Hardware Simulation Guide

7280 Encoder

Version 1.0

VISIBLY BETTER

Headquarters:
Hantro Products Oy
Kiviharjunlenkki 1, FI-90220 Oulu, Finland
Tel: +358 207 425100, Fax: +358 207 4205299

Hantro Finland
Lars Sonckin kaari 14
FI-02600
Espoo
Finland
Tel: +358 207 425100
Fax: +358 207 425298

Hantro Germany
Ismaninger Str. 17-19
81675
Munich
Germany
Tel: +49 89 4130 0645
Fax: +49 89 4130 0663

Hantro USA
1762 Technology Drive
Suite 202. San Jose
CA 95110,
USA
Tel: +1 408 451 9170
Fax: +1 408 451 9672

Hantro Japan
Yurakucho Building 11F
1-10-1, Yurakucho
Chiyoda-ku, Tokyo
100-0006, Japan
Tel: +81 3 5219 3638
Fax: +81 3 5219 3639

Hantro Taiwan
6F, No. 331
Fu-Hsing N. Rd.
Taipei
Taiwan
Tel: +886 2 2717 2092
Fax: +886 2 2717 2097

Hantro Korea
#518 ho, Dongburoot, 16-2
Sunaedong, Bundanggu,
Seongnamsi, Kyunggido,
463-825 Korea
Tel: +82 31 718 2506
Fax: +82 31 718 2505

Email: sales@hantro.com
WWW.HANTRO.COM

Copyright Information

Hantro Products Oy
Kiviharjunlenkki 1
90220 Oulu
FINLAND
www.hantro.com

Glossary

AHB	Advanced High Performance Bus, introduced in AMBA 2.0 specification.
AMBA	Advanced Microcontroller Bus Architecture, on-chip bus specification and an open standard
APB	Advanced Peripheral Bus protocol specification
AXI	AMBA Advanced eXtensible Interface protocol specification.
BASH	Bourne-Again Shell, a *de facto* standard for shell scripting in Linux/Unix
CSV	Comma Separated Value, an ASCII report form compatible with MS Excel
GUI	Graphical User Interface
Macroblock	A data unit typically consisting of four 8x8 luminance pixel blocks, one 8x8 Cb and one 8x8 Cr block
MB	Macroblock
Modelsim	Logic simulator from Mentor Graphics Corp.
OCP	Open Core Protocol
Verilog	Verilog hardware description language
VCS	Logic simulator from Synopsys
VHDL	Very high-speed integrated circuit Hardware Description Language
P-frame	Coding of a picture using inter prediction, which is derived from decoded samples of a reference picture other than the current picture

Table of Contents

1 Introduction

This document describes the 7280 AHB test environment for Mentor Graphics Modelsim and Synopsys VCS logic simulators, including the test bench files and the script files for running the tests. The test environment requires a Linux/Unix workstation with a BASH shell.

While the test benches probably support also other simulators, the benches have not been tested with them, and there is no script support included for them.

Chapter 2 describes the test environment with all the related files, and instructs in running simulation. Chapter 3 lists the reports the test environment creates and Chapter 4 describes the advanced verification methods for further stress testing of the product. In Chapter 5 the reference C-model generated trace files are presented.

2 Top-level Test Environment

The 7280 top-level RTL test environment consists of shell scripts, test data, and VHDL or Verilog test bench components. The RTL bus environment originates from the Micropack test environment for AHB by ARM Corporation, and has been modified by Hantro Products Oy. It supports 32-bit and 64-bit AHB, AXI and OCP bus master interfaces. The supported slave interfaces are 32-bit AHB, AXI, APB and OCP.

The test environment is available in both VHDL and Verilog languages.

The hierarchy of the test environment is depicted in Figure 1. The 7280 encoder top-level name depends on the bus configuration, and is named *<bus>7280enc*.

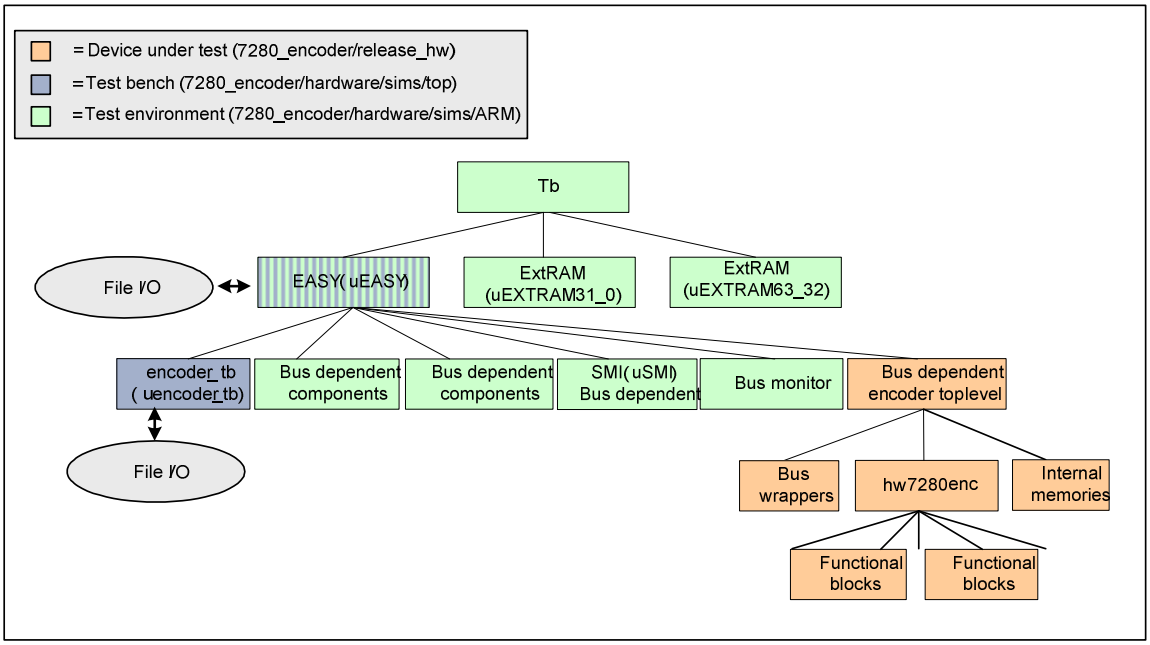

FIGURE 1. THE TEST ENVIRONMENT HIERARCHY.

All files required by the test environment are located in *7280_encoder/hardware/sims/top* and 7280*_encoder/hardware/sims/ARM* folders. The following paragraphs introduce the basic use of this environment, and then describe the files of the test environment.

Note: test_env_params.txt and simulation_ctrl.txt files (referred to often in the following paragraphs) are generated and written during the execution of test_rtl.sh. test_env_params.txt is written to 7280_encoder/hardware/sims/top/ folder and simulation_ctrl.txt to7280_encoder/hardware/sims/top/testdata folder . These files cannot be found in the 7280_testdata.tar.gz package.

2.1 Quick-start for running basic simulation

Running the functional test cases is very simple once these step-by-step instructions are followed:

1. Recover the 7280 encoder hardware folder from the delivery packet.
2. Go to *7280_encoder/hardware/sims/top* folder
3. Edit *test_rtl_variables.sh* for changing simulation parameters. Make sure that:
   - *$TESTDATASOURCE* is set to "zip"
   - *$TESTDATAPATH* is set to point to the folder where the 7280_testdata.tar.gz test data file is located
   - *$RTL_LANGUAGE* is set to either verilog or VHDL, depending on the delivered language
   - *$TB_LANGUAGE* is set to either verilog or VHDL, depending on the delivered test bench
   - *$RTL_TOP is set to ahb, axi, axiahb, axiapb or ocp, depending on the delivered bus combination*
4. Run *test_rtl.sh*. Usage instructions as well as the available test cases will be displayed if no arguments are given. See 2.4 for further info.

2.2 AHB test environment VHDL files in *ARM* folder

The VHDL/Verilog files used by the test environment in *7280_encoder/hardware/sims/ARM* folder are following:

- **ADecoder** Provides the *HSEL** module select outputs to the AHB system slaves, and controls the read data multiplexer. Defines the base address for the memory-mapped registers of the encoder. Not used in AXI or OCP test environments.

- **swreg_macros** This file contains the procedures *AHBaction, AXIaction, APBaction* and *OCPaction* that are used by the test bench to access the different type of slave interfaces.

- **Arbiter** The arbiter processes requests for ownership of the AHB bus and grants one bus master according to the arbitration scheme. The arbitration scheme of this implementation is a simple priority encoded scheme where the highest priority master requesting the bus is granted. Not used in AXI or OCP test environments.

  Uses the *test_env_params.txt* containing timing specific environment parameters.

- **Busmonitor** This file monitors the master burst correctness, and measures the data traffic caused by the encoder master interface.

- **Conv** Contains the type conversion functions *Hex2StdVec* and *StdVec2Int* used with *textio* modules.

- **EASY_hantro** EASY_hantro is the top-level of the bus architecture. It connects the bus components, test bench and encoder together and has an interface to the external memory. It includes a three-state multiplexer that selects which component is accesses external memory.

  EASY_hantro loads input data to external memory before encoding and checks external memory content after each encoded image.

- **ExtRAM** A behavioural model of a 32-bit off-chip RAM. The address bus is 22 bits wide which gives an address range of [0, 4194303]. Two instances of this component are used to form a 64-bit memory model.

- **MuxM2S** Central multiplexer for signals from AHB bus masters to bus slaves. Also generates the default master outputs when no other masters are selected. Not used in AXI or OCP test environments.

- **MuxS2M** Central multiplexer for signals from AHB bus slaves to bus masters. Not used in AXI or OCP test environments.

- **SMI_AHB** AHB static memory interface with configurable NONSEQ/SEQ wait states or SDRAM row change delay. Generates chip, output and write enables to ExtRAM.

  Uses the package *bus_config_pkg* containing timing specific environment parameters.

- **SMI_AXI** AXI static memory interface with configurable command and write buffer size and delay. Generates chip, output and write enables to ExtRAM.

  Uses the *test_env_params.txt* containing timing specific environment parameters.

- **SMI_OCP** OCP static memory interface with configurable command and write buffer size and delay. Generates chip, output and write enables to ExtRAM.

  Uses the *test_env_params.txt* containing timing specific environment parameters.

tb The test bench top-level. Maps the EASY and two 32-bit SDRAM components (The other is used to model the 64-bit bus system). Generates system clock *HCLK* and asynchronous reset *HRESETn* (AHB names).

2.3 AHB test environment VHDL files in *top* folder

The used files in 7280_encoder/hardware/sims/top folder are following:

- **bus_config_pkg.vhd** This package file is written by test scripts and it includes the information of encoder master and slave interfaces set in test_rtl_variables.sh (RTL_TOP). The file is read by tb,vhd, EASY_hantro.vhd and busmonitor.vhd.

- **encoder_tb.vhd** The testbench modeling the CPU of the system. Translates the system model traces and environment variables set in *test_rtl_variables.sh* into slave interface register write operations. More specifically, the bench performs following operations:
  - Prints information about the current test case (e.g. encoding mode, picture size, input format)
  - Defines byte base addresses for encoder input and output data
  - Writes control words to hardware memory-mapped registers
  - Checks that control writing succeeded by reading all registers and comparing with written data
  - Enables hardware and polls for an interrupt (or listens to interrupt line)
  - Interprets reason for given interrupt by analyzing status bit and encoding
  - If synchronous reset testing is enabled:
    - Generates extra synchronous reset during encoding of each picture
  - Terminates the simulation if interrupt is not coming early enough (Timeout)
  - Writes denali format recording of the slave register write operations
  - Measures encoding performance in used clock cycles per macroblock
  - Calculates average performance for the whole test sequence

- Enables *EASY_hantro.vhd* test bench file after proper interrupt (EASY will then perform result checking)

Input from scripts:
- *test_env_params.txt* (all simulation and reporting related parameters)

2.4 AHB test environment scripts in *top* folder

All scripts required in simulation are found in *the 7280_encoder/hardware/sims/top* folder. The scripts are following (in alphabetical order):

- **compile_rtl.txt** Contains a list of the VHDL source codes in hierarchical order. This file is used by the simulation script to generate the compile script, and can also be used in synthesis script creation.

- **test_rtl_functions.sh** This file contains the functions that the main simulation script *test_rtl.sh* calls. The included functions are described below. The script itself is commented as well.

TABLE 1. FUNCTIONS FOR THE RTL SIMULATION IN THE *TEST_RTL_FUNCTIONS.SH*

Function name	Description
showInstructions	Shows instructions if no arguments or invalid arguments are given to *test_rtl.sh.*
setCasesToRun	Defines the list of test cases to be simulated based on the given command line argument.
validCase	Checks that the defined test case is valid (case must be found in *testcase_list*)
parseArgs	Parses the command line arguments.
autoContinue	Function for requesting permission to continue from user with a configurable time out.
createCompileScript	Creates the compile script and sets commands for the selected simulator and RTL language.
compileRTL	Compiles the 7280 source codes if the current RTL files have not already been compiled. Runs compileTB function.
compileTB	Compiles the test environment.
VcsElab	Performs VCS elaboration
randomizeVariable	Function for randomizing a simulation parameter.
randomizeSettings	Randomizes the input parameters (if randomization enabled), and checks the validity of the randomized values against current test case requirements.
writeConfigFiles	Writes the simulation parameters to files (*simulation_ctrl.txt and test_env_params.txt*) that will be read by the test benches.
unzipTestData	Extracts data from the test data zip.
setup2hex	Converts simulation_ctrl.txt content to hexadecimal format for verilog test bench
simRTL	Starts the simulator, runs a single test case, and writes the simulation report to *./reports* folder
reportCoverage	Creates a HTML coverage report after all cases have been simulated. VCS simulator only.
initPerformanceFile	Initializes performance CSV file
setPerformanceVariables	Overrides given parameters with performance parameters

simPerformance	Measures encoding speed vs. increasing latencies and memory wait states by running the selected test case several times in a loop while changing timing parameters
writePerformanceFile	Stores the performance results to a CSV file
reportActivity	Creates a logic activity report using Modelsim toggle coverage. Activity information can be used for estimating the hardware power consumption.
reportCSV	Appends the current case test status to the CSV report file. Executed after each test case run.
writeCSVfile	Copies the final CSV report to target folder. Executed after all cases run.
makeGraph	Generates busload and performance graph pdf from simulated test case.

- **test_rtl.sh** The main script that must be executed when simulating the product. Calls the *test_rtl_functions.sh* and the *testcase_list* files.

  Usage: `test_rtl.sh [case_nbr] [options]`, where

  `[case_nbr]` is the test case number, a test category, or a set of test cases in double quotes, e.g. "7 303 1014 2103",

  [`options`]
  '`gui`' opens the simulator in graphical user interface mode. Omitting "gui" runs the simulation in command-line mode.
  '`clear`' Clears all libraries and compiles the design again.
  '`check`' compiles white-box test structures. Hantro internal use only.

Note: The script prints additional usage information to terminal window if executed without parameters.

Note: The script calls also two Hantro internal script files (test_rtl_functions_internal.sh, test_rtl_exhaustive.sh) that are not releasable to customers. These files contain reference C-model interfacing, hardware configuration and additional test and measurement related functions. The lack of these files does not prevent the sanity testing of the product in any way.

TABLE 2. COMMAND LINE OPTIONS IN TEST_RTL.SH

Command line option	Affected environment variable	Purpose
-gui	N/A	Start simulator in GUI mode
-clean	N/A	Force full RTL compilation, remove old libraries, test data folders and reference models
-nodata	WRITE_TESTDATA	Runs the simulation with the current, previously generated test data
-rand <nbr>	RANDOMIZE_ENV, RANDOM_ROUNDS	Randomize all relevant settings for each testcase. If <nbr> is omitted, RANDOM_ROUNDS is set to 1.
-bus <bus>	BUS_IF	Select bus interface for design. Valid settings: ahb, axi, axiahb, axiapb, ocp.
-sim <simulator>	SIMULATOR	Select simulator to use. Valid settings:

		vcs, vsim
-pics <nbr>	PICTURES	Amount of pictures to simulate per test case
-first <nbr>	FIRST_PICTURE	First picture to start the simulation from. Only VHDL test bench supports frame skipping.
-lang <language>	RTL_LANGUAGE, TB_LANGUAGE	RTL and test bench language. Valid settings: vhdl, verilog
-cover <mode>	GET_COVERAGE, HW_COVERAGE_VERSION	Enable coverage simulation. Valid settings: local, latest, <tagname>
-csv	N/A	Enable CSV report writing to $REPORT_PATH/simulation folder. Local report is generated always
-check	N/A	Compile white-box test structures
-report_path <path>	REPORT_PATH	Define report path

- **test_rtl_variables.sh** The file containing all simulation settings and environment variables. The variables are commented well in the script file so they are not described here in detail. The settings are categorized into following groups:
    - Simulator settings
    - Test data source settings
    - Reference C-model settings (Hantro internal)
    - Compile-time configuration settings (Hantro internal)
    - Compilation settings
    - Constrained random test settings
    - Simulation settings
    - SDRAM settings
    - Bus environment settings
    - Memory-mapped register settings not set by reference model
    - Simulation result reporting settings
    - Performance simulation settings
    - Activity measurement settings

3 Reporting

This chapter describes the reports created by the test environment.

3.1 Test status report

The test bench will print following test status messages at the end of a test case that indicate the success of the test run:

- teststatus : OK
  - Test case passed successfully
  - OK for VOPS 0-X, testdata end too early refers to case where test data tables are too small.

Following test status messages are used to report an error during test run:

- teststatus : FAILED
  - Test case failed, output data does not match reference data
  - Test case failed, register values after interrupt does not match reference data
- teststatus : TIMEOUT
  - HW did not give picture ready interrupt in a limited time
- teststatus : INVALID
  - Too much external memory was allocated
  - Test is executed only in FPGA environment

The transcripts from the simulation run are stored to *./reports* folder and are labeled *transcript_{CASE}.rpt,* where *{CASE}* stands for test case number. The *reports* folder may contain reports from older simulation runs.

Note: Assertions with severity "failure" are used for exiting the simulator even in OK cases. The severity print should not be interpreted as a failed test case, instead see the teststatus print.

Note: Simulator settings should be such that severity 'failure' ends simulation.

3.2 Simulation report

The test bench will always generate a MS excel compatible simulation report *CSVreport.tmp* in the running folder. If the $REPORT_CSV variable is set to "YES", the report will be copied to the folder defined by the *$CSV_PATH* variable.

3.3 Coverage report

If the *$GET_COVERAGE* variable is set to "YES", the test bench will simulate the RTL with coverage measurement activated. Also, a coverage report will be created in *vcs_coverage_reports* folder.

Note: Coverage reporting is supported only with VCS simulator

3.4 Performance report

Simple performance results for the current run are always printed by the test bench. There is a separate 'used clock cycles per macroblock' print after encoding each picture (Picture n performance : xxxx cycles / MB), and one at the end of simulation that calculates the average performance for the whole sequence (Average performance : xxxx cycles / MB).

Furthermore, if the *$MEASURE_PERFORMANCE* variable is set to 1, the test bench will simulate the selected test case several times in a loop while changing the latency, bus width and memory waitstate parameters between runs. With the resulting CSV file, performance graphs such as the ones presented in the Hardware Integration Guide [1] can easily be created. The report is written to the location defined by the *$REPORTPATH* variable.

The performance simulation loop values are set in the *simPerformance()* function in *test_rtl_functions.sh*.

3.5 Bus traffic report

The test bench will record all bus traffic issued by the encoder master interface when the *$ENABLE_BUS_TRAFFIC_RECORDING* variable is set to 1. When the traffic recording is enabled, the test bench will print burst distribution data at the end of the each encoded picture. This amount of bursts is given per picture and per macroblock. Also the total amount of transferred data per picture and the average amount of transferred data per macroblock is shown at the bottom of the report.

If the *$ENABLE_BUS_TRAFFIC_RECORDING* variable is set to 2 the test bench will also generate record file containing all encoder master transactions. The record file is generated to *./ bus_access_record_<bus>.csv* in the running folder. The Excel-compatible traffic record shows the memory accesses the encoder issues and how they are distributed over time. The appearance of the report is a little bit different for AXI and AHB bus environments

3.6 Activity report

If *$MEASURE_ACTIVITY* parameter is set to 1, the toggle coverage option of the Modelsim simulator will be used. The list of measured units is located in *the test_rtl_functions.sh* file, in the *measure_units* variable set in *reportActivity()* function. The activity is printed out at the end of the simulation. These figures can be used for power analysis purposes.

4 Advanced Verification Methods

This chapter presents the verification methods available for the stress testing of the product.

4.1 Constrained Random Verification

One of the most efficient verification methods available, the constrained random verification is possible with the 7280 test environment. Random verification is enabled by setting the *$RANDOMIZE_ENV* variable to 1 in *test_rtl_variables.sh* file. Parameters will then be randomized with the bash *$RANDOM*-function and constrained with the modulo operator (%) to a desired range. See *test_rtl_functions.sh,* function *writeConfigFiles* for details.

Each run will be different as all possible environment and bus configuration related parameters are changed at random.

Note: In order to gain maximal test coverage, randomizing should be enabled.

5 Reference Trace Files and Format

In this chapter the trace files used in RTL simulation are presented. VHDL simulation uses the *.trc* files and Verilog simulation uses the *.hex* files. All trace files are generated with the reference C-models.

5.1 Reference trace file overview

In Table 3 the list of the trace files in top-level HW simulation is presented. All *.trc* files have their hexadecimal counterparts for Verilog simulation.

TABLE 3. TRACE FILES USED IN HW SIMULATION

Trace file name	Used in encoding scheme	Description
swreg_params.trc	H.264 MPEG-4 H.263 JPEG	Memory-mapped encoder register parameters for each picture
cam_image.trc	H.264 MPEG-4 H.263 JPEG	Encoder input picture in specified mode. This image is encoded.
next_cam_image.trc	STABILIZATION	Stabilization input picture in specified mode. This image is used to find correct stabilization vector for next encoded image.
jpeg_tables.trc	JPEG	JPEG quantization tables.
recon.trc	H.264 MPEG-4 H.263	Encoder reference image in raster scan order. This data will be used for checking the output results.
stream_data_hw_out.trc	H.264 MPEG-4 H.263 JPEG	Output stream. Encoded stream is compared to this data.
nal_unit_size.trc	H.264 H.263	Size of each NAL unit for H.264, or GOB in H.263.
mb_control.trc	MPEG-4	Macroblock control data. Required if software is performing entropy encoding.(video packet MPEG-4 stream)
rlc.trc	MPEG-4	Run-length-coded data. Required if software is performing entropy encoding.

5.2 Reference trace file descriptions

The following paragraphs describe the contents of each trace file. For verilog test bench there is similar "trace".trc.hex for each trace, where is no comments, and all values are hexadecimal.

5.2.1 swreg_params.trc

SW register parameters trace contains the control parameters of the 7280 encoder. There is parameter value with comment per line in trace. If value may require more than 16 bits to be presented, it is divided in to two parts in same line.

```
xxxx    interFavor                          vop=0 irq=0
xxxx    intra16Favor (H264)
xxxx    prevModeFavor (H264)
xxxx    diffMvPenalty
xxxx    lumWidth
xxxx    lumHeight
xxxx    lumWidthSrc
xxxx    lumHeightSrc for TB
xxxx    XFill
xxxx    YFill
xxxx    byteStream
xxxx    roundControl (Mpeg4/H263)
xxxx    frameNum
xxxx    idrPicId
xxxx    picInitQp
xxxx    qpLum
xxxx    qpMax
xxxx    qpMin
xxxx    checkPointDist
xxxx    hwOutMode
xxxx    streamType: 0=MPEG4, 1=H263, 3=H264
xxxx    frameType: 0=INTER, 1=INTRA
xxxx    constIntraPred (H264)
xxxx    mbRowPerSlice (H264)
xxxx    inputFormat
xxxx    rotation
xxxx    stabilization
xxxx    pixelOffsetLum
xxxx    pixelOffsetChr
xxxx    disableDeblockingFilterIdc (H264)
xxxx    filterOffsetA (H264)
xxxx    filterOffsetB (H264)
xxxx    chromaQpOffset (H264)
xxxx    irqInterval
xxxx    DataBufferLimit
xxxx    rateControlCheckpoint1
xxxx    rateControlCheckpoint2
xxxx    rateControlCheckpoint3
xxxx    rateControlCheckpoint4
xxxx    rateControlCheckpoint5
xxxx    rateControlCheckpoint6
xxxx    rateControlCheckpoint7
xxxx    rateControlCheckpoint8
xxxx    rateControlCheckpoint9
xxxx    rateControlCheckpoint10
xxxx    rateControlCountError1
xxxx    rateControlCountError2
xxxx    rateControlCountError3
```

```
xxxx    rateControlCountError4
xxxx    rateControlCountError5
xxxx    rateControlCountError6
xxxx    rateControlDeltaQP1
xxxx    rateControlDeltaQP2
xxxx    rateControlDeltaQP3
xxxx    rateControlDeltaQP4
xxxx    rateControlDeltaQP5
xxxx    rateControlDeltaQP6
xxxx    rateControlDeltaQP7
xxxx    NALSizeWriteOut
xxxx    bufferStartOffset
xxxx    JPEGSliceEnable
xxxx    restartMarkerInterval
xxxx    restartMarker
xxxx    MBRowGOB
xxxx    GOBFrameID
xxxx    GOBHeaderEnable
xxxx    nonZeroCoeffCount
xxxx    HWStreamDataCount
xxxx    MBQPSum
xxxx    picParameterSetId
xxxx    vpSize
xxxx    vpMbBits
xxxx    hec
xxxx    moduloTimeBase
xxxx    intraDcVlcThr
xxxx    vopFcodeForward
xxxx    vopTimeInc
xxxx    vopTimeIncBits
xx xx   headerRemainderBits1
xx xx   headerRemainderBits2
xxxx    interFavor                      vop=0 irq=0
xxxx    intra16Favor (H264)
xxxx    prevModeFavor (H264)
xxxx    diffMvPenalty
. . .
```

5.2.2 cam_image.trc

cam_image.trc describes the encoder input picture. The format of the trace depends on the input picture format, which can be either YCbYCr 4:2:2 or YCbCr 4:2:0 planar or semi-planar.

```
vop=0 lum
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
…
vop=0 Cb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
…
vop=0 Cr
CrCrCrCrCrCrCrCrCrCrCrCrCrCr
CrCrCrCrCrCrCrCrCrCrCrCrCrCr
CrCrCrCrCrCrCrCrCrCrCrCrCrCr
…
vop=1 lum
…
```

FIGURE 2. CAM_IMAGE.TRC WHEN INPUT FORMAT IS YCBCR PLANAR 4:2:0

```
vop=0 lum
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
…
vop=0 Ch
CbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCr
CbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCr
CbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCrCbCr
…
vop=1 lum
…
```

FIGURE 3. CAM_IMAGE.TRC WHEN INPUT FORMAT IS YCBCR SEMI-PLANAR 4:2:0

```
vop=0 YUYV 422
YUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYV
YUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYV
YUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYV
…
vop=1 YUYV 422
…
```

FIGURE 4. CAM_IMAGE.TRC WHEN INPUT FORMAT IS YCBYCR 4:2:2

```
vop=0 UYVY 422
UYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVY
UYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVY
UYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVYUYVY
…
vop=1 UYVY 422
…
```

FIGURE 5. CAM_IMAGE.TRC WHEN INPUT FORMAT IS CBYCRY 4:2:2

5.2.3 next_cam_image.trc

next_cam_image.trc describes the encoder input picture. The format of the trace depends on the input picture format, which can be either YCbYCr 4:2:2 or YCbCr 4:2:0 planar or semi-planar. In case of YCbCr 4:2:0 trace does not include chrominance parts, since those are not used in counting stabilization vector. In 422 format trace format is identical with cam_image.trc

```
 vop= 0 lum
    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
…

 vop= 1 lum

    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
    YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

FIGURE 6 . NEXT_CAM_IMAGE.TRC

5.2.4 recon.trc

Reconstructed output data trace contains data for new reference picture.

```
vop=0 lum
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
…
vop=0 Cb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
CbCbCbCbCbCbCbCbCbCbCbCbCbCb
…
vop=0 Cr
CrCrCrCrCrCrCrCrCrCrCrCrCr
CrCrCrCrCrCrCrCrCrCrCrCrCr
CrCrCrCrCrCrCrCrCrCrCrCrCr
…
vop=1 lum
…
```

5.2.5 stream_data_hw_out.trc

In stream_data_hw_out.trc is encoded stream byte by byte, 8 bytes per line.

```
Data 1   Data 0   VOP   MB
(63:32)  (31:0)
---------------------------------
0                  vop=0 mb=0
0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0
...
0                  vop=1 mb=0
0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0
...
```

FIGURE 7. ENCODER STREAM DATA OUTPUT TRACE

5.2.6 rlc.trc

rlc.trc includes runs and values of each run length coded words. Each value describes the content of a 16-bit RLC word. At the start of each block there is a comment. The trace is used only when software is performing entropy encoding.

```
xxx xxxxx   vop=0 mb=0
xxx xxxxx
...
xxx xxxxx
xxx xxxxx   vop=0 mb=1
...
```

5.2.7 mb_control.trc

mb_control.trc includes macroblock control data in 16 bit words. The trace is used only when software is performing entropy encoding.

```
32769  4119  vop=0 mb=0
 4654   262
  61    54
  62    55
 128   120
   0     0
32769  3598  vop=0 mb=1
 3849  1798
  57   107
  57   110
 128   115
   0     0
   ...
```

FIGURE 8. MB_CONTROL.TRC

5.2.8 nal_unit_size.trc

nal_unit_size.trc includes size of each NAL unit in bytes. If slice size equals picture size, these is only one value for each picture.

```
   size | vop   | nal
----------------------
55408 | 000   | 000
  7240 | 001   | 000
```

References

[1] Hantro Products (2008). 7280 Hardware Integration Guide.