0-1-2-3-4-5-6-7-8-9-a-b-c-d-e-f (128-bit Big Endian)

- -streamendian <endianness mode>

  This option sets an endianness mode of bitstream.
  – 0 : f-e-d-c-b-a-9-8-7-6-5-4-3-2-1-0 (128-bit little endian)
  – 1 : e-f-d-c-a-b-8-9-6-7-4-5-2-3-0-1
  – 2 : d-c-f-e-9-8-b-a-5-4-7-6-1-0-3-2

  - 3 : c-d-e-f-8-9-a-b-4-5-6-7-0-1-2-3
  - 4 : b-a-9-8-f-e-d-c-3-2-1-0-7-6-5-4
  - 5 : a-b-8-9-e-f-c-d-2-3-0-1-6-7-4-5
  - 6 : 9-8-b-a-d-c-f-e-1-0-3-2-5-4-7-6
  - 7 : 8-9-a-b-c-d-e-f-0-1-2-3-4-5-6-7
  - 8 : 7-6-5-4-3-2-1-0-f-e-d-c-b-a-9-8
  - 9 : 6-7-4-5-2-3-0-1-e-f-c-d-a-b-8-9
  - 10 : 4-5-7-6-1-0-3-2-d-c-f-e-9-8-b-a
  - 11 : 5-4-6-7-0-1-2-3-c-d-e-f-8-9-a-b
  - 12 : 3-2-1-0-7-6-5-4-b-a-9-8-f-e-d-c
  - 13 : 2-3-0-1-6-7-4-5-a-b-8-9-e-f-c-d
  - 14 : 1-0-3-2-5-4-7-6-9-8-b-a-d-c-f-e
  - 15 : 0-1-2-3-4-5-6-7-8-9-a-b-c-d-e-f (128-bit Big Endian)

- -sourceendian <endianness mode>

  This option sets an endianness mode of source YUV.
  - 0 : f-e-d-c-b-a-9-8-7-6-5-4-3-2-1-0 (128-bit little endian)
  - 1 : e-f-d-c-a-b-8-9-6-7-4-5-2-3-0-1
  - 2 : d-c-f-e-9-8-b-a-5-4-7-6-1-0-3-2
  - 3 : c-d-e-f-8-9-a-b-4-5-6-7-0-1-2-3
  - 4 : b-a-9-8-f-e-d-c-3-2-1-0-7-6-5-4
  - 5 : a-b-8-9-e-f-c-d-2-3-0-1-6-7-4-5
  - 6 : 9-8-b-a-d-c-f-e-1-0-3-2-5-4-7-6
  - 7 : 8-9-a-b-c-d-e-f-0-1-2-3-4-5-6-7
  - 8 : 7-6-5-4-3-2-1-0-f-e-d-c-b-a-9-8
  - 9 : 6-7-4-5-2-3-0-1-e-f-c-d-a-b-8-9
  - 10 : 4-5-7-6-1-0-3-2-d-c-f-e-9-8-b-a
  - 11 : 5-4-6-7-0-1-2-3-c-d-e-f-8-9-a-b
  - 12 : 3-2-1-0-7-6-5-4-b-a-9-8-f-e-d-c
  - 13 : 2-3-0-1-6-7-4-5-a-b-8-9-e-f-c-d
  - 14 : 1-0-3-2-5-4-7-6-9-8-b-a-d-c-f-e
  - 15 : 0-1-2-3-4-5-6-7-8-9-a-b-c-d-e-f (128-bit Big Endian)

- -frameformat <format>

  This option defines a frame format of input source YUV.
  - 0 : planar
  - 1 : sub-CTU map
  - 2 : NV12
  - 3 : NV21
  - 4 : packed mode (YUYV)
  - 5 : packed mode (YVYU)
  - 6 : packed mode (UYVY)
  - 7 : packed mode (VYUY)

- -pixelformat <format>

  This option defines a pixel format of input source frame.
  – 0 : 8-bit pixel

Caution | An error might occur if you give other than those above.

- -sec_axi_on

  This option enables to use all secondary AXI channels in VPU.

- -rpt <filename>

  The scripts writes a report file with the specified filename for each bitstream files in the list file specified with the option -f. Default name is ./result/sim.rpt. The script records conditions applied to the current simulation and whether the result from RTL simulation is bit-exactly matched to the result from the reference model for each bitstream.

- -log <filename>

  The VCS saves the log file to the specified filename. Default name is ./log/$cfg_name.log.

- -t 0/1/2

  This option enables bit-exact model and the RTL simulation to generate trace files for intermediate data.
  - 0 : generates only a trace file of bitstream.
  - 1 : generates trace files only for bitstream and compressed data from FBC.
  - 2 : generates all the trace files, which will take long and the size of those files might be large. (for debugging purpose)

- -multi_ins <number of instance>

  This option enables to simulate multi-instance operation. Up to 4 instances for encoder and 4 instances for decoder are allowed.

Note For other runit options related to encoding operation, please refer to the given .cfg file. The details are also described in the *Chapter 3, HEVC Encoder C-model*.

2.1.4. Getting Simulation Result

The testbench compares encoded streams from the reference model and the RTL. When the simulation script is running, simulation results written in the report file show whether each cfg in the cfg list file is successfully verified or not. Actually, the comparison between the reference model and the RTL is executed after encoding each frame. If there is any mismatch, the simulation stops.

2.1.5. Creating a New Test Case

The following are steps to create a new test case:

1. Move to the `<install_dir>/sim/vectors/` directory
2. Create a new directory.
3. Move to the new directory.
4. Copy all of the files in <install_dir>/sim/vectors/codec to your new directory.
5. Modify cfg_basic.f or create a new one that describes cfg files you want to test.
6. Execute the runit script with proper options.

Chapter 3
HEVC Encoder C-model

3.1. Overview

Reference software has been made to provide a reference implementation of Chips&Media HEVC hardware encoder. So, you can use this encoder model to evaluate the performance or quality with a variety of parameters or to verify the hardware by comparing the result between RTL and C-model. This chapter describes how to run the c-model and a list of available options.

3.2. Running the HEVC Encoder Model

The executable file is found in the design/enc_ref_c/hevc_enc/bin/ directory.

Usage:

hevc_enc [--help] [-c config.cfg] [--parameter=value]

Table 3.1. Description

Option	Description
--help	Prints parameter usage.
-c	Defines configuration file to use. Multiple configuration files may be used with repeated -c options.
--parameter=value	Assigns a value to a given parameter as further described below. Some parameters are also supported by shorthand -opt value.
-t ./dir --TLevel=1	Generates trace files and rate control report files for encoded frames in the designated directory. The RC report files such as trace_rc_report_mv_col.txt or trace_rc_report_dist_map.txt are created when the relevant option like EnReportMvCol or EnReportDistMap is on in the .cfg file.

The following sample configuration files are provided with the executable.

- aimain.cfg
- ramain.cfg
- ldmain.cfg
- lpmain.cfg

Note aimain, remain, ldmain, and lpmain have temporal structure like JCT-VC common test condition. See JCTVC-L1100 for further details.

3.3. Encoder Parameters

Running the encoder c-model, you should just leave the parameters as they are. It is because we found from our experiments that default setting in most cases showed the best result in terms of performance and encoded picture quality. For tuning encoder parameters or further guidelines to fit into your particular application or requirements, please feel free to contact engineers at our side.

Note Among these parameters, you see some in bold which cannot work with the default setting. You should give them a value as your intention.

Note There are some parameters that no longer exist - ForceIntraQP, KeepHierarchicalBit, EnHvsLambda, EnSeparateBitrate, and RateLayer0~7. You need to remove those parameters from .cfg files, since the current c-model cannot work with them.

Table 3.2. File, I/O, and Source Parameters

Option	Shorthand	Default	Description
InputFile	-i		Specifies the input video file. Video data must be in a raw 4:2:0 planar format (Y0CbCr). **Note** When the bit depth of samples is larger than 8, each sample is encoded in 2 bytes (little endian, LSB-justified).
BitstreamFile	-b		Specifies the output bitstream file.
ReconFile	-o		Specifies the output locally reconstructed video file. -o "" or -ReconFile "" does not generate recon video file.
SourceWidth	-w	0	Specifies the width of input video in luma samples.
SourceHeight	-h	0	Specifies the height of input video in luma samples.
InputBitDepth		8	Specifies the bit depth of input video.
InternalBitDepth		8	Specifies the bit depth used for coding. If this value is given, VPU encodes with it instead of InputBitDepth. For example, if InputBitDepth is 8 and InternalBitDepth is 10, VPU converts the 8-bit source frames into 10-bit ones and then encodes them. If this is not given, InputBitDepth is used for encoding.
FrameRate		0	Specifies the frame rate of input video. **Note** This option only affects the reported bit rates.
FrameSkip		0	Specifies the number of frame(s) to skip at the beginning of input video file.
FramesToBeEncoded	-f	-1 (all)	Specifies the number of frames to be encoded. A value of -1 implies all frames.
ConfWindSizeTop		0	Specifies the top size of output image to crop in pixel.
ConfWindSizeBot		0	Specifies the bottom size of output image to crop in pixel.
ConfWindSizeRight		0	Specifies the right size of output image to crop in pixel.
ConfWindSizeLeft		0	Specifies the left size of output image to crop in pixel.

Table 3.3. Coding Structure Parameters

Option	Shorthand	Default	Description
IntraPeriod		0	Specifies the intra frame period. intra frame period picture. 0: Only first I 1: All I 2: I,P,I,P,…

Option	Shorthand	Default	Description
			… In case that it is not low-delay encoding and decoding refresh type is IRAP(IDR or CRA), GOPSize(gop_size) and IntraPeriod should follow below requirements. (GopSize%IntraPeriod) == 0 or (GopSize%(IntraPeriod+1)) == 0
DecodingRefreshType		1	Specifies the decoding refresh type that are applied at IntraPeriod. 0: applies an I picture (not a clean random access point). 1: applies a non-IDR clean random access (CRA) point. 2: applies an IDR random access point.
GopPreset		0	Specifies the predefined GOP structure. 0: Custom GOP 1: I-I-I-I,..I (all intra, gop_size=1) 2: I-P-P-P,.. P (consecutive P, gop_size=1) 6: I-P-P-P-P, (consecutive P, gop_size=4)
GOPSize		1	Specifies the size of the cyclic GOP structure (1 ~ 8). It requires FrameN to be entered. In other words, you can set your own GOPSize and FrameN when not using GopPreset.
DeriveLambdaWeight		0	1: uses lambda_weight derived from VPU. @ 0: uses lambda_weight given in the configuration file. Please see *Appendix A, DeriveLambdaWeight*> for further details. **Note** This is only valid when rate control is off (constant qp).
FrameN			Multiple options that define the cyclic GOP structure that will be used repeatedly throughout the sequence. The table should contain GOPSize elements.

Table 3.4. Slice Coding Parameters

Option	Shorthand	Default	Description
IndeSliceMode		0	Defines an independent slice partitioning mode. 0: A single slice 1: Number of CTUs per slice
IndeSliceArg		0	Specifies the maximum number of CTUs in an independent slice depending on IndeSliceMode.
DeSliceMode		0	Defines a dependent slice mode. Dependency exists between slice segments. 0: A single slice segment 1: Number of CTUs per slice segment 2: Number of bytes per slice segment

Option	Shorthand	Default	Description
DeSliceArg		0	Specifies the maximum number of CTUs or bytes in a dependent slice depending on DeSliceMode.

Table 3.5. Quantization Parameters

Option	Shorthand	Default	Description
QP		30	Specifies the base value of the quantization parameter.
ScalingList		0	Enables or disables a pre-defined scaling list.

Table 3.6. Deblocking Filter Parameters

Option	Shorthand	Default	Description
EnDBK		1	Enables or disables the in-loop deblocking filter.
LFCrossSliceBoundaryFlag		1	Enables or disables the use of in-loop filtering across slice boundaries.
BetaOffsetDiv2		0	Specifies BetaOffsetDiv2 for deblocking filter.
TcOffsetDiv2		0	Specifies TcOffsetDiv2 for deblocking filter.

Table 3.7. Coding Tools Parameters

Option	Shorthand	Default	Description
ConstrainedIntraPred		0	Enables or disables constrained intra prediction. Constrained intra prediction only permits samples from intra blocks in the same slice as the current block to be used for intra prediction.
LosslessCoding		0	Enables or disables lossless coding. 0: disables lossless coding 1: enables lossless coding
EnCu8x8		1	Enables or disables CU8x8. **Note** Host should enable all CU sizes (CU08, CU16, CU32) due to hardware constraint. The RDO block determines the best CU size for CTU on its own.
EnCu16x16		1	Enables or disables CU16x16.
MaxNumMerge		2	Specifies maximum number of merge candidate.
EnDynMerge		1	Enables or disables dynamic merge mode.
EnTemporalMVP		1	Enables or disables temporal MVP.
EnIntraInInterSlice		1	Enables or disables intra CU in P or B slices.
IntraCtuRefreshMode		0	Specifies an intra CTU refresh mode. 0: No intra refresh 1: Row 2: Column 3: A step size in CTU
IntraCtuRefreshArg		0	Specifies an intra CTU refresh interval. Depending on IntraCtuRefreshMode it can mean one of the followings.

Option	Shorthand	Default	Description
			• The number of consecutive CTU rows for IntraCtuRefreshMode of 1 • The number of consecutive CTU columns for IntraCtuRefreshMode of 2 • A step size in CTU for IntraCtuRefreshMode of 3

Table 3.8. Rate Control Parameters

Option	Shorthand	Default	Description
RateControl	-R	0	Enables rate control. 0: disable rate control. 1: enable rate control.
EncBitrate	-E	0	Specifies encoding bitrate in bps. (0 ~ 700000000) For example, set 3000000 for 3 Mbps. This value is valid when RateControl is 1.
TransBitrate	-T	0	Specifies a peak transmission bitrate in bps. (0 ~ 700000000) • 0: ABR where transmission bitrate is so fast like in storage application • TransBitrate > EncBitrate: VBR • TransBitrate = EncBitrate: CBR • 0 < TransBitrate < EncBitrate: not available This value is valid when RateControl is 1. The higher TransBitrate is, the better quality you get.
InitialDelay	-D	3000	Specifies an initial CPB delay in msec. (10 ~ 3000) For example, 3000 should be set for 3 seconds. This value is valid when RateControl is 1. InitialDelay has relevance to picture quality and bitrate accuracy. As InitialDelay is shorter, encoder can reach target bitrate accurately with worse quality. On the other hand, as InitialDelay is longer, it can achieve rather better quality under rate control. This parameter is used for deriving an RC bitrate error by the following equation. max bitrate error = (InitialDelay / 1000) / (FramesToBeEncoded / FrameRate) * 100 (%) In other words, bitrate error decreases as InitialDelay decreases or FramesToBeEncoded increases. For example, when InitialDelay is 3000msec (default value) and FrameRate is 30, • If FramesToBeEncoded is 30 (=1sec), max bitrate error is 300%. • If FramesToBeEncoded is 300 (=10sec), max bitrate error is 30%.

Option	Shorthand	Default	Description
			• If FramesToBeEncoded is 3000 (=100sec), max bitrate error is 3%. • If FramesToBeEncoded is 30000 (=1000sec), max bitrate error is 0.3%.
BitAllocMode		0	Bit allocation mode 0: allocates bits of each picture of GOP adaptively according to temporal layer id. This value is valid when RateControl is 1 and GOP size is larger than 1.
CULevelRateControl	-C	0	Enables CU level rate control. 0: enables picture level rate control. 1: enables CU level rate control. This value is valid when RateControl is 1. If Roi is enabled (EnRoi is equal to 1), CU level rate control is turned off automatically. Enable this where bitrate error should be severely low like in broadcasting application. CULevelRateControl can do bitrate control better than PictureLevelRateControl, while CULevelRateControl can have a slight quality drop.
InitBufLevelx8		1	This parameter is used for determining an initial RC buffer level as the below. The RC refers to this buffer level for optimal bit allocation. Initial RC buffer level = buffer size * (InitBufLevelx8 / 8) buf_size = (trans_bps / 1000) * VbvBufferSize where, trans_bps = EncBitrate for ABR trans_bps = TransBitrate for VBR or CBR **Note** ▎Only available when RateControl is 1.
IntraQpOffset		0	An intra QP offset (-51 ~ 51) The lower this value is, the higher quality intra pictures get. This is valid when RateControl is enabled.
EnHvsQp		1	Enables or disables CU QP derivation based on CU variance . 0: disable 1: enable
HvsQpScaleDiv2		2	Specifies a scale for variance based CU QP derivation (0 ~ 4). The given HvsQpScaleDiv2 is divided by 2 and then applied to QP derivation. This value is valid when EnHvsQp is 1.

Option	Shorthand	Default	Description
MinQp		8	Specifies minimum CU QP (0 ~ 51). This value is valid when RateControl is 1.
MaxQp		51	Specifies maximum CU QP (0 ~ 51). This value is valid when RateControl is 1.
MaxDeltaQp		10	Specifies maximum delta QP of HVS QP (0 ~ 51). This value is valid when EnHvsQp is 1.
EnRoi		0	Enables ROI throughout the sequence level. This value is valid when RateControl is 1.
RoiFile			Specifies an ROI file name to be loaded. See *Table 3.12, "Elements of CTU ROI Map"* and *Section 3.5, "ROI/Mode/QP Map File Format"* for details on ROI file.
RoiDeltaQP		3	Specifies an ROI delta qp (1 ~ 51). This value is used for determining a QP for a specific ROI level as follows. QP(ROI_level) = QP(non-ROI) - (RoiDeltaQP * ROI_level) This value is valid when EnRoi is 1.
FixedBitRatioN		1	A fixed bit ratio (1 ~ 255) for each picture of GOP's bit allocation N = 0 ~ (MAX_GOP_SIZE - 1) MAX_GOP_SIZE = 8 For instance, when MAX_GOP_SIZE is 3, FixedBitRaio0 to FixedBitRaio2 can be set as 2, 1, and 1 respectively for the fixed bit ratio 2:1:1. This is only valid when BitAllocMode is 2.

Table 3.9. High Level Syntax Parameters

Option	Shorthand	Default	Description
EncAUD		0	Encodes AUD NAL unit in every picture.
EncEOS		0	Encodes EOS NAL unit at the end of sequence.
EncEOB		0	Encodes EOB NAL unit at the end of bitstream.
NumUnitsInTick		0	Specifies the number of time units of a clock operating at the time_scale frequency Hz. This value is valid when NumUnitsInTick or TimeScale is not equal to 0. TimeScale / NumUnitsInTick = frame_time for example, 27,000,000(time_scale) / 1,080,000 (num_units_in_tick) = 25fps (0.04 sec) 250,000,000(time_scale) / 3,333,333 (num_units_in_tick) = 75fps (0.013 sec)

Option	Shorthand	Default	Description
TimeScale		0	Specifies the number of time units that pass in one second. This value is valid when NumUnitsInTick or TimeScale is not equal to 0.
NumTicksPocDiffOne		0	Specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1. This value is valid when NumUnitsInTick or TimeScale is not equal to 0 and NumTicksPocDiffOne > 0.

Table 3.10. VUI Parameters

Option	Shorthand	Default	Description
VuiParamFlags		0	Specifies present flags of VUI syntaxs. [0] neutral_chroma_indication_flag syntax of VUI parameters [1] reserved [2] reserved [3] aspect_ratio_info_present_flag syntax of VUI parameters [4] overscan_info_present_flag syntax of VUI parameters [5] video_signal_type_present_flag syntax of VUI parameters [6] colour_description_present_flag syntax of VUI parameters [7] chroma_loc_info_present_flag syntax of VUI parameters [8] default_display_window_flag syntax of VUI parameters [9] bitstream_restriction_flag syntax of VUI parameters [31:10] reserved VUI parameters are encoded when VuiParamFlags is not equal to 0.
VuiAspectRatioIdc		0	aspect_ratio_idc of VUI syntaxs. This value is valid when VuiParamFlags[3] is equal to 1.
VuiSarSize		0	Sar width/height of VUI syntaxs [15:0] sar_width [31:16] sar_height This value is valid when VuiAspectRatioIdc is equal to 255.
VuiOverScanAppropriate		0	overscan_appropriate_flag of VUI syntaxs. This value is valid when VuiParamFlags[4] is equal to 1.
VideoSignal		0	Video signal parameters of VUI syntaxs. This value is valid when VuiParamFlags[5] is equal to 1. [2:0] video_format [3] video_full_range_flag [11:4] colour_primaries [19:12] transfer_characteristics [27:20] matrix_coeffs

Option	Shorthand	Default	Description
			[31:28] reserved
VuiChromaSampleLoc		0	chroma_sample_loc_type_top/bottom field of VUI syntaxs. This value is valid when VuiParamFlags[7] is equal to 1. [15:0] chroma_sample_loc_type_top_field [31:16] chroma_sample_loc_type_bottom_field
VuiDispWinLeftRight		0	def_disp_win_left/right_offsets of VUI syntaxs. This value is valid when VuiParamFlags[8] is equal to 1. [15:0] def_disp_win_left_offset [31:16] def_disp_win_right_offset
VuiDispWinTopBottom		0	def_disp_win_top/bottom_offsets of VUI syntaxs. This value is valid when VuiParamFlags[8] is equal to 1. [15:0] def_disp_win_top_offset [31:16] def_disp_win_bottom_offset

Table 3.11. SEI/HRD Parameters

Option	Shorthand	Default	Description
EnPrefixSeiData		0	It enables to encode the prefix SEI which is given by host.
PrefixSeiDataFile			It specifies the file name which contains the prefix sei nal data and start code.
PrefixSeiDataSize		1	The total byte size of the prefix SEI
PrefixSeiTimingFlag		0	A flag whether to encode PREFIX_SEI_DATA with a picture of this command or with a source picture of the buffer at the moment 0: encode PREFIX_SEI_DATA when a source picture is encoded. 1: encode PREFIX_SEI_DATA at this command.
EnSuffixSeiData		0	It enables to encode the suffix SEI which is given by host.
SuffixSeiDataFile			It specifies the file name which contains the suffix nal data and start code.
SuffixSeiDataSize		1	The total byte size of the suffix SEI
SuffixSeiTimingFlag		0	A flag whether to encode SUFFIX_SEI_DATA with a picture of this command or with a source picture of the buffer at the moment 0: encode SUFFIX_SEI_DATA when a source picture is encoded. 1: encode SUFFIX_SEI_DATA at this command.
EncodeRbspHrdInVui		0	A flag to encode the HRD syntax into VUI
EncodeRbspHrdInVps		0	A flag to encode the HRD syntax into VPS
RbspHrdFile			Specifies the file name to be loaded which contains the HRD syntax in rbsp.
RbspHrdSize		1	The bit size of the HRD rbsp data

Option	Shorthand	Default	Description
EncodeRbspVui		0	A flag to encode the VUI syntax in rbsp which is given by host
RbspVuiFile			Specifies the file name to be loaded which contains the VUI syntax in rbsp.
RbspVuiSize		1	The bit size of the VUI rbsp data

Table 3.12. Elements of CTU ROI Map

Elements	Description
CtuXStart	X start position of the #th CTU
CtuXEnd	X end position of the #th CTU
CtuYStart	Y start position of the #th CTU
CtuYEnd	Y end position of the #th CTU
ImportanceLevel	Importance level of the ROI (1 ~ 8) for the #th CTU. The larger value means the #th CTU (ROI) is more important than other CTUs.

3.4. GOP Structure Table

The GOP structure table defines a cyclic GOP structure that will be used repeatedly throughout the sequence. The table should contain GOPSize lines, named Frame1, Frame2, etc. The frames are listed in decoding order, so Frame1 is the first frame in the encoding order, Frame2 is the second and so on. Among other things, this table specifies reference pictures used by the current picture.

Note that some specified reference frames for pictures encoded in the very first GOP after an IDR frame might not be available. This is handled automatically by the encoder, so the reference pictures can be given in the GOP structure table as if there were infinitely many identical GOPs before the current one. Each line in the table contains the parameters used for the corresponding frame, separated by whitespace:

Table 3.13. Elements of GOP Structure Table

Element	Description
Type	Slice type (I or P)
POC	Display order of the frame within a GOP, ranging from 1 to GOPSize.
QPoffset	QP offset is added to the QP parameter to set the final QP value to use for this frame.
rambda_weight	Weight used during rate distortion optimization. Higher values mean lower quality and less bits. Typical range is between 0.3 and 1.
temporal_id	Temporal layer of the frame. A frame cannot predict from a frame with a higher temporal id. (0 ~ 6)
reference_L0	The POC of the reference picture L0
reference_L1	The POC of the reference picture L1

Element	Description	
	Note	reference_L1 can have the same POC as reference_L0 in B slice. But for compression efficiency it is recommended that reference_L1 has a different POC from reference_L0.

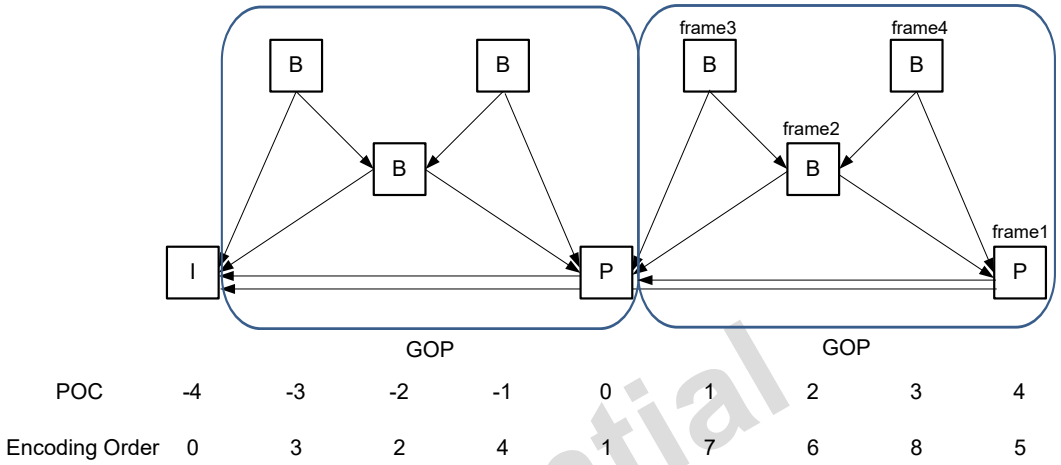

Figure 3.1. Example of Temporal Structure

3.5. ROI/Mode/QP Map File Format

The HEVC encoder model can get picture-based ROI , CTU mode, and CTU QP information from user in a certain file format as the following.

ROI File Format

```
FFFF
T (Picture index)
N (Number of ROIs in Tth picture)
CtuXStart CtuXEnd CtuYStart CtuYEnd ImportanceLevel (0th ROI information parameters)
...
CtuXStart CtuXEnd CtuYStart CtuYEnd ImportanceLevel (N-1th ROI information parameters)
FFFF
T+1 (picture index)
...
```

CTU Mode File Format

```
FFFF
T (Picture index)
N (Number of ROIs in Tth picture)
CtuXStart CtuXEnd CtuYStart CtuYEnd Mode (0th CTU mode information parameters)
...
CtuXStart CtuXEnd CtuYStart CtuYEnd Mode (N-1th CTU mode information parameters)
FFFF
T+1 (picture index)
...
```

CTU QP File Format

```
FFFF
T (Picture index)
N (Number of ROIs in Tth picture)
CtuXStart CtuXEnd CtuYStart CtuYEnd QP (0th CTU QP information parameters)
...
CtuXStart CtuXEnd CtuYStart CtuYEnd QP (N-1th CTU QP information parameters)
FFFF
T+1 (picture index)
...
```

The below is an example of ROI file in which two ROIs for 0th picture, nothing for 1st picture, four ROIs for 2nd picture are given.

Example of ROI file

```
FFFF
0
2
3 4 6 8 0
12 15 3 4 1
FFFF
1
0
FFFF
2
4
1 2 5 6 3
3 4 5 5 2
0 4 9 10 0
10 12 5 9 1
```

Chapter 4
SOFTWARE VERIFICATION ENVIRONMENT

4.1. Overview

This chapter describes software verification environment of VPU which is located in the reference software release package *cnm-wave420l-ref_sw_pkg-version.zip*. With executing the script file *Testrunner.sh* we provide on Linux bash shell, customers can easily do encode or decode tests on your SoC or FPGA board for checking conformance or verifying software.

Figure 4.1, "Conformance Test Flow" briefly shows how this test environment runs.

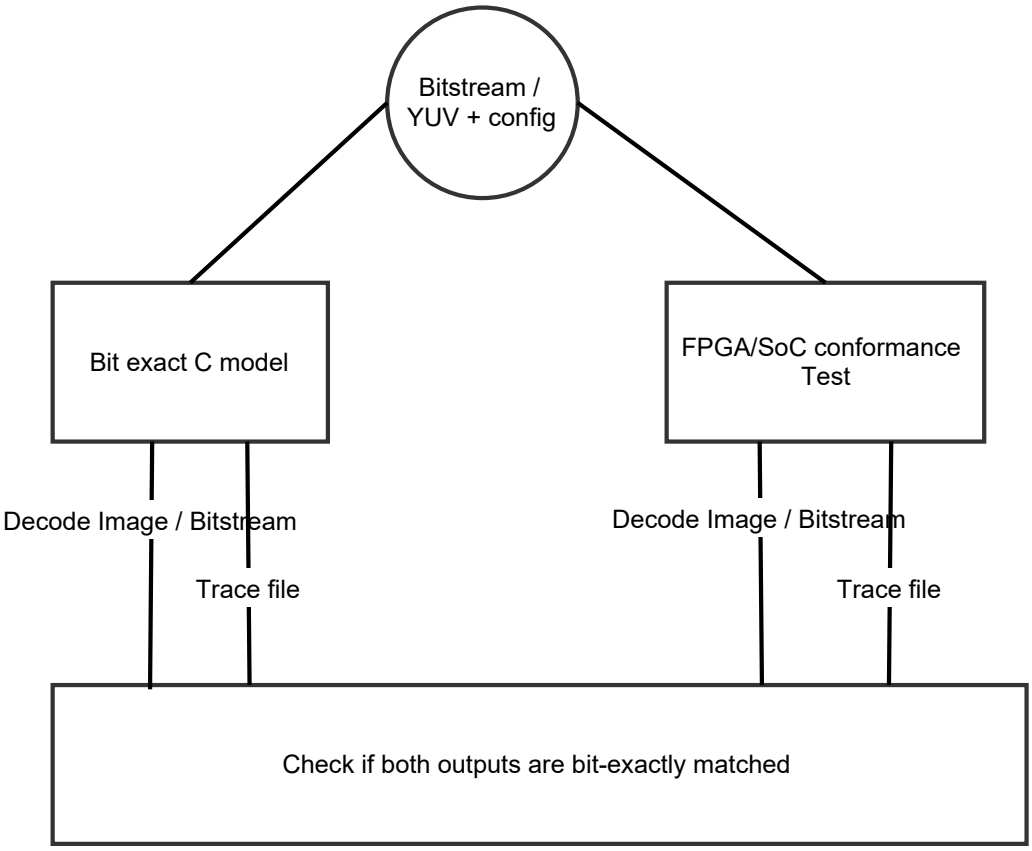

Figure 4.1. Conformance Test Flow

The Testrunner.sh script can execute both c-model and FPGA conformance test with YUVs and configuration files. Then it stores the results of executions at both sides and compares if they are bit-exactly matched. Once c-model have produced encoded bitstream, it will not anymore executed. The output of c-model is a kind of control

group so that you can see if there is any problem with something you have change or implemented by comparing it with output from your FPGA/SoC.

4.2. Directory Structure

Figure 4.2, "Source Tree of VPU API Reference Software" shows the directory structure and files that are associated with VPU software test.

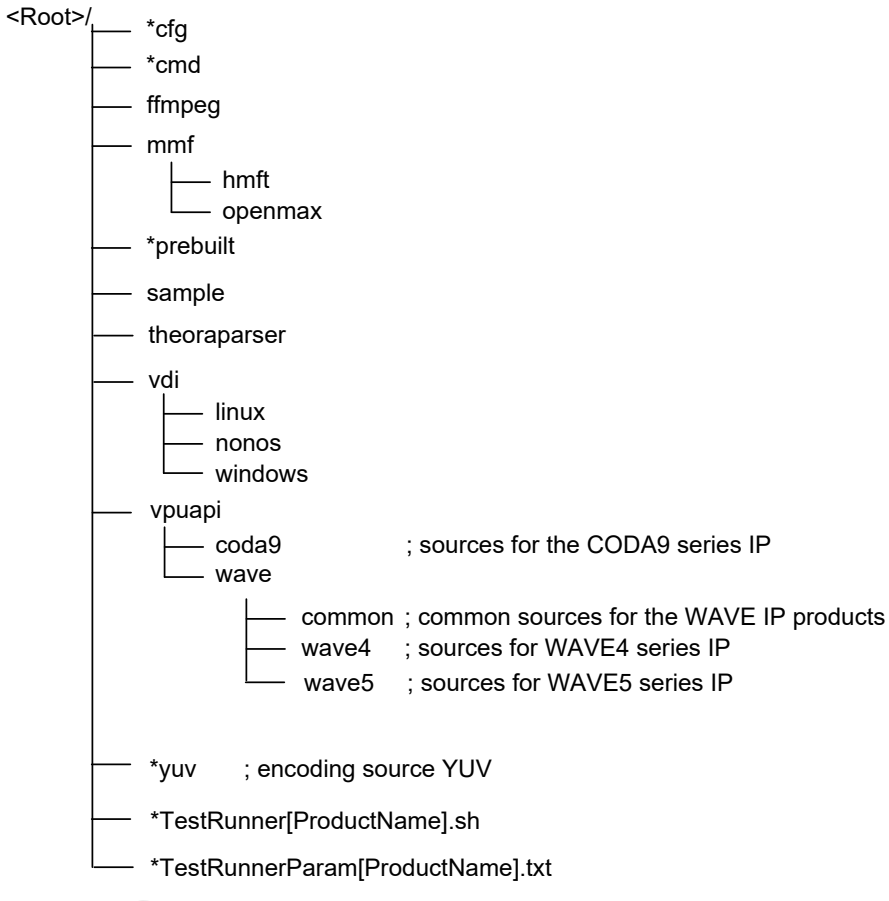

Figure 4.2. Source Tree of VPU API Reference Software

* represents folders that are required for conformance test in the above source tree. Please make sure that firmware bin (.bin) and the reference c-model binary file should be located in the root directory in order to execute the conformance test.

cfg
Encoder configuration files (.cfg). You can edit the path locating .cfg files with cfg_dir in Testrunner.sh.

cmd
For encoder .cmd files include a list of .cfg files, for example hevc_enc_icu08_01.cmd including icu08_001A.cfg and icu08_002A.cfg. You can edit the path locating .cfg files with cmd_dir in Testrunner.sh. For decoder, there is a list of test streams in the .cmd file.

ffmpeg
FFMPEG parser library for Linux and Windows

mmf
openmax codes

prebuilt
Encoded bitstreams as output file of reference c-model

sample
Application c sources & helper

stream
Source bitstream files to decode

theoraparser
Theora VLD parser module

vdi
VPU device driver interface

vpuapi
VPU API source and header files

yuv
Source YUV files for encoder while output YUV for decoder

TestRunnerWave420lEnc.sh
A script file listing all of the encode test commands with other test inputs such as YUV/bitstream and option.

TestRunnerWave420lDec.sh
A script file listing all of the decode test commands with other test inputs such as YUV/bitstream and option.

TestRunnerParamWave420lEnc.txt
Parameters for running TestRunnerWave420lEnc.sh such as clock frequencies and endianess.

TestRunnerParamWave420lDec.txt
Parameters for running TestRunnerWave420lDec.sh such as clock frequencies and endianess.

4.3. Testrunner.sh

4.3.1. Sequence of Testrunner.sh

Figure 4.3, "Testrunner" shows execution flow of Testrunner.sh script especially for comprehensive software-level encode and decode tests.

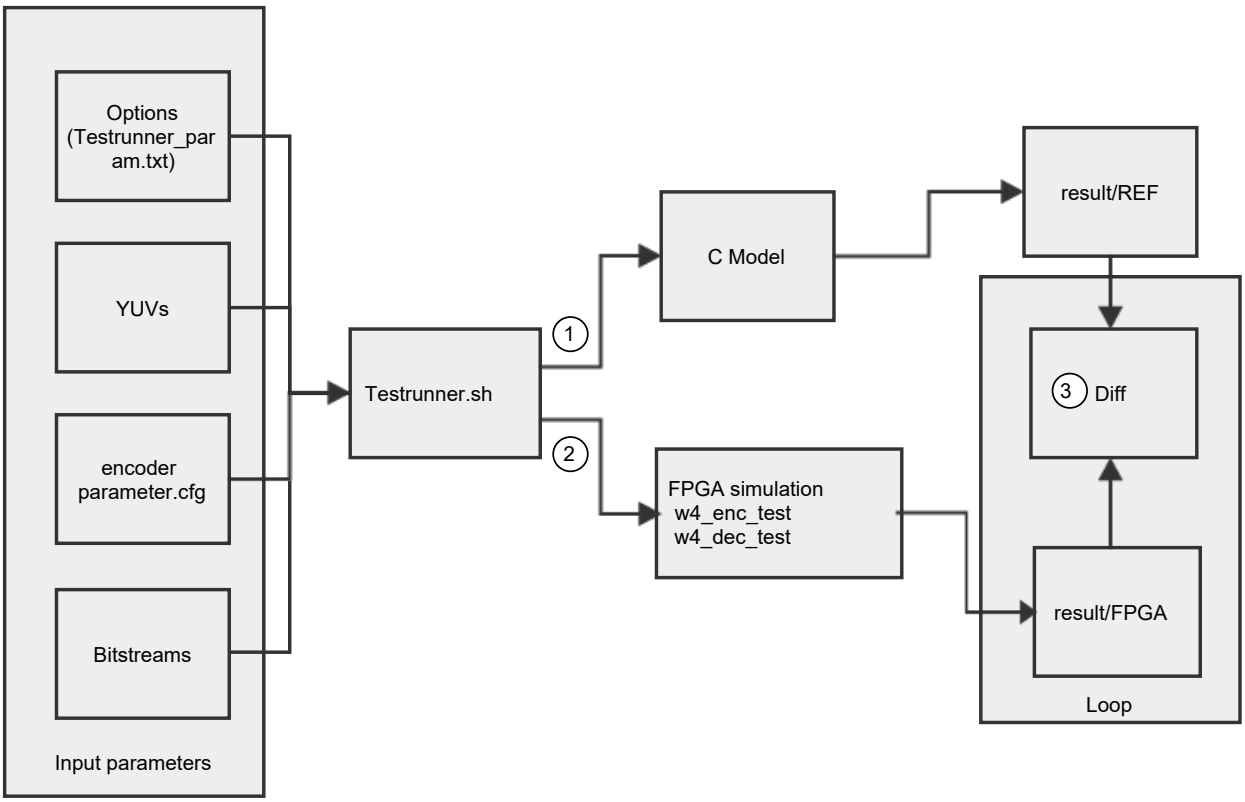

Figure 4.3. Testrunner

- If you execute TestrunnerWave420lEnc.sh, it uses YUV files and cmd files, other static or random options which are defined in TestrunnerParamWave420lEnc.txt, as its inputs for proceeding the predefined test sequence.
- If you execute TestrunnerWave420lDec.sh, it also reads bitstream files to test and other static or random options which are defined in TestrunnerParamWave420lDec.txt as its inputs.

Then the script performs tests as the following sequence.

1. Initially, ref-c is executed to generate prebuilt stream/YUV. The execution of ref-c is done only when there is no prebuilt stream/YUV.
2. FPGA test is proceeded through w4_enc_test or w4_dec_test.
3. Compare the prebuilt output of ref-c and the output of FPGA test and check whether they are bit-matched or not.

4.3.2. Running the Test Script

4.3.2.1. TestRunnerWave420Enc.sh

1. Load the device driver for VPU.

```
# cd vdi/linux/driver
# make
# sh load.sh
# cd ../../..
```

2. Build the makefile.

```
# make -f Wave420Enc.mak
```

3. Execute the TestRunnerWave420lEnc.sh.

```
# ./TestRunnerWave420lEnc.sh -c hevc --yuv-dir=yuv --ref-dir=prebuilt
  cmd/hevc_enc_test.cmd --build-stream --enable-random
```

Options

- -c : Type of codec
- --yuv-dir: The base directory of input yuv files
- --ref-dir: Location of prebuilt stream
- *.cmd: The given cmd filename i.e. example_enc.cmd which designates enc_test/test_case_001.cfg and enc_test/test_case_002.cfg to encode with
- --build-stream: An option to run ref-c on PC and save the bitstream (output file of ref-c)
- --enable-random: Random variables are used for test.

4.3.3. Parameters of Testrunner.sh

For random generation of options, please enter -99 to each option. These test options are used in individual test in the same manner, so if you want to see the list of options, refer to or *Table 4.1, “w4_enc_test Options”* .

4.4. Individual Test

4.4.1. w4_enc_test

You can test each w4_enc_test separately from TestRunnerWave420lEnc.sh by using the following options.

```
./w4_enc_test --secondary-axi=0
--stream-endian=0 --frame-endian=0
--enable-cbcrInterleave --nv21=1 --i422=1
--output=icu08_001A.cfg.bin --cfgFileName=../cfg/icu08_001A.cfg
--ref_stream_path=/nstream/qctool/work/ENC_REF_STREAM/wave420/ref_stream/icu08_001A.cfg_0.265
```

Table 4.1. w4_enc_test Options

W4_enc_test options	Description
-c	Enable to compare FPGA output and ref-c ouput 1: YUV compare 0: no compare
--secondary-axi	Bitmask to use secondary-axi For Multi-codec, 0x01: BIT-processor (Prediction) 0x02: IP/AC-DC 0x04: De-blocking filter (Luminance) 0x08: De-blocking filter (Chrominance) 0x32: ME For HEVC, 0x01: IMD (Intra mode decision)

W4_enc_test options	Description
	0x02: RDO 0x04: LF
--stream-endian	Stream endian 16 : f-e-d-c-b-a-9-8-7-6-5-4-3-2-1-0 (128-bit little endian) 17 : e-f-d-c-a-b-8-9-6-7-4-5-2-3-0-1 18 : d-c-f-e-9-8-b-a-5-4-7-6-1-0-3-2 19 : c-d-e-f-8-9-a-b-4-5-6-7-0-1-2-3 20 : b-a-9-8-f-e-d-c-3-2-1-0-7-6-5-4 21 : a-b-8-9-e-f-c-d-2-3-0-1-6-7-4-5 22 : 9-8-b-a-d-c-f-e-1-0-3-2-5-4-7-6 23 : 8-9-a-b-c-d-e-f-0-1-2-3-4-5-6-7 24 : 7-6-5-4-3-2-1-0-f-e-d-c-b-a-9-8 25 : 6-7-4-5-2-3-0-1-e-f-c-d-a-b-8-9 26 : 4-5-7-6-1-0-3-2-d-c-f-e-9-8-b-a 27 : 5-4-6-7-0-1-2-3-c-d-e-f-8-9-a-b 28 : 3-2-1-0-7-6-5-4-b-a-9-8-f-e-d-c 29 : 2-3-0-1-6-7-4-5-a-b-8-9-e-f-c-d 30 : 1-0-3-2-5-4-7-6-9-8-b-a-d-c-f-e 31 : 0-1-2-3-4-5-6-7-8-9-a-b-c-d-e-f
--frame-endian	Frame endian It is same as stream endian
--output	An ouput file path
--cfgFileName	A cfg file path
--ref_stream_path	A reference stream path
--enable-ringBuffer	Enable ring buffer mode (default: linebuffer mode)
--enable-cbcrInterleave	Enable cbcrInterleave(NV12) (default: off)
--nv21	Enable NV21 (default: off)
--src_format	A source type 0 : planar 1 : nv12 2 : nv21 3 : YUYV **Note** When you perform each test, not all of the tests by Testrunner.sh, this option is seen differently, for example, `--enable-cbcrInterleave --nv21=0` for 2 (nv12) and `--packedFormat=1` for 3 (YUYV).

Encoder Test Flow

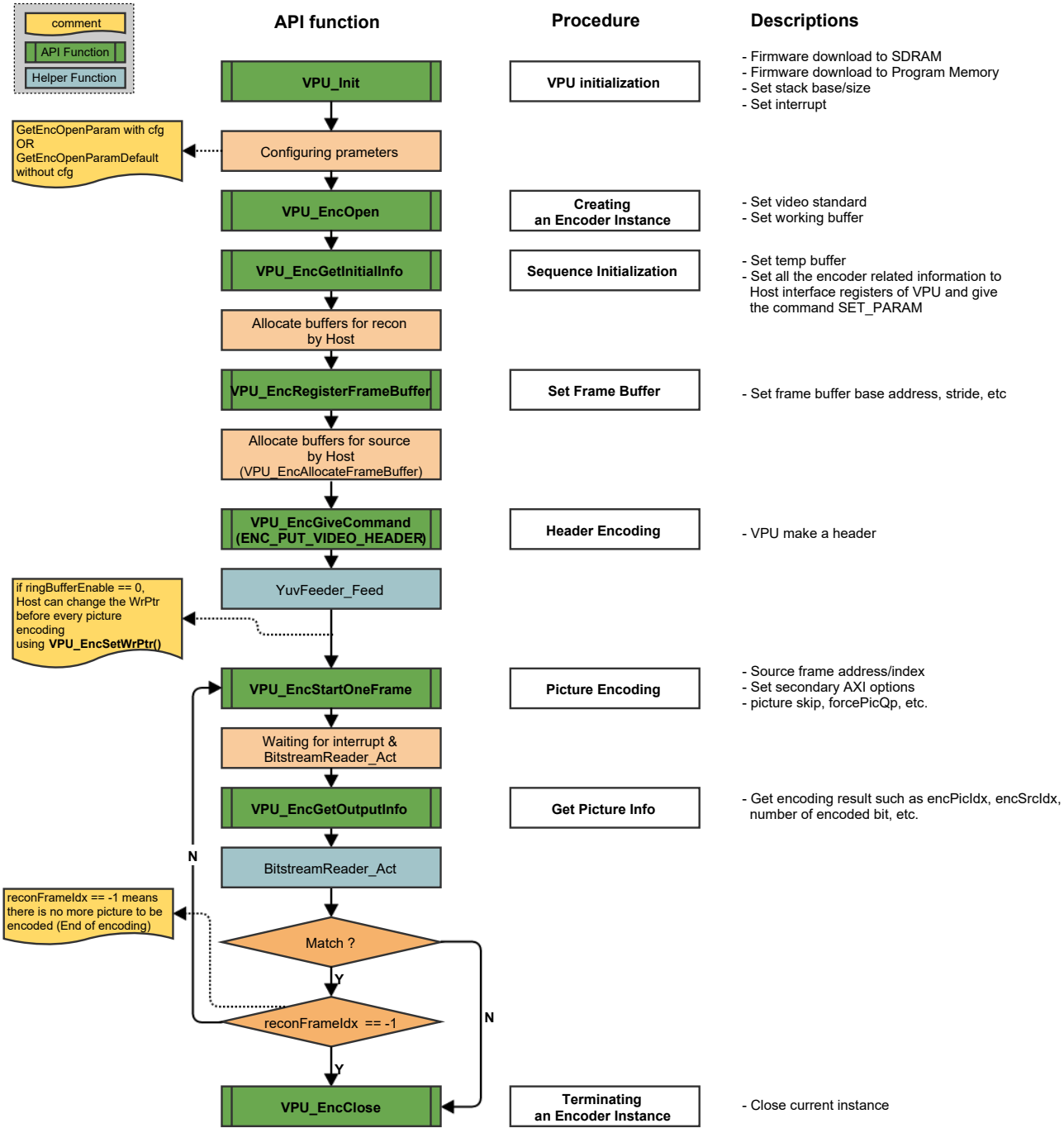

Figure 4.4. w4_enc_test

Appendix A
DeriveLambdaWeight

This chapter explains about lambda weight encoding parameter. It covers how LambdaWeight is derived when DeriveLambdaWeight is 1 and cost functions where lambda weights are applied.

A.1. Lagrangian Constant Values

In WAVE4 encoder, two lambda values are used for cost computation as the below defined.

$$\lambda_{mode} = \begin{cases} \alpha * 2^{((QP-12)/3.0)} & if\ DeriveLambdaWeight = 1 \\ LamdaWeight * W * 2^{((QP-12)/3.0)} & if\ DeriveLambdaWeight = 0 \end{cases}$$

$$\lambda_{pred} = \sqrt{\lambda_{mode}}$$

Figure A.1. Lambda Values

- If DeriveLambdaWeight is equal to 0, the lambda values are derived using LambdaWeight given from host.
- If DeriveLambdaWeight is equal to 1, VPU derives the lambda values using α.

α represents a weighting factor dependent to picture type and reference count within a GOP, as specified in the following table.

Table A.1. Derivation of α

Picture Type	α
I	0.57 x Clip3(0, 0.5, 0.05 x (GOP size – 1))
P or B	0.57 x β x Clip3(0, 0.5, 0.05 x (GOP size – 1)) where, β = (ref_cnt_I + 1.0) / (ref_cnt +1.0) • ref_cnt_I : for I picture, the number of reference counting by inter pictures within a GOP (+1.0 for P picture, +0.5 for B picture), • ref_cnt : for the current picture, the number of reference counting by inter pictures within a GOP (+1.0 for P picture, +0.5 for B picture)

Table A.2. Derivation of W

Encoding order of the current picture in a GOP	Wk
1st picture	Clip3(2.0, 4.0, (QP-12)/6.0)
Otherwise	1

A.2. SAD based Cost Function

The cost for prediction parameter decision Jpred is specified by the following formula.

Jpred = SAD + λ pred * β pred,

where Bpred specifies bit cost to be considered for mode decision, and SAD is the sum of difference between original block and reconstructed block. This cost function is used for motion estimation, merge candidate decision, and so on. VPU decides motion vectors which minimize Jpred.

A.3. SSD based Cost Function

The cost for mode decision Jmode is specified by the following formula.

Jmode = SSD + λ mode * β mode,

where β mode specifies bit cost to be considered for mode decision, and SSD is the sum of squared difference between original block and reconstructed block. This cost function is used for CU size decision, PU type decision, intra mode decision, and so on. VPU selects the mode and type, which has a minimum Jmode among the candidates.

About Chips&Media

Chips&Media, Inc. is a leading video IP provider, headquartered in Seoul, South Korea. The company was established in 2003 by leading members with years of profound experiences in video standard technologies and the semiconductor industry. Our extensive catalogue of IP solutions includes video postprocessing, video frame buffer compression as well as video codecs covering vast range of video standards from MPEG-2, MPEG-4, H.263, Sorenson, H.264, VC-1, AVS/AVS+, VP8/9, HEVC(H.265), and AV1 for HD to UHD (4K/8K) resolution.

Chips&Media has been developing a line of reliable, high-quality IP solutions that allow our customers and partners to satisfy the growing consumer demand for high-performance multi-media digital devices. Especially as a leading multi-standard video codec solution provider, we have been providing our advanced ultra-low power multi-codec video IPs to top-tier semiconductor companies.

With a mission to become a global top SoC IP provider, Chips&Media has introduced Image Signal Processing (ISP) IP and Deep learning-based super resolution IP to the market and continues to widen our solution portfolio to cope with growing demand on image processing and related solutions. To find further information, please visit the company's web site at *http://www.chipsnmedia.com*