Sail River Gen III (WS-RLN-00101)

Simulation User Manual

Last updated: **2025-05-22**

Altera Confidential

Contents

List of Figures

List of Tables

1.0 Introduction

1.1 Objective

This document describes the steps to run the simulation in Questasim tool and the coverage report for all the test cases ran.

1.2 Tools

The tools required to run the simulation are **Questasim** version **23.1 (Full Edition)** and **Quartus** version **24.3.**

1.3 Defines Used in Simulation

Below defines are present in **o_ru_defines.sv** inside **~/verif/o_ru/o_ru_tb_files/**. This File contains the defines used for simulation like clock Period for each interface and the scoreboard enable and disable options.

Figure 1-1. Picture of Defines used in Simulation

```
`define MAC_CLK_PERIOD 2.4824          /// 402.83 MHz MAC Clock Period
`define DSP_CLK_PERIOD 2.0345          /// 491.52 MHz DSP Clock Period
`define ETH_CLK_PERIOD 2.56            /// 390.625 MHz ETH Clock Period
`define CSR_CLK_PERIOD 10              /// 100 MHz CSR Clock Period
`define ORAN_LPBK_EN 0                 /// ORAN LOOPBACK will be used in case of oRAN Loopback Design
`define PCAP_LPBK_EN 0                 /// IF Enabled Reference Model will read the input PCAP at UL and compare against RTL

`define ECPRI_DMPR_EN 1                /// eCPRI Demapper Checker 1 - Enable 0 -Disable
`define ORAN_DMPR_EN 1                 /// oRAN Demapper Checker 1 - Enable 0 -Disable
`define FDV_DMPR_EN 1                  /// FDV Demapper Checker 1 - Enable 0 -Disable
`define IFFT_EN 1                      /// IFFT Checker 1 - Enable 0 -Disable
`define INTRLVR_EN 1                   /// Interleaver Checker 1 - Enable 0 -Disable
`define DUC_EN 1                       /// DUC Checker 1 - Enable 0 -Disable
`define MIXER_DL_EN 1                  /// Mixer DL Checker 1 - Enable 0 -Disable
`define SUMMER_EN 1                    /// Summer Checker 1 - Enable 0 -Disable
`define INTERPOLATOR_EN 1              /// Interpolator Checker 1 - Enable 0 -Disable
`define DECIMATOR_EN 1                 /// Decimator Checker 1 - Enable 0 -Disable
`define MIXER_UL_EN 1                  /// Mixer UL Checker 1 - Enable 0 -Disable
`define DDC_EN 1                       /// DDC Checker 1 - Enable 0 -Disable
`define DEINTRLVR_EN 1                 /// Deinterleaver Checker 1 - Enable 0 -Disable
`define FFT_EN 1                       /// FFT Checker 1 - Enable 0 -Disable
`define PRACH_EN 1                     /// PRACH Checker 1 - Enable 0 -Disable
`define C_U_COUPLING_EN 1              /// CU_Coupling Checker 1 - Enable 0 -Disable
`define ORAN_MPR_EN 1                  /// oRAN Mapper Checker 1 - Enable 0 -Disable
`define ECPRI_MPR_EN 1                 /// eCPRI Mapper Checker 1 - Enable 0 -Disable
`define FILE_BASED 1                   /// Enable File based Checkers for Low PHY blocks  0 - For C-Model iFFT - FFT
`define FILE_WRITE_EN 0                /// Eanble file write for all output blocks
```

2.0 Simulation Process

2.1 Pre-requisites

1. Refer "**WS-RLN-00098 Sail River Gen III Hardware User Manual -> Section 4.2 Quartus Software Installation**" for Quartus and IP downloads.
2. Refer "**WS-RLN-00098 Sail River Gen III Hardware User Manual -> Section 5.1 Quartus IP Patches**" for FH Comp IP patch.
3. Set the "**QUARTUS_ROOTDIR**" path to your Quartus installation path.
4. Update PATH Environment variable with "**QUARTUS_ROOTDIR"/bin:$PATH**.
5. Set the "**QSYS_ROOTDIR**" path to your Qsys path.
   Example:

```
export QSYS_ROOTDIR="Quartus_installation_directory/qsys/bin"
```

6. Ensure G++ version is higher than v4.8.5.

2.2 Steps to run Simulation in Questasim

2.2.1 Generate Expected Vectors

Refer **Simulating MATLAB Model for Expected Vectors** section for generating simulation expected vectors. Make sure to generate the expected vectors before starting the simulation.

2.2.2 Steps to run Simulation from Command Line

1. Clone the Release repository.
2. Go to **~/sim/verif/o_ru/sim/** path in the Linux terminal.
3. Enable Read and Write permission for executable scripts using chmod 777 <Script Name>.
4. Ensure to have required cmake version in the simulation path to execute the setup scripts (cmake3 version is required).
5. If cmake3 is not available, cmake version 3.26.4 can also be used. If cmake is used, replace cmake3 with cmake in **setup_cmodel.sh** file in sim folder.

Figure 2-1. setup_cmodel file for cmake3 replace

```
#generate .so file for fft c-model
cd ../pp_c_model/fft_cmodel/
rm -r build
mkdir build
cd build
cmake3 ../
make

#generate .so file for ifft c-model
cd ../../ifft_cmodel/
rm -r build
mkdir build
cd build
cmake3 ../
make
```

6. This setup_cmodel.sh file generates .so files for all C models. The .so files are directly included in Makefile and compiled.
7. Below **Table 2-1** explains the compile time options present in the "run_ques" file inside sim/ folder.

Table 2-1. List of compile options in run_ques file

Compile Options	Usage Description
make clean	This command will clean the compilation logs.
make comp_lib	This command is used for compiling the Device Libraries and will create a work library. It will also generate IP's and QSYS RTL Files needed for running the simulation. It is enough to compile this for the first time. Note: If work/ folder is deleted, then need to do this make comp_lib again
COV	0 – No Coverage (Default). 1 – Coverage is Enabled.
WAVE	0 – Waveform Dump will not be created. 1 – Waveform Dump will be created in the name phipps_peak.wlf. (Default)
GUI	0 – GUI will be disabled (Default). 1 – It will automatically open the Questa Sim GUI, and the simulation will be running, and the display messages can be viewed in transcript window of the GUI
ERR_CNT	1 – Simulation will stop after UVM_ERROR 1. (Default) >1 – Simulation will stop after the value which you give.
QUESTASIM	1 – Needed for running the Simulation in Questa Sim, Other tool compatibility will be brought later. (Don't Change the value).

Figure 2-2. Options inside run_ques file

```
make clean
GUI=0

COV=0

TEST_NAME=$1

echo $TEST_NAME

QUESTASIM=1

ERR_CNT=1

WAVE=1

SYM_NUM=$2
SF_NUM=$3
DFE_LPBK_EN=$4
IFFT_FFT_LPBK_EN=$5
ECPRI_ORAN_SS_EN=$6
LONG_SHORT_PRACH=$7
CC2_DIS=$8

make comp_ques GUI=$GUI UVM_TESTNAME=$TEST_NAME SEED=1 QUESTASIM==$QUESTASIM UVM_MAX_QUIT_COUNT=$ERR_CNT WAVE=$WAVE COV=$COV SYM_NUM=$SYM_NUM SF_NUM=$SF_NUM DFE_LPBK_EN=$DFE_LPBK_EN
IFFT_FFT_LPBK_EN=$IFFT_FFT_LPBK_EN ECPRI_ORAN_SS_EN=$ECPRI_ORAN_SS_EN LONG_SHORT_PRACH=$LONG_SHORT_PRACH CC2_DIS=$CC2_DIS
```

8. To run the simulation for one test, use the command below in sim/ folder,
9. **./run_ques <ARGS1> <ARGS2> <ARGS3> < ARGS4> < ARGS5> <ARGS6> <ARGS7> <ARGS8>**

   ARGS1 - Testcase name

   ARGS2 – Number of symbols required to run the testcase

   ARGS3 – Number of sub-frames required to run the testcase

   ARGS4 – DFE Loopback Enable/Disable

   ARGS5 – IFFT-FFT Loopback Enable/Disable

   ARGS6 – ECPRI-ORAN Subsystem Enable/Disable

   ARGS7 – Long-Short PRACH Enable/Disable, 0 – **Long PRACH,** 1 - **Short PRACH**

   ARGS8 – CC2 Disable, **0 – Both CC1 & CC2 Enable, 1 – Only CC1 Enable**

   Example: Below is for the testcase mentioned to run for 28 Symbols, Subframe in DFE Loopback with CC2 Disable.

```
./run_ques tc_c_u_coupling_case_1_compress 29 1 1 0 0 0 1
```

Note: If all the loopbacks are disabled, then that testcase is for UL injection.

10. If in case if you face any header file issues for files such as **"svdpi.h" or "pcap.h"**, please add those header files in C-Library path**(/usr/include/pcap.h**) or add the relative path of your tool pointing to that file inside **~/verif/o_ru/o_ru_c_model/pcap_parse.c**.

    ~/QuestaCore/questasim/include/svdpi.h

11. After compilation, a simulation log file is created in sim/ folder. The simulation log file name will be in the mentioned format,

    o_ru_$(FEAT_NAME)_$(SEED)_$(DFE_LPBK_EN)_$(IFFT_FFT_LPBK_EN)_sim.log

    Example: o_ru_tc_c_u_coupling_case_1_compress_1_1_0_sim.log

12. The simulation log file will contain all the information regarding the Register writes done, number of packets/symbols Passed/Failed in each RTL block with corresponding module names.

Figure 2-3. Display Messages inside Simulation Log file

```
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 284386479000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: MIXER_UL_LINEUP_1, PKT_NO : 3  SCB_NUM : 18, MAX_VALUE : 3 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 284386479000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: MIXER_UL_LINEUP_1, PASSED PKT_CNT=4
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DDC_LINEUP_2, PKT_NO: 3, Scb_num : 21, Reference_data_size: 17792, Observed data_size: 17792
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DDC_LINEUP_2, PKT_NO : 3  SCB_NUM : 21, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DDC_LINEUP_2, PASSED PKT_CNT=4
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DDC_LINEUP_1, PKT_NO: 3, Scb_num : 20, Reference_data_size: 17792, Observed data_size: 17792
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DDC_LINEUP_1, PKT_NO : 3  SCB_NUM : 20, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 285961182000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DDC_LINEUP_1, PASSED PKT_CNT=4
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PKT_NO: 0, Scb_num : 23, Reference_data_size: 16384, Observed data_size
 16384
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PKT_NO : 0  SCB_NUM : 23, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PASSED PKT_CNT=1
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PKT_NO: 0, Scb_num : 22, Reference_data_size: 16384, Observed data_size
 16384
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PKT_NO : 0  SCB_NUM : 22, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 294320942500: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PASSED PKT_CNT=1
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PKT_NO: 1, Scb_num : 23, Reference_data_size: 16384, Observed data_size
 16384
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PKT_NO : 1  SCB_NUM : 23, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DEINTERLEAVER_LINEUP_2, PASSED PKT_CNT=2
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PKT_NO: 1, Scb_num : 22, Reference_data_size: 16384, Observed data_size
 16384
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PKT_NO : 1  SCB_NUM : 22, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 303258501000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: DEINTERLEAVER_LINEUP_1, PASSED PKT_CNT=2
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: FFT_LINEUP_2, PKT_NO: 0, Scb_num : 25, Reference_data_size: 7776, Observed data_size: 7776
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: FFT_LINEUP_2, PKT_NO : 0  SCB_NUM : 25, MAX_VALUE : 8 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(439) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [NMSE] Scoreboard_name: FFT_LINEUP_2, PKT_NO : 0  SCB_NUM : 25, NMSE (dB): -47.202846
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: FFT_LINEUP_2, PASSED PKT_CNT=1
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: FFT_LINEUP_1, PKT_NO: 0, Scb_num : 24, Reference_data_size: 7776, Observed data_size: 7776
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: FFT_LINEUP_1, PKT_NO : 0  SCB_NUM : 24, MAX_VALUE : 8 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(439) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [NMSE] Scoreboard_name: FFT_LINEUP_1, PKT_NO : 0  SCB_NUM : 24, NMSE (dB): -49.576911
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 306949084000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: FFT_LINEUP_1, PASSED PKT_CNT=1
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 309532160000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: COUPLING_PUSCH, PKT_NO: 0, Scb_num : 5, Reference_data_size: 7776, Observed data_size: 7776
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 309532160000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: COUPLING_PUSCH, PKT_NO : 0  SCB_NUM : 5, MAX_VALUE : 8 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 309532160000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: COUPLING_PUSCH, PASSED PKT_CNT=1
# UVM_INFO ../sr_scb/sr_cmp.sv(398) @ 310827520000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: COUPLING_PUSCH, PKT_NO: 1, Scb_num : 5, Reference_data_size: 7776, Observed data_size: 7776
# UVM_INFO ../sr_scb/sr_cmp.sv(433) @ 310827520000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: COUPLING_PUSCH, PKT_NO : 1  SCB_NUM : 5, MAX_VALUE : 8 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(191) @ 310827520000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: COUPLING_PUSCH, PASSED PKT_CNT=2
# UVM_INFO ../sr_scb/sr_cmp.sv(292) @ 311623680000: uvm_test_top.m_env.m_scb.m_cmp [Size Matched] Scoreboard_name: ORAN_MAPPER, PKT_NO: 0, Scb_num : 3, Reference_data_size: 4544, Observed data_size: 4544
# UVM_INFO ../sr_scb/sr_cmp.sv(368) @ 311623680000: uvm_test_top.m_env.m_scb.m_cmp [MAX_ERROR_RANGE] Scoreboard_name: ORAN_MAPPER, PKT_NO : 0  SCB_NUM : 3, MAX_VALUE : 2 , MIN_VALUE : 0
# UVM_INFO ../sr_scb/sr_cmp.sv(176) @ 311623680000: uvm_test_top.m_env.m_scb.m_cmp [SR_CMP] Scoreboard_name: ORAN_MAPPER, PASSED PKT_CNT=1
```

13. After simulation, open the **phipps_peak.wlf** waveform, select, **top => phipps_peak_inst**. This has a list of all modules. Right click the module for which the waveform should be added and select Add Wave to add the waveform.

    Command: vsim -view <WLF FILE NAME >

    Example: `vsim -view phipps_peak.wlf`

Figure 2-4. Illustration to add waveform to the waveform window

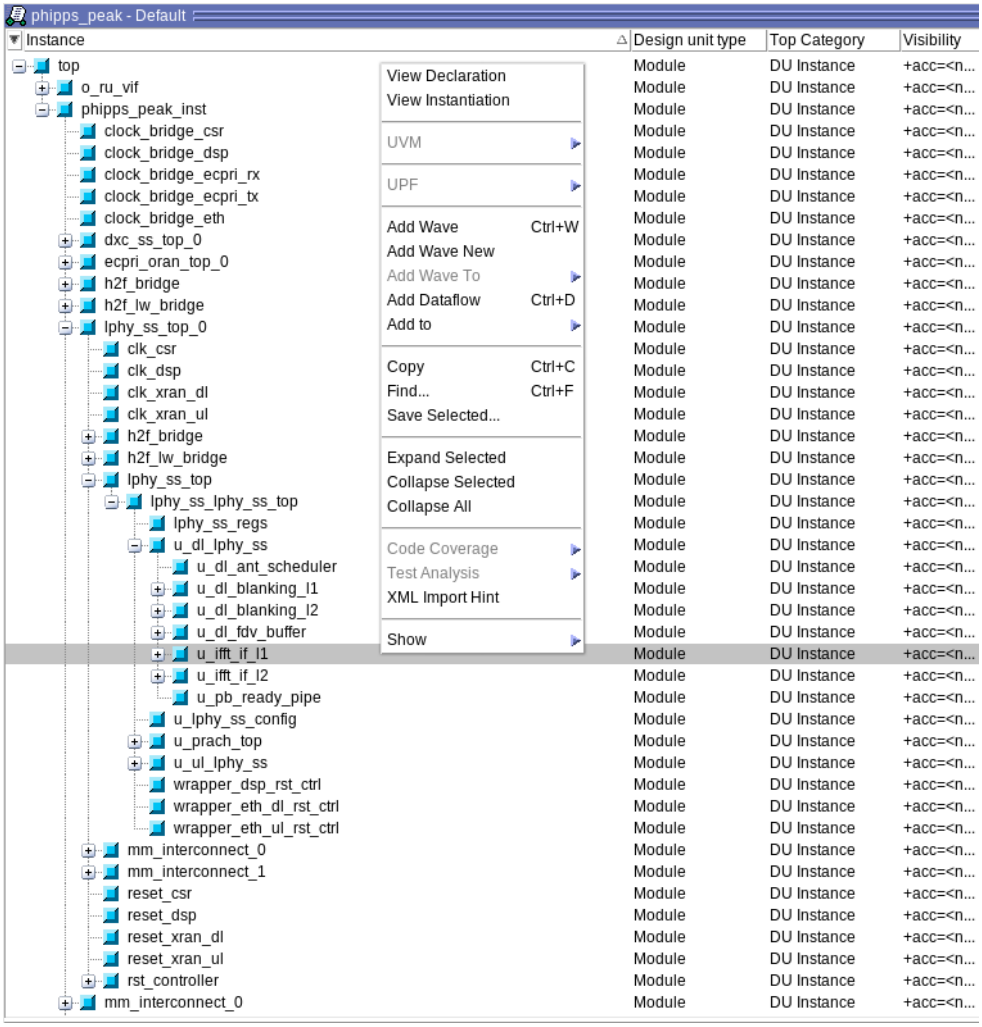


Figure 2-5. Picture of Questasim Waveform window

14. **"run_script.sh"** will contain all the testcases mentioned in **WS-VTP-00072 Sail River GenIII**, to run regression of all the testcases mentioned, execute **"./run_script.sh"**.
15. Execute **"./run_reg.sh"** to get the status of total testcase ran and how many test passed or failed.
16. "**run_reg.sh**" is a bash shell script, if you use CSH need to update the script with **set** command before all the arguments inside the script.

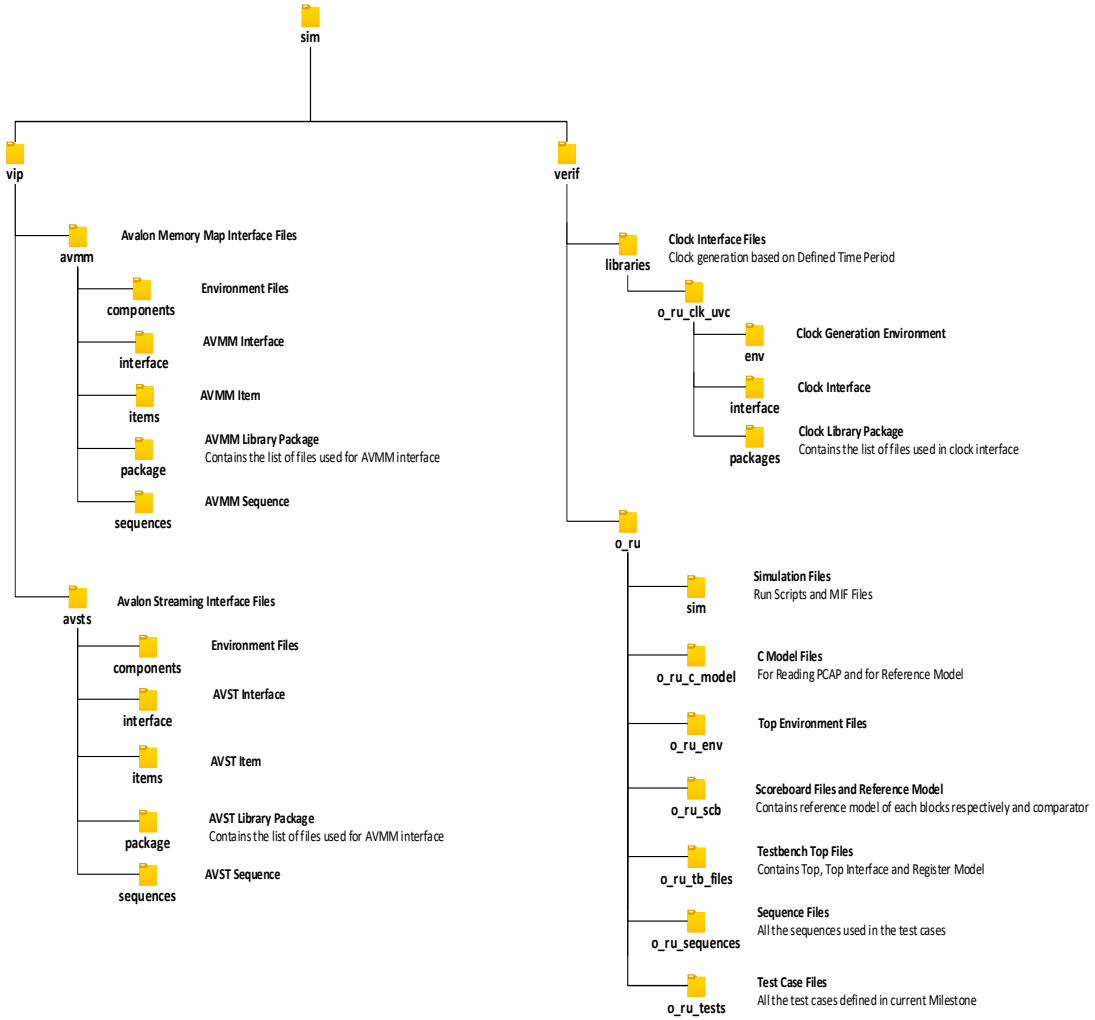
    Example: `set total_cnt=`ls o_ru_tc*_sim.log | wc -l``

2.2.3 IPXACT XML

"PhippsPeak_Top_Subsystem_mmap.xml" contains register description for all Phipps peak registers. Simulation RAL model is generated using this IPXACT XML file.

2.3 Folder Structure

Figure 2-6. Simulation Folder Structure



3.0 DSPBA B2B Simulation

3.1 Objective

The following section describes the requirements for DSPBA back-to-back (b2b) simulation. Major activities described are

- MATLAB model: expected IO generation methodology
- Individual DSPBA model: simulation procedure
- B2B DSPBA simulation strategy

4.0 Simulating MATLAB Model for Expected Vectors

This section describes the generation of expected vectors for RTL verification and DSPBA back-to-back simulation.

4.1 File structure for MATLAB model expected generation

The top-level folder fpga/matlab_scripts for generating vectors are shown below.

Figure 4-1. Folder Structure

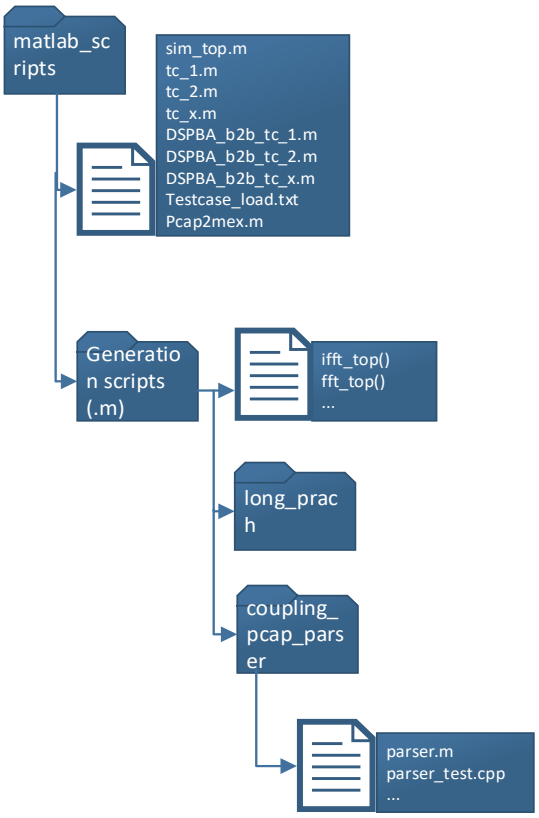


Table 4-1. File Structure

Sl.No.	Folder/Files	Format	Remarks
1	sim_top	.m	Top script to launch vector generation. Used for both RTL vector generation and b2b DSPBA simulation. For RTL vectors,it asks for input from the user whether to generate vectors for a specific testcase(Press 0) or for all testcase(Press 1). For b2b DSPBA vector generation, this script is called from fpga/src/ dspba/sim/top.m. In that case a prompt comes up and asks for testcase number, number of CCs to run and number of symbols to run.
2	tc_1,2,3...	.m	Used for RTL simulation. Contains settings for different testcases. Executed by sim_top depending on which testcase is selected.
3	DSPBA_b2b_tc_1 ,2,3…	.m	Used for DSPBA back-to-back simulation. Contains settings for different testcases. Executed by sim_top depending on which testcase is selected.
5	generation scripts	.m scripts	Contains MATLAB functions for expected vector generation
	generation scripts	long_prach	Contains the necessary .m files required to generate rtl vectors for long prach.
	generation scripts	coupling_pcap_p arser	Contains liblibrary.so file, .cpp files, .m files, .h files required for .mat file extraction from .pcap file (ifft_in extraction from .pcap) .
6	testcase_load	.csv files	Contains the testcase number along with the testcase name and feature like cc disable,lpbk select for RTL vector generation (RTL testcases).
7	pcap2mex	.m file	Top .m file to generate ifft_in from pcap.

4.1.1 Steps to extract expected vectors from MATLAB model

The sim_top.m is the top script for generating expected vectors. It can run for generating RTL verification vectors or for back-to-back simulation vectors. If the script is called from back-to-back simulation top script, a prompt pops up and asks for testcase number, number of CCs to run and number of symbols to run for. In this case vector generation is for back-to-back simulation inside [**OUTPUT_BASEPATH**] /output/b2b_sim/ folder. Otherwise, if sim_top.m is directly triggered it will ask for two options (vector generation for all testcase / vector generation for specific testcase) and create the simulation vectors inside [**OUTPUT_BASEPATH**] /output/rtl_sim under each **[testcase_name]** and under each module names.

Consider the following points:

1. The sim_top.m script has the path name specified in OUTPUT_BASEPATH variable.
2. [testcase_name] will be updated based on the testcase names mentioned in Error! Reference source not found..
3. When we invoke matlab scripts directly,it generates RTL verification vectors in .csv format.

4. When we invoke dspba b2b sim script it will call matlab scripts internally to generate dspba b2b in .mat format.

For DSPBA b2b simulation

Go to DSPBA back-to-back simulation section.

For SIM_vectors generation

5. View testcase_load.txt to know the features enabled for each testcases.
   - testcase_no: Refers to testcase number.
   - testcase_name: Names of testcases are updated based on names mentioned in Simulation VTP document.
   - pcap_folder_name: Names of folder where pcaps and ifft_in.mat are available for each testcase.
   - cc_disable: '1' configured for 1 cc_disable, '0' configured for 2 cc_enable.
   - DFE_lpbk/IFFT_FFT_lpbk: '0' configured for DFE_lpbk, '1' configured for IFFT_FFT_lpbk.

Figure 4-2. testcase_load.txt file

testcase_load.txt

testcase_no	testcase_name	pcap_folder_name	cc_disable	dfe_lpbk/ifft_fft_lpbk
1	tc_c_u_coupling_case_1_compress	tcs_01	0	0
2	tc_c_u_coupling_case_1_compress_60	tcs_02	0	0
3	tc_c_u_coupling_case_1_compress_mixed	tcs_03	0	0
4	tc_delay_compensation	tcs_05	0	0
5	tc_c_u_coupling_case_1_compress_mixed_60_100	tcs_04	0	1
6	tc_prb_blanking_case_1_compression	tcs_07	0	1
7	tc_prb_blanking_case_2_compression	tcs_08	0	1
8	tc_eaxc_blanking_case_1_compression	tcs_06	0	1
9	tc_c_u_coupling_case_1_compress_phasecomp	tcs_01	0	1
10	tc_c_u_coupling_case_1_compress_60_phasecomp	tcs_02	0	1
11	tc_c_u_coupling_case_1_compress_cc2_dis	tcs_01	1	0
12	tc_c_u_coupling_case_1_compress_60_cc2_dis	tcs_02	1	0
13	tc_c_u_coupling_case_1_compress_ifft_fft_lpbk	tcs_01	0	1
14	tc_c_u_coupling_case_1_compress_60_ifft_fft_lpbk	tcs_02	0	1
15	tc_prach_long_format_c_plane	tcs_01	0	0

6. Run the sim_top.m script.
   The sim_top.m script asks for the following inputs from command window.

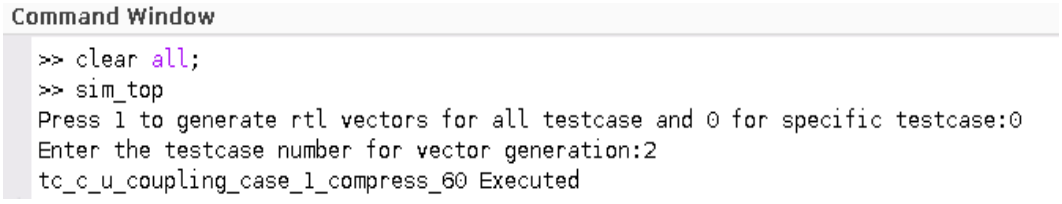
Figure 4-3. Inputs required for sim_top.m

```
Press 1 to generate rtl vectors for all testcase and 0 for specific testcase:0
Enter the testcase number for vector generation:2
```

For example, in the above Error! Reference source not found. we have selected s pecific testcase option, so it asks for testcase_number .tc_c_u_coupling_case_1_compress_60 testcase is loaded.

Wait for the vector generation to be completed.

Figure 4-4. Vector generation completed



7. The ifft_in.mat file will be loaded from **[INPUT_SIM_PATH].**
    - Expected vectors for ifft include:
        - ifft_out.mat
    - Expected vectors for duc include:
        - duc_out.mat
        - mixer_out.mat
    - Expected vectors for carrier aggregator and interpolator include:
        - summer_out.mat
        - interp_out.mat
    - Expected vectors for decimator delay compensation include:
        - decimator_out.mat
    - Expected vectors for ddc include:
        - mixer_out_ddc.mat
        - ddc_out.mat
    - Expected vectors for fft include:
        - fft_out

    All the above-mentioned vectors will be stored under the auto-generated folder structure (i.e.,) [**OUTPUT_BASEPATH**] /output/b2b_sim/.

    The rtl vectors (.csv files) will be stored under the auto-generated folder structure (i.e.,) [**OUTPUT_BASEPATH**]/output/rtl_sim/ under the corresponding **testcase_name** and under each module name.

Note: **[INPUT_SIM_PATH] =** ../../testvector/input/sim/[pcap_folder_name]

5.0 DSPBA Simulation of Individual Model

5.1 Low-Phy DL Model

This block converts the frequency domain IQ data into time domain IQ data, which is suitable for transmission over RF. It contains phase compensation, BRSC, iFFT, gain compensation, CP addition and Block to stream conversion for downlink processing.

5.1.1 File Structure

The following files are present in the directory.

Table 5-1. Low-Phy DL Model File Structure

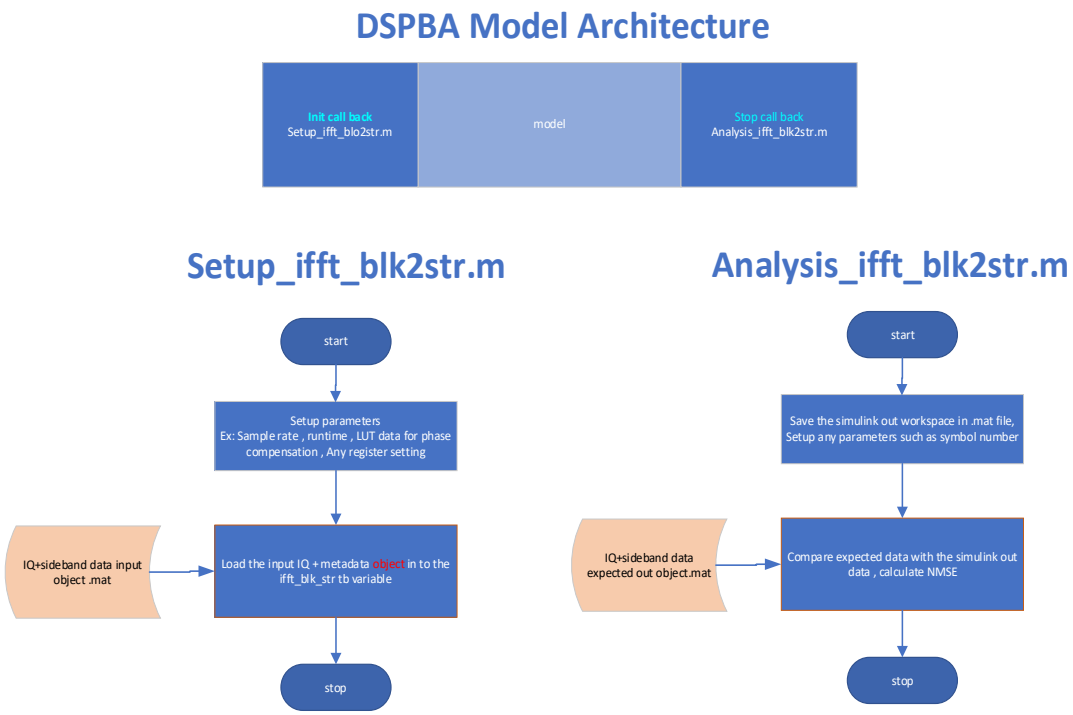
Sl.No.	Type	File Name	Format	Remarks
1	Main Design	Ifft_streamtoblock.slx	Simulink model	Top wrapper file of DSPBA model
2		Lib_lowphy.mdl	Simulink library component	Library component which has signal processing blocks. Will be present inside lib_lowphy folder
3		Setup_ifft_blocktostream.m	MATLAB script	Contains setup scripts, loads IQ data into model
4		TestAnalysis.m	MATLAB script	Contains a script to compare expected and obtained output.
5		local_test_input_output.mat	Mat file/ data file	Contains input excitation vector / expected output vectors
6		Datagen_testcase_Run.m	MATLAB Script	Contains test cases and generated input, expected output data local_test_input_output.mat
7	Support Files	Ifft_blocktostream.slxc, Ifft_blocktostream_params.xml Liblowphy/demo_fsm_control_reads2b.fsm Liblowphy/demo_fsm_control.fsm Liblowphy/ fsmpaths.m		

5.1.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Add lib folder to the path.
3. Open Datagen_testcase_run.m
4. Run Datagen_testcase_run.m file.
5. Enter Testcase ID as '3' when prompted.

6. Datagen_testcase_run.m will
    - internally create test input-output data,
    - stores it in local_test_input_output.mat
7. Open and run ifft_blocktostream.slx
    - Internally, initialization scripts call Setup_ifft_blocktostream.m and load the data present in local_test_input_output.mat into the model.
    - Then RTL is generated in DSPBA/rtl and DSPBA model is simulated.
    - Internally, post-run scripts call TestAnalysis and display the results.

Figure 5-1. DSPBA Model Architecture



5.1.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.1.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation

5.2 Low-Phy UL Model

This block converts the time domain IQ data into frequency domain IQ data, which is received over RF. It contains phase compensation, BRSC, FFT, CP removal and stream to block conversion used in uplink processing.

5.2.1 File Structure

The following files are present in the directory.

Table 5-2. Low-Phy UL Model File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	Streamtoblock_fft.mdl	Simulink model	Top wrapper file of DSPBA model
2		Lib_lowphy.mdl	Simulink library component	Library component which has signal processing blocks. Will be present inside lib_lowphy folder
3		Setup_streamtoblock_fft.m	MATLAB script	Contains setup scripts and loads IQ data into model
4		TestAnalysis.m	MATLAB script	Contains a script to compare expected and obtained output.
5		local_test_input_output.mat	Mat file/ data file	Contains input excitation vector / expected output vectors
6		Datagen_testcase_Run.m	MATLAB Script	Contains test cases, it generates local_test_input_output .mat,
7	Support Files	streamtoblock_fft.slxc, streamtoblock_fft.xml Liblowphy/demo_fsm_control_reads2b.fsm Liblowphy/demo_fsm_control.fsm Liblowphy/ fsmpaths.m		

5.2.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Add lib_lowphy folder to the path.
3. Open Datagen_testcase_run.m
4. Run Datagen_testcase_run.m file.
5. Enter Testcase ID as '3' when prompted.
6. Datagen_testcase_run.m will
    - internally create test input-output data,
    - stores it in local_test_input_output.mat

7. Open and run streamtoblock_fft.slx
    - Internally, initialization scripts call Setup_streamtoblock.m and load the data present in local_test_input_output.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run scripts call TestAnalysis and display the results.

5.2.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.2.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

5.3 DUC Model

We are splitting the earlier DUC design into two parts: duc_model (channel filter+interpolator+mixer) and ca_interp (summer+poly phase interpolator) to support 1CC /2CC compile time configuration.

This block, named DUC, is built for Channel filtering; the Channel filter output is interpolated by 4 by a cascade of two half-band filters before the complex mixing. The stream of the digitally-upconverted samples is fed to the Numerically Controlled Oscillator (NCO), Complex mixer Modules.

5.3.1 File Structure

The following files are present in the directory.

Table 5-3. DUC Model File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	duc_model.mdl	Simulink model	Top wrapper file of DSPBA model
2		setup_duc_model.m	MATLAB script	Contains setup scripts and loads ifft_out data into model
3		DUC_Output_analysis.m	MATLAB script	Contains a script to compare expected and obtained output.
4		tmwave_to_NRinput.m	MATLAB script	Converts tmwave to NRinput
5		Settings.mat	Mat file/ data file	Contains Setting for duc_model

6		ifft_out.mat	Mat file/ data file	Contains input excitation vector
7		mixer_out.mat, duc_out.mat	Mat file/ data file	Contains expected output vectors
8		NMSE_NPSE_calc.m	MATLAB Script	Contains MATLAB function to calculate NMSE and NPSE
9	Support Files	duc_model.slxc, duc_model_params.xml		

5.3.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Open and run duc_model.mdl
    - Internally, initialization script call setup_duc_model.m and load the settings present in Settings.mat and loads converted data from generated inputs60100.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run script call DUC_Output_analysis.m and display the results.

5.3.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.3.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

5.4 Carrier Aggregation and Interpolator Model

The NCO-tuned data from the Complex mixer modules are then summed up through a series of adder stages. The final stage of Poly-phase HB interpolator up-samples each antenna.

5.4.1 File Structure

The following files are present in the directory.

Table 5-4. CA-Interp Model File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	ca_interp.mdl	Simulink model	Top wrapper file of DSPBA model
2		setup_ca_interp.m	MATLAB script	Contains setup scripts and loads ifft_out data into model
3		ca_interp_output_analysis.m	MATLAB script	Contains a script to compare expected and obtained output.
4		Settings.mat	Mat file/ data file	Contains Setting for duc_model
5		mixer_out.mat	Mat file/ data file	Contains input excitation vector
6		summer_out.mat, interpolator_out.mat	Mat file/ data file	Contains expected output vectors
7		NMSE_NPSE_calc.m	MATLAB Script	Contains MATLAB function to calculate NMSE and NPSE
8	Support Files	ca_interp_params.xml ca_interp.slxc		

5.4.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Open and run ca_interp.mdl
    - Internally, initialization script call setup_ca_interp.m and loads the settings present in Settings.mat and loads data from mixer_out.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run script call ca_interp_output_analysis.m and display the results.

5.4.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.4.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

5.5 Decimator Delay Compensation Model

We are splitting DDC design into dec_dly_comp (polyphase decimator + bypass + ant_gain_delay) and ddc_module (complex mixer + DDC + channel filter) to support 1CC /2CC compile time configuration.

5.5.1 File Structure

The following files are present in the directory.

Table 5-5. Dec_dly_comp File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	dec_dly_comp.mdl	Simulink model	Top wrapper file of DSPBA model
2		setup_dec_dly_comp.m	MATLAB script	Contains setup scripts and loads ifft_out data into model
3		dec_dly_comp_output_analysis.m	MATLAB script	Contains a script to compare expected and obtained output.
4		Settings.mat	Mat file/ data file	Contains Setting for duc_model
5		interpolator_out.mat	Mat file/ data file	Contains input excitation vector
6		decimator_out.mat	Mat file/ data file	Contains expected output vectors
7		NMSE_NPSE_calc.m	MATLAB Script	Contains MATLAB function to calculate NMSE and NPSE
8	Support Files	dec_dly_comp.slxc, dec_dly_comp_params.xml		

5.5.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Open and run dec_dly_comp.mdl
    - Internally, initialization script call setup_dec_dly_comp.m and loads the settings present in Settings.mat and loads data from generated interpolator_out.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run script call dec_dly_comp_output_analysis.m and display the results.

5.5.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.5.4 Steps to run auto test bench simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

5.6 DDC Model

We are splitting the earlier DDC design into two parts: dec_dly_comp (polyphase decimator + bypass + ant_gain_delay) and ddc_module (complex mixer + DDC + channel filter) to support 1CC /2CC compile time configuration.

This section describes details of ddc_module, now called DDC.

5.6.1 File Structure

The following files are present in the directory.

Table 5-6. DDC File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	ddc_module.mdl	Simulink model	Top wrapper file of DSPBA model
2		setup_ddc_model.m	MATLAB script	Contains setup scripts and loads ifft_out data into model
3		ddc_output_analysis.m	MATLAB script	Contains a script to compare expected and obtained output.
4		Settings.mat	Mat file/ data file	Contains Setting for duc_model
5		decimator_out.mat	Mat file/ data file	Contains input excitation vector
6		ddc_out.mat, mixer_out_ddc.mat	Mat file/ data file	Contains expected output vectors
7		NMSE_NPSE_calc.m	MATLAB Script	Contains MATLAB function to calculate NMSE and NPSE
8	Support Files	ddc_model.slxc, ddc_model_params.xml		

5.6.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Open and run ddc_model.mdl
    - Internally, initialization script call setup_ddc_model.m and loads the settings present in Settings.mat and loads data from generated decimator_out.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run script call ddc_output_analysis.m and display the results.

5.6.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.6.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

5.7 Long PRACH Model

This block converts the time domain IQ data into frequency domain IQ data, which is received over RF. It supports Long PRACH format-0.

It contains FFT, CP removal, decimation, mixer and stream to block conversion used in uplink PRACH processing.

5.7.1 File Structure

The following files are present in the directory.

Table 5-7. Long PRACH File Structure

Sl.No.	Type	File Name	Format	Remarks
1	Main Design	LongPRACH_sim.slx	Simulink model	Top wrapper file of DSPBA model
2		Lib_LongPRACH.mdl	Simulink library component	Library component which has signal processing blocks. Will be present inside lib folder
3		Setup_longPRACH.m	MATLAB script	Contains setup scripts loads local_test_input_output.mat into model

4		Prach_Analysis.m	MATLAB script	Contains a script to compare expected and obtained output.
5		local_test_input_output .mat	Mat file/ data file	Contains input excitation vector / expected output vectors
6		Datagen_testcase.m	MATLAB Script	Contains test cases, it generates local_test_input_output.mat
7	Support Files	longPRACH .slxc, longPRACH.xml Load_usecases.m		

5.7.2 Steps to run DSPBA Simulation

1. Open the folder in MATLAB.
2. Add lib folder to the path.
3. Open Datagen_testcase_run.m
4. Run Datagen_testcase_run.m file.
5. Enter Testcase ID as '1' when prompted.
6. Datagen_testcase_run.m will
    - internally create test input-output data,
    - stores it in local_test_input_output.mat
7. Open and run longprach_sim.slx
    - Internally, initialization scripts call Setup_longprach.m and load the data present in local_test_input_output.mat into the model.
    - Then RTL is generated and DSPBA model is simulated.
    - Internally, post-run scripts call TestAnalysis and display the results.

5.7.3 Steps to Generate RTL

When the model is run using the above steps RTL is generated in RTL folder.

5.7.4 Steps to run auto test bench Simulation

1. Ensure Generate RTL & Generate ATB option is enabled in model.
2. Run the DSPBA simulation as explained above.
3. Launch quartus from the DSPBA model (by double clicking on quartus symbol present in model).
4. Ensure the simulation tool is selected as questasim/alterafpga in EDA editor.
5. Run fitter and eda netlistwriter in quartus tool.
6. Click on dspbuilder->RTL simulation->device to initiate the RTL simulation.

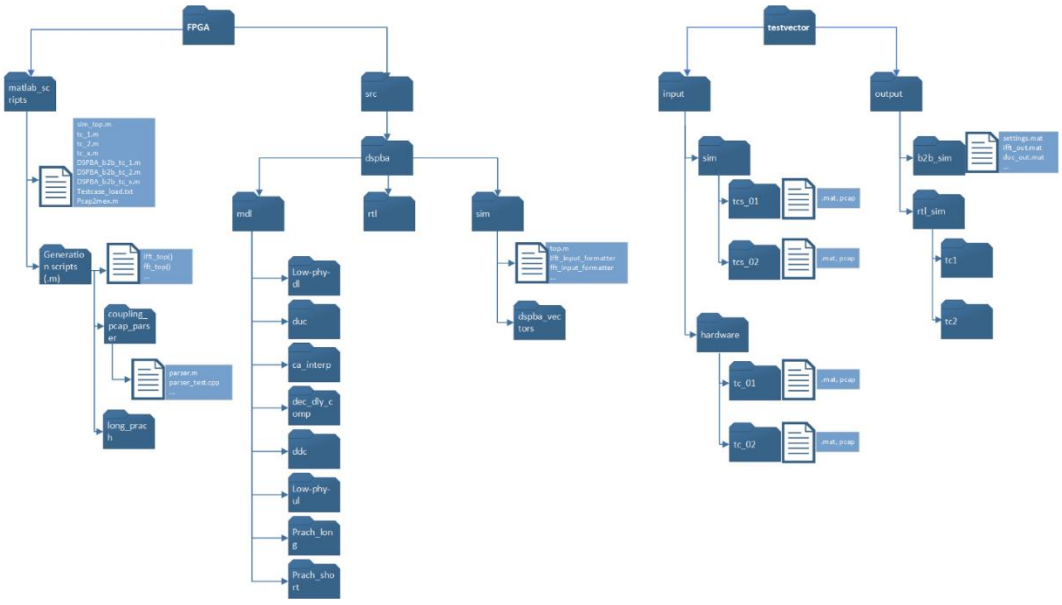
6.0 DSPBA back-to-back Simulation

In this section, we will be simulating all the DSPBA modules explained in above section (low-phy DL, DUC, DDC, low-phy UL) back-to-back. Output of the chain will be compared against the MATLAB floating point model. Top.m is the top file to trigger the simulation. The following are the top-level steps followed:

1. Running the MATLAB vectors top file to generate each module's expected vectors and settings.
2. Running the modules one by one, starting from iFFT all the way to FFT.
3. Storing output of each module to be used by next module.

6.1 Folder Structure used for back-to-back Simulation

Figure 6-1. Folder Structure used for back-to-back Simulation



The top script for running back-back simulation is top.m present inside src/dspba/sim. There are other supporting scripts by the name of formatters (e.g. ifft_input_formatter). anext block. Apart from that there is a folder named dspba_vectors. This stores the output vector of each DSPBA subsystem. Formatter scripts fetch vectors from this folder for conversion. The folder location for expected vectors is parametrized and stored inside locate_testvector.txt file inside sim/. Please do not edit this file for proper functioning of the flow. The location for expected vectors as specified in locate_testvector.txt is testvectors/output/b2b_sim/.

6.2 Top-Level Script

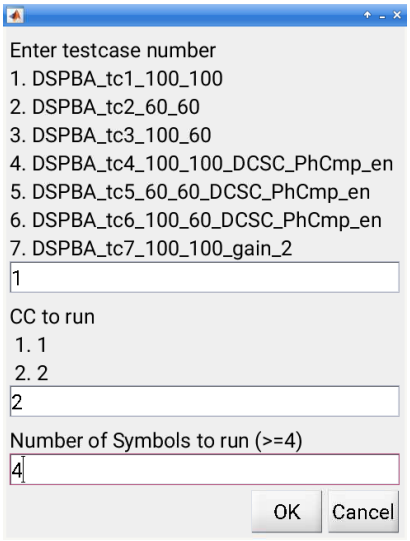
fpga/src/DSPBA/sim/top.m is the top file for running b2b simulation. This top script will execute a sequence of steps as follows:

Figure 6-2. Flow Diagram

Steps to run b2b Simulation

1. Go to src/DSPBA/sim/ folder. Make sure this is the current folder.
2. Open and run top.m script.
3. When prompted, enter the testcase number, number of CCs to run and number of symbols to run for.

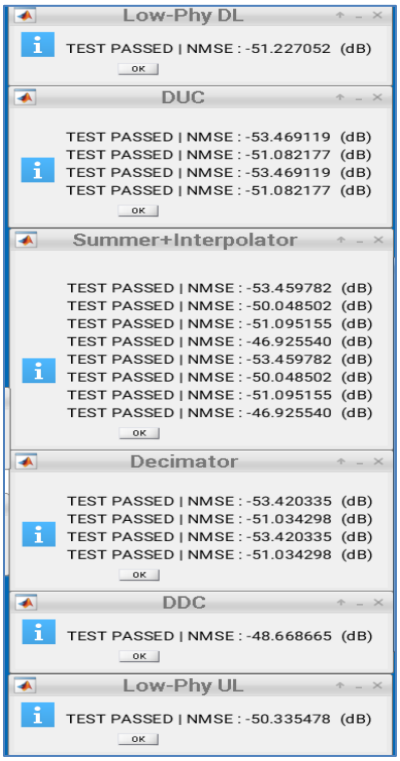
Figure 6-3. Simulation Fields



4. The simulation will start.
5. Internally the following steps will be followed:
   - sim_top.m is called to generate expected vectors and settings for the selected testcase.
   - Vectors generated are stored in folder testvectors/output/b2b_sim/.

- ifft_input_formatter.m will take in settings and vectors from b2b_sim/ folder and convert it into DSPBA compatible input. This converted data is stored into DSPBA/mdl/low_phy_dl.
- Low_phy DSPBA is triggered, and its output is compared against expected vector. This output is saved in /dspba/sim/dspba_vectors to be used for input of next block.
- Steps c and d are followed for DUC, ca_interp, dec_dly_comp, DDC and FFT as well.
- At each step a dialog box pop-up will show the NMSE and if the test is passed.

Figure 6-4. Simulation Test results Template



6. At the end of simulation, constellation will be plotted, and log file generated in the same sim/ folder.

6.3 Simulation Analysis

Two functionalities have been added to analyze the results of b2b simulation:

6.3.1 Log File

As the b2b simulation runs, it creates a log file with the name b2b_log.txt. It has the details including:

- Testcase run

- Simulation length (Number of symbols)
- Time elapsed at each stage
- NMSE observed at each stage

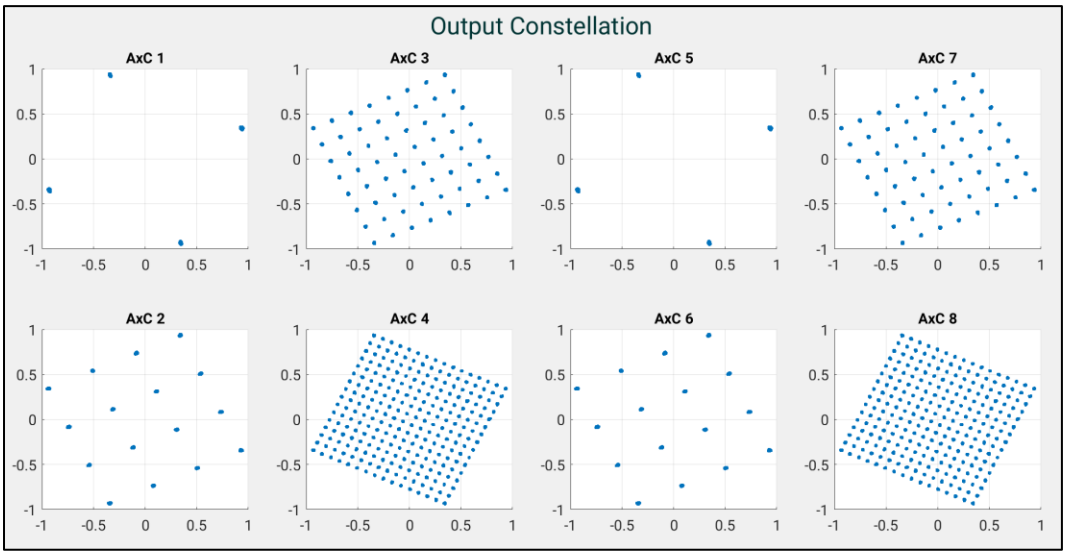
Figure 6-5. Log Template

```
1  ------------------------------------------------
2  DSPBA_b2b_version: 2.2
3  ------------------------------------------------
4  Version description: Additional inputs like CC_run and NumSym also made to be inserted in user prompt
5
6  Simulation Started for 1 CC...
7
8  ------------------------------------------------
9  Simulation length selected
10 ------------------------------------------------
11 Time Elapsed:  0.0 mins
12 Number of Symbols: 4
13
14 ------------------------------------------------
15 Vector generation done
16 ------------------------------------------------
17 Testcase selected: DSPBA_tc1_100_100
18 Time Elapsed:  3.7 mins
19
20 ------------------------------------------------
21 iFFT CC1 simulated
22 ------------------------------------------------
23 Time Elapsed: 13.1 mins
24 iFFT NMSE: -53.4465
25
26 ------------------------------------------------
27 DUC CC1 simulated
28 ------------------------------------------------
29 Time Elapsed: 17.3 mins
30 DUC NMSE: [-54.7237 -53.1557 -54.7237 -53.1557]
```

6.3.2 Constellation

The output of FFT is plotted on a scatterplot. This shows the constellation for all the 8AxCs. It can be caught if the NMSE is good but there is no mistake in the vector generation itself. The correct simulation would give plot as in the following Error! R eference source not found..

Figure 6-6. Constellation Template



6.4 File format conversions from MATLAB ip/op to DSPBA ip/op

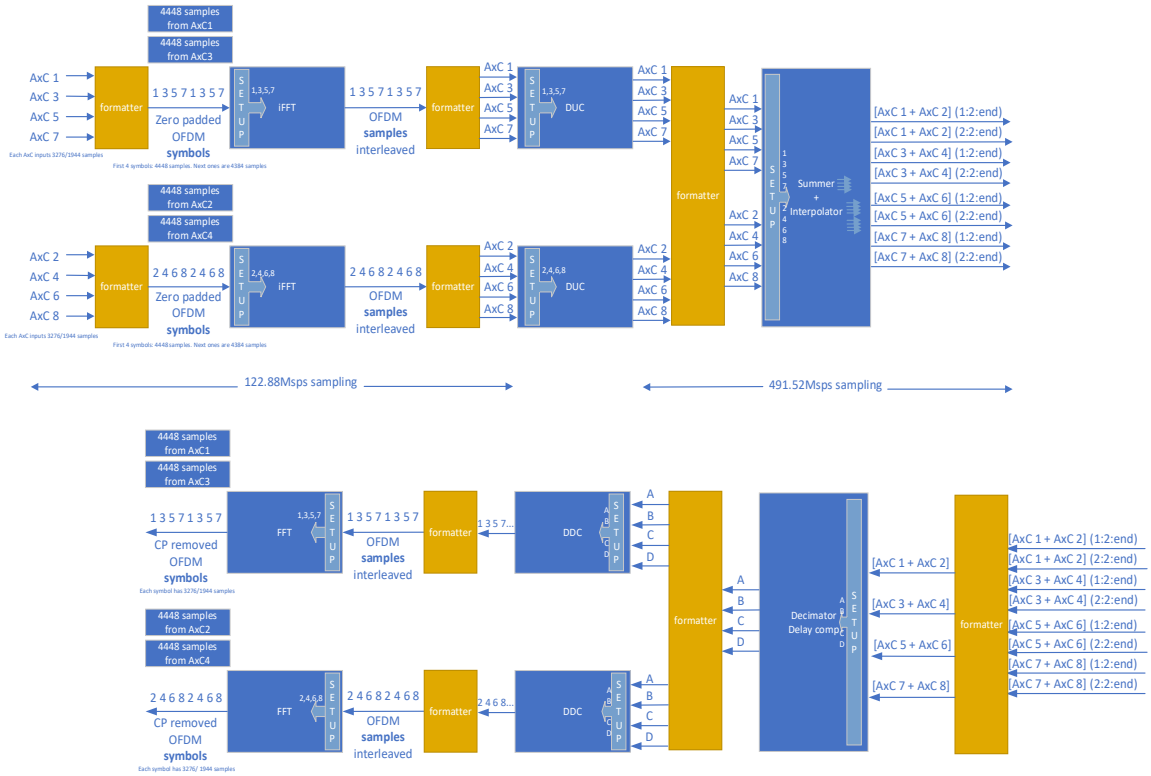
The sim folder has one conversion script for each DSPBA model. The scripts create the input in the format expected by the DSPBA standalone models inside the mdl folder. The script then saves these inputs to the respective standalone model's folder.

Table 6-1. File format conversions from MATLAB ip/op to DSPBA ip/op

Input of conversion process		Conversion Script	Output of conv process (Stored in corresponding module's folder)	Details of Objects	Remarks
Settings for creating sideband settings data (from MATLAB script)	IQ data (output of MATLAB model or of previous DSPBA model)				
	ifft_in.mat (MATLAB model)	ifft_input_formatter.m	ifft_tb_in_local.mat	Appendix A	Input for ifft_blockto_stream model
	ifft_out.mat (MATLAB model)		ifft_tb_exp_out_local.mat		Input for DL test analysis to get NMSE

settings.mat	Data output of iFFT DSPBA model	duc_input_formatter.m	ifft_out.mat		Input for DUC model's setup file
	mixer_out.mat (MATLAB model)		mixer_out.mat		Input for DUC test analysis
	Data output of DUC DSPBA model	ca_interp_input_formatter	mixer_out.mat		Input for ca_interp model's setup file
	interpolator_out.mat (MATLAB model)		interpolator_out.mat		Input for ca_interp test analysis
	Data output of ca_interp DSPBA model	dec_dly_comp_input_formatter.m	interpolator_out.mat		Input for dec_dly_comp model's setup file
	decimator_out.mat		decimator_out.mat		Input for dec_dly_comp test analysis
	Data output of dec_dly_comp DSPBA model	ddc_input_formatter.m	decimator_out.mat		Input to DDC model's setup file
	ddc_out.mat		ddc_out.mat		Input to DDC test analysis
	Data output of DDC DSPBA model	fft_input_formatter.m	fft_tb_in_local		Input to FFT model's setup file
	ddc_out.mat		fft_tb_exp_out_local		Input to FFT test analysis

Figure 6-7. Data Flow



7.0 DSPBA settings, input-output format details

7.1 Settings definitions: used by MATLAB model and conversion scripts, not by DSPBA models directly

This section defines the objects that are passed to MATLAB model for settings. They come from the testcase definition files (like DSPBA_tc1_100_100.m).

These parameters are saved inside a single settings.mat file. Different structs for each DSPBA model are saved inside this settings.mat. This will be used by formatter scripts as input later.

7.1.1 IFFT settings

Table 7-1. IFFT settings

Sl.No.	Parameter	Description	Valid Values	Default Values
1	lut	LUT values		[2864.58333333333 38541.6666666667 74218.7500000000 109895.833333333 145572.916666667 181250.000000000 216927.083333333 252604.166666667 288281.250000000 323958.333333333 359635.416666667 395312.500000000 430989.583333333 466666.666666667]*1e9
2	fftsize_in	FFT size	4096	4096
3	NPRB	BW selection	3276 and 1944	3276
4	constgain	Gain on data	1	1
5	Muxsel	Data selection	0 or 1	0
6	MuxConstData	This data is selected if Muxsel=1	Data for iFFT	0
7	ifftshift		12	12
8	DC_SC_EN	DC sub-carrier enable/disable	0 or 1	0
9	Tech_CP	Phase compensation enable/disable	0 or 1	0
10	Digital_power_real	Complex gain	1-2^-14	1-2^-14
11	Digital_power_imag	Complex gain	0	0

12	F1	Center frequency for phase compensation	3.7487	3.7487
13	Tsym	Reserved	0	0
14	TCP	Reserved	0	0
15	CPLen_in	CP length	352	352

7.1.2 DUC Settings

Table 7-2. DUC Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	bw_sel	Bandwidth (MHz)	60e6/100e6	100e6
2	duc_nco	NCO frequency values		[50 -50 -50 50 50 -50 -50 50]
3	HB1_coeffs	HB filter1 coefficients	71x1	
4	HB2_coeffs	HB filter2 coefficients	21x1	
5	Coef100	Channel filter coefficients for 100MHz BW	1x311	
6	Coef60	Channel filter coefficients for 60MHz BW	1x311	
7	bw_cc_config	CC config based on bw_sel (added for DSPBA model)	60MHz-10 / 100MHz-14	60MHz-10 100MHz-14
8	Chan_filter_coeffs	Coeff100/Coeff60 in 5 and 6 will be loaded according to bw_sel.	Coef60/Coef100	Coef100

7.1.3 Carrier Aggregator and Interpolator Settings

Table 7-3. CA-Interp Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	HB3_coeffs	HB3 filter coeffecients	81x1	

Table 7-4. CA-Interp Additional Settings

Sl.No	Parameters	Description	Valid values	Default values
1	duc_gain	gain parameters	1x4	[8192 8192 8192 8192]
2	duc_cc1_delay	Line up 1 delay	1x4	[0 0 0 0]
3	duc_cc2_delay	Line up 2 delay	1x4	[0 0 0 0]

7.1.4 Decimator Delay Compensation Settings

Table 7-5. dec_dly Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	HB3_coeffs	HB3 filter coeffecients	81x1	

Table 7-6. dec_dly additional Settings

Sl.No	Parameters	Description	Valid values	Default values
1	ddc_gain	gain parameters	1x4	[8192 8192 8192 8192]
2	ddc_cc1_delay	Lineup 1 delay	1x4	[0 0 0 0]
3	ddc_cc2_delay	Lineup 2 delay	1x4	[0 0 0 0]

7.1.5 DDC Settings

Table 7-7. DDC Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	bw_sel	Bandwidth (MHz)	60e6/100e6	100e6
2	ddc_nco	NCO frequency values		[50 -50 -50 50 50 -50 -50 50]
3	HB1_coeffs	HB filter1 coefficients	71x1	
4	HB2_coeffs	HB filter2 coefficients	21x1	
5	Coef100	Channel filter coefficients for 100MHz BW	1x311	
6	Coef60	Channel filter coefficients for 60MHz BW	1x311	

7	bw_cc_config	CC config based on bw_sel (added for dspb model)	60MHz-10 / 100MHz-14	60MHz-10 / 100MHz-14
8	Chan_filter_coeffs	Coeff 100/Coeff60 in 5 and 6 will be loaded according to bw_sel.	Coef60/Coef100	Coef100

Table 7-8. DDC additional Settings

Sl.No	Parameter	Description	Valid values	Default values
1	ddc_mux_const_real	ddc mux constant real values	1x4	[0 0 0 0]
2	ddc_mux_const_imag	ddc mux constant imag values	1x4	[0 0 0 0]

7.1.6 FFT Settings

Table 7-9. FFT Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	lut	LUT values		[2864.58333333333 38541.6666666667 74218.7500000000 109895.833333333 145572.916666667 181250.000000000 216927.083333333 252604.166666667 288281.250000000 323958.333333333 359635.416666667 395312.500000000 430989.583333333 466666.666666667]
2	fftsize_in	FFT size	12	12
3	Nprb_v	BW selection	3276 and 1944	3276
4	Const_gain_v	Gain on data	1	1
5	Dout_v	Input data for FFT	Data for iFFT	0
6	Ifftshift_v		0	0
7	DC_SC_EN	DC subcarriers enable/disable	0 or 1	0

8	Tech_CP	Phase compensation enable/disable	0 or 1	0
9	CPLen_in	CP length	352	352
10	F1	Center frequency for phase compensation	3.7487	3.7487
11	Tsym	Reserved	11010	11010
12	Hcsshift_v	Reserved	0	0

7.1.7 Long PRACH Settings

Table 7-10. Long PRACH Settings

Sl.No.	Parameter	Description	Valid values	Default values
1	Sequence index	Sequence index of Zadoff Chu polynomial	Table 6.3.3.1-3: Mapping from logical index i to sequence number u for preamble formats with LRA = 839.	53
2	Preamble index	Sequence index of Zadoff Chu polynomial	Table 6.3.3.1-3: Mapping from logical index i to sequence number u for preamble formats with LRA = 839.	23
3	Frequency offset	Rb offset	0-269	0
4	Config idx	Time indication	Table 6.3.3.2-3: Random access configurations for FR1 and unpaired spectrum.	27
5	snr	RF signal SNR		
6	technology	4G/5G selection	1=5G 0=4G	1

7.2 Data IO Ports Definition

This defines the interface between DSPBA and the external environment.

- DSPBA can get this data from MATLAB model.
- DSPBA can get this data from back2back conv scripts.

This section defines the objects that are passed to DSPBA model as input/output. These will be used by DSPBA_b2b script while invoking DSPBA model.

7.2.1 IFFT Module

7.2.1.1 Input to IFFT module

Top object

ifft_tb_in

Data elements in object

ifft_tb_in.valid_in

ifft_tb_in.ChNo_in

ifft_tb_in.CPLen_in

ifft_tb_in.fftsize_in

ifft_tb_in.NPRB

ifft_tb_in.constgain

ifft_tb_in.Muxsel

ifft_tb_in.MuxConstData

ifft_tb_in.ifftshift

ifft_tb_in.DC_SC_EN

ifft_tb_in.Tech_CP

ifft_tb_in.Tsym

ifft_tb_in.TCP

ifft_tb_in.Digital_power_real

ifft_tb_in.Digital_power_imag

ifft_tb_in.dummyones_in

ifft_tb_in.F1

7.2.1.2 Output of IFFT module

DLpath_Tap_data_out

7.2.2 DUC Module

7.2.2.1 Input to DUC module

Top object.

DUC_LINEUP_param

Elements in the object.

Table 7-11. DUC Lineup Parameters

Parameter	MATLAB settings mapping	Format	Description
DUC_LINEUP_param. bw_select	duc_tb_in.bw_sel	100e6/ 60e6	100MHz or 60MHz bandwidth select
DUC_LINEUP_param. NCO1.Freq	nco_freq_tmp = duc_tb_in.duc_nco /1e6; [nco_freq_tmp(AXC_SEL (1)) nco_freq_tmp(AXC_SEL (2)) nco_freq_tmp(AXC_SEL (3)) nco_freq_tmp(AXC_SEL (4))];	4x1	Based on the AXC NCO frequencies are selected For example: AXC_SEL= [1,3,5,7] for 100MHz Bandwidth select then NCO1.freq =[duc_nco(1), duc_nco (3), duc_nco (5), duc_nco (7)]
DUC_LINEUP_param. HB_filt1.coeffs	duc_tb_in.HB1_coeffs	71x1	Interpolator HB1 filter coefficient
DUC_LINEUP_param. HB_filt2.coeffs	duc_tb_in.HB2_coeffs	21x1	Interpolator HB2 filter coefficient
DUC_LINEUP_param. chan_filter.coeffs(1,:)	duc_tb_in.Coef60	311x1	Chan filter coefficients for 60MHz
DUC_LINEUP_param. chan_filter.coeffs(2,:)	duc_tb_in.Coef100	311x1	Chan filter coefficients for 100MHz
DUC_LINEUP_param. inputdata_l1	inputdata_60_100	sfix16_14	tmwave_to_NRinput converted of ifft_out.mat input to duc_model
DUC_LINEUP_param. BW_Config_CC1	duc_tb_in.bw_cc_config	14/10	CC1 is configured for 100 MHz/60MHz

7.2.2.2 Output of DUC module

Mixer_dOut

7.2.3 Carrier Aggregator and Interpolator

7.2.3.1 Input to CA-Interp

Top object.

DUC_LINEUP_param

Elements in the object

Table 7-12. CA-Interp Lineup Parameters

Parameter	MATLAB mapping	Format	Description
DUC_LINEUP_param. HB_filt3.coeffs	ca_interp_tb_in.HB3_coeffs	81x1	Polyphase filter coefficient
DUC_LINEUP_param. inputdata_l1	DUC_LINEUP_param.inputdata_l1 (:,1) = mixer_out(:,AXC_SEL (1)) DUC_LINEUP_param.inputdata_l1 (:,2) = mixer_out(:,AXC_SEL (2)) DUC_LINEUP_param.inputdata_l1 (:,3) = mixer_out(:,AXC_SEL (3)) DUC_LINEUP_param.inputdata_l1 (:,4) = mixer_out(:,AXC_SEL (4))	sfix16_14 (datalen x 4)	mixer output.mat file provide 8 AXC mixer_out data, in that based on AXC_SEL input is assigned. For example: AXC_SEL = [1,3,5,7] for 100MHz Bandwidth select then DUC_LINEUP_param.inputdata_l1 (:,1) = mixer_out(:,1); DUC_LINEUP_param.inputdata_l1 (:,2) = mixer_out(:,3); DUC_LINEUP_param.inputdata_l1 (:,3) = mixer_out(:,5); DUC_LINEUP_param.inputdata_l1 (:,4) = mixer_out(:,7);

DUC_LINEUP_param. inputdata_l2	DUC_LINEUP_param.inputdata_l2 (:,1) = mixer_out(:,AXC_SEL (1)) DUC_LINEUP_param.inputdata_l2 (:,2) = mixer_out(:,AXC_SEL (2)) DUC_LINEUP_param.inputdata_l2 (:,3) = mixer_out(:,AXC_SEL (3)) DUC_LINEUP_param.inputdata_l2 (:,4) = mixer_out(:,AXC_SEL (4))	sfix16_14 (datalen x 4)	mixer output.mat file provide 8 AXC mixer_out data, in that based on AXC_SEL input is assigned. For example: AXC_SEL = [2,4,6,8] for 60MHz Bandwidth select then DUC_LINEUP_param.inputdata_l2 (:,1) = mixer_out(:,2); DUC_LINEUP_param.inputdata_l2 (:,2) = mixer_out(:,4); DUC_LINEUP_param.inputdata_l2 (:,3) = mixer_out(:,6); DUC_LINEUP_param.inputdata_l2 (:,4) = mixer_out(:,8);

7.2.3.2 Output of CA-Interp

PolyPhase0_dOut

7.2.4 Decimator Delay Compensation Module

7.2.4.1 Input to dec_dly_comp module

Top object.

DDC_LINEUP_param

Elements in the object.

Table 7-13. dec_dly_comp Lineup Parameters

Parameter	MATLAB mapping	Format	Description
DDC_LINEUP_param. HB_filt3.coeffs	dec_dly_comp_tb_in.HB3_coeffs	81x1	Polyphase filter coefficient
DDC_LINEUP_param.DDC_Ant_Data_in_1 DDC_LINEUP_param.DDC_Ant_Data_in_2	DDC_LINEUP_param.DUC_A1 = interp_out(:,1); DDC_LINEUP_param.DDC_Ant_Data_in_1 = DDC_LINEUP_param.DUC_A1(1:2:end);	sfix16_14 (datalen x 1)	Interpolator_out.mat is fed as input of dec_dly_comp model

	DDC_LINEUP_param.DDC_Ant_Data_in_2 = DDC_LINEUP_param.DUC_A1(2:2:end);		
DDC_LINEUP_param.DDC_Ant_Data_in_3 DDC_LINEUP_param.DDC_Ant_Data_in_4	DDC_LINEUP_param.DUC_A3 = interp_out(:,2); DDC_LINEUP_param.DDC_Ant_Data_in_3 = DDC_LINEUP_param.DUC_A3(1:2:end); DDC_LINEUP_param.DDC_Ant_Data_in_4 = DDC_LINEUP_param.DUC_A3(2:2:end);	sfix16_14 (datalen x 1)	
DDC_LINEUP_param.DDC_Ant_Data_in_5 DDC_LINEUP_param.DDC_Ant_Data_in_6	DDC_LINEUP_param.DUC_A5 = interp_out(:,3); DDC_LINEUP_param.DDC_Ant_Data_in_5 = DDC_LINEUP_param.DUC_A5(1:2:end); DDC_LINEUP_param.DDC_Ant_Data_in_6 = DDC_LINEUP_param.DUC_A5(2:2:end);	sfix16_14 (datalen x 1)	
DDC_LINEUP_param.DDC_Ant_Data_in_7 DDC_LINEUP_param.DDC_Ant_Data_in_8	DDC_LINEUP_param.DUC_A7 = interp_out(:,4); DDC_LINEUP_param.DDC_Ant_Data_in_7 = DDC_LINEUP_param.DUC_A7(1:2:end); DDC_LINEUP_param.DDC_Ant_Data_in_8 = DDC_LINEUP_param.DUC_A7(2:2:end);	sfix16_14 (datalen x 1)	

7.2.4.2 Output of dec_dly_comp

decimator_out_DSPBA1

7.2.5 DDC Module

7.2.5.1 Input to DDC module

Top object.

DDC_LINEUP_param

Elements in the object.

Table 7-14. DDC Lineup Parameters

Parameter	MATLAB mapping	Format	Description
DDC_LINEUP_param. bw_select	ddc_tb_in.bw_sel	100e6/ 60e6	100MHz or 60MHz bandwidth select

DDC_LINEUP_param.Complex_NCO1.Freq	nco_freq_tmp = ddc_tb_in.ddc_nco1e6;
	
	[
	
	nco_freq_tmp(AXC_SEL (1))
	nco_freq_tmp(AXC_SEL (2))
	nco_freq_tmp(AXC_SEL (3))
	nco_freq_tmp(AXC_SEL (4))];
		4x1	Based on the AXC NCO frequencies are selected
			
			For example:
			
			AXC_SEL= [1,3,5,7] for 100MHz Bandwidth select then NCO1.freq = [ddc_nco (1), ddc_nco (3), ddc_nco (5), ddc_nco (7)]

DDC_LINEUP_param. HB_filt1.coeffs	ddc_tb_in.HB1_coeffs	71x1	Decimator HB1 filter coefficient
DDC_LINEUP_param. HB_filt2.coeffs	ddc_tb_in.HB2_coeffs	21x1	Decimator HB2 filter coefficient
DDC_LINEUP_param. chan_filter.coeffs(1,:)	ddc_tb_in.Coef60	311x1	Chan filter coefficients for 60MHz
DDC_LINEUP_param. chan_filter.coeffs(2,:)	ddc_tb_in.Coef100	311x1	Chan filter coefficients for 100MHz
DDC_LINEUP_param. DDC_Ant_Data_in_1	decimator_out(:,1)	Sfix32_27	decimator_out.mat input to ddc_model
DDC_LINEUP_param. DDC_Ant_Data_in_2	decimator_out(:,2)	Sfix32_27	decimator_out.mat input to ddc_model
DDC_LINEUP_param. DDC_Ant_Data_in_3	decimator_out(:,3)	Sfix32_27	decimator_out.mat input to ddc_model
DDC_LINEUP_param. DDC_Ant_Data_in_4	decimator_out(:,4)	Sfix32_27	decimator_out.mat input to ddc_model
DDC_LINEUP_param. BW_Config_CC1	ddc_tb_in.bw_cc_config	14/10	CC1 is configured for 100 MHz/60MHz

DDC_LINEUP_param.DDC_Const_Data_in_1	Const_Data_in_1	Sfix32_27	constant_out.mat input to ddc_model
DDC_LINEUP_param.DDC_Const_Data_in_2	Const_Data_in_2	Sfix32_27	constant_out.mat input to ddc_model
DDC_LINEUP_param.DDC_Const_Data_in_3	Const_Data_in_3	Sfix32_27	constant_out.mat input to ddc_model
DDC_LINEUP_param.DDC_Const_Data_in_4	Const_Data_in_4	Sfix32_27	constant_out.mat input to ddc_model
DDC_LINEUP_param.DDC_Const_Data_Mux_sel	Const_Data_Mux_sel	0/1	Mux selection for boolean 0 – JESD data 1 – Constant data injection at UL

7.2.5.2 Output of DDC Module

chFiltOut

7.2.6 FFT Module

7.2.6.1 Input to FFT module

Top object

fft_tb_in

Data elements in object

fft_tb_in.dout_v

fft_tb_in.valid_out

fft_tb_in.chout_v

fft_tb_in.fftsize_in

fft_tb_in.CPLen_in

fft_tb_in.nprb_v

fft_tb_in.hcshift_v

fft_tb_in.const_gain_v

fft_tb_in.ifftshift_v

fft_tb_in.Tsym

fft_tb_in.DC_SC_EN

fft_tb_in.Tech_CP

fft_tb_in.F1

fft_tb_in.lut

7.2.6.2 Output of FFT module

Final_Dout

7.2.7 Long PRACH

7.2.7.1 Input to Long PRACH

Top Object

Lprach_tb_in

Data elements in object

lprach_tb_in_local.Prach_tb_data

lprach_tb_in_local.Prach_tb_ch

lprach_tb_in_local.Prach_tb_valid

lprach_tb_in_local.SFN

lprach_tb_in_local.subframe

lprach_tb_in_local.cp_len

lprach_tb_in_local.Timeoffset_input

lprach_tb_in_local.c_m_plane_sel

lprach_tb_in_local.prach_tech

lprach_tb_in_local.gain

lprach_tb_in_local.prach.StartSym

lprach_tb_in_local.prach.FreqOffset

lprach_tb_in_local.prach.ConfigIndex

7.2.7.2 Output of Long PRACH

lprach_tb_exp_out_local.Prach_tb_data

lprach_tb_exp_out_local.Prach_tb_valid

lprach_tb_exp_out_local.prach.ConfigIndex

lprach_tb_exp_out_local.prach.SeqIdx=prach.SeqIdx;

7.3 Filter Co-efficients address and data format

The filter coefficients are programmed as follows:

O_RU_duc_configure.m:channel_coeff_duc(1:65)=channel_coeff_duc_60(1:65);
O_RU_duc_configure.m:channel_coeff_duc(66:130)=channel_coeff_duc_100(1:65);
O_RU_ddc_configure.m:channel_coeff_ddc(1:156)=channel_coeff_ddc_60(1:156);
O_RU_ddc_configure.m:channel_coeff_ddc(157:312)=channel_coeff_ddc_100(1:156);

Data format: [16S 15], Coefficient RAM is 16 Bit width, Byte2 and Byte3 are unused.

Address:
DUC Real Coeff address range: Base address + Byte Offset (0 to 519) d (Coeff loaded for both 60MHz and 100MHz)
DUC Imag Coeff address range: Base address +Byte Offset (1600 to 2119) d (Coeff loaded for both 60MHz and 100MHz)

DDC Real Coeff address range: Base address + Byte Offset (0 to 1247) d (Coeff loaded for both 60MHz and 100MHz)
DDC Imag Coeff address range: Base address +Byte Offset (1600 to 2847) d (Coeff loaded for both 60MHz and 100MHz)

Figure 7-1. Co-efficients

7.4 MATLAB Model IQ sample format at various stages

This section defines the MATLAB model expected vector's output format. This will be used by formatter scripts to convert these IQ into DSPBA compatible format.

7.4.1 IFFT Input

Table 7-15. IFFT Input Format

OFDM symbol	AXC 1	AXC 2	AXC 3	AXC 4	AXC 5	AXC 6	AXC 7	AXC 8
1	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
280	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples

7.4.2 DUC Input

Table 7-16. DUC Input Format

OFDM symbol	AXC 1	AXC 2	AXC 3	AXC 4	AXC 5	AXC 6	AXC 7	AXC 8
1	4096 +352	4096 +352	4096 +352	4096 +352	4096 +352	4096 +352	4096 +352	4096 +352
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
280	4096 +288	4096 +288	4096 +288	4096 +288	4096 +288	4096 +288	4096 +288	4096 +288

7.4.3 Carrier Aggregator and Interpolator Input

Table 7-17. CA-Interp Input Format

Samples	AXC 1	AXC 2	AXC 3	AXC 4	AXC 5	AXC 6	AXC 7	AXC 8
1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1
2	AXC 1 Sample2	AXC 2 Sample2	AXC 3 Sample2	AXC 4 Sample2	AXC 5 Sample2	AXC 6 Sample2	AXC 7 Sample2	AXC 8 Sample2
3	AXC 1 Sample3	AXC 2 Sample3	AXC 3 Sample3	AXC 4 Sample3	AXC 5 Sample3	AXC 6 Sample3	AXC 7 Sample3	AXC 8 Sample3
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
4916600	AXC 1 Sample 4916600	AXC 2 Sample 4916600	AXC 3 Sample 4916600	AXC 4 Sample 4916600	AXC 5 Sample 4916600	AXC 6 Sample 4916600	AXC 7 Sample 4916600	AXC 8 Sample 4916600

7.4.4 Decimator and Delay Compensation Input

Table 7-18. dec_dly_comp Input Format

Samples	AXC 1	AXC 3	AXC 5	AXC 7
1	Sample 1	Sample 1	Sample 1	Sample 1
2	AXC 2 Sample 1	AXC 4 Sample 1	AXC 6 Sample 1	AXC 8 Sample 1
3	AXC 1 Sample 2	AXC 3 Sample 2	AXC 5 Sample 2	AXC 7 Sample 2
4	AXC 2 Sample 2	AXC 4 Sample 2	AXC 6 Sample 2	AXC 8 Sample 2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

9833279	AXC 1 Sample 4916640	AXC 3 Sample 4916640	AXC 5 Sample 4916640	AXC 7 Sample 4916640
9833280	AXC 2 Sample 4916640	AXC 4 Sample 4916640	AXC 6 Sample 4916640	AXC 8 Sample 4916640

7.4.5 DDC Input

Table 7-19. DDC Input Format

Samples	AXC 1	AXC 3	AXC 5	AXC 7
1	Sample 1	Sample 1	Sample 1	Sample 1
2	AXC 2 Sample 1	AXC 4 Sample 1	AXC 6 Sample 1	AXC 8 Sample 1
3	AXC 1 Sample 2	AXC 3 Sample 2	AXC 5 Sample 2	AXC 7 Sample 2
4	AXC 2 Sample 2	AXC 4 Sample 2	AXC 6 Sample 2	AXC 8 Sample 2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
9833279	AXC 1 Sample 4916640	AXC 3 Sample 4916640	AXC 5 Sample 4916640	AXC 7 Sample 4916640
9833280	AXC 2 Sample 4916640	AXC 4 Sample 4916640	AXC 6 Sample 4916640	AXC 8 Sample 4916640

7.4.6 FFT Input

Table 7-20. FFT Input Format

Sample	AXC 1	AXC 2	AXC 3	AXC 4	AXC 5	AXC 6	AXC 7	AXC 8
1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1	Sample1
2	AXC 1 Sample2	AXC 2 Sample2	AXC 3 Sample2	AXC 4 Sample2	AXC 5 Sample2	AXC 6 Sample2	AXC 7 Sample1	AXC 8 Sample2
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1229520	AXC 1 Sample 1229520	AXC 2 Sample 1229520	AXC 3 Sample 1229520	AXC 4 Sample 1229520	AXC 5 Sample 1229520	AXC 6 Sample 1229520	AXC 7 Sample 1229520	AXC 8 Sample 1229520

7.4.7 FFT Output

Table 7-21. FFT Output Format

OFDM symbol	AXC 1	AXC 2	AXC 3	AXC 4	AXC 5	AXC 6	AXC 7	AXC 8
1	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
280	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples	3276 samples

8.0 PCAP to .mat Conversion

8.1 Scope

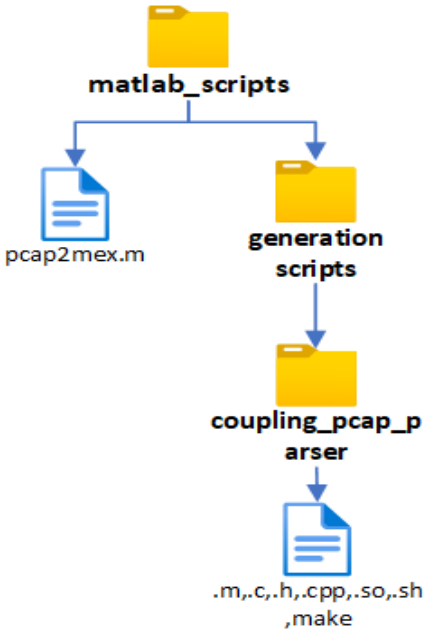
To Generate Hardware Expected using MEX file which stands for "MATLAB executable". It provides an interface between MATLAB and functions written in C, C++. When compiled, MEX files are dynamically loaded, allowing external functions to be invoked from within MATLAB as if they were built-in functions.

8.2 Required Tools

Matlab 2023b

8.3 Folder Structure

Figure 8-1. Folder Structure



8.4 Steps for ifft_in extraction from PCAP

1. Run ./run.sh from terminal in coupling_pcap_parser folder.
   liblibrary.so file required to create mex function will be generated in coupling_pcap_parser_folder if the above command is executed.
2. Run the pcap2mex.m in matlab.
   parser_test.mexa64 will be generated under matlab scripts folder.
3. Refer for the output which is ifft_in.mat file in the path ../../testvectors/input/sim/tcs_01/ .

Consider the following points

- pcap2mex.m generates ifft_in.mat only for tcs_01_100_9000_tc1_bfp9_fs_12.pcap
- **[INPUT_SIM_PATH]** = **../../**testvector /input/sim/ **[pcap_folder_name].**
- For the remaining all testcases ifft_in.mat files will be available in **[INPUT_SIM_PATH].**

8.5 MEX (MATLAB EXECUTABLE)

MEX filenames compiles and links one or more C++ source files written with the MATLAB Data API for C++ into a binary MEX file in the current folder.

If writing MEX files based on the C Matrix API, then MEX filenames build one or more C, C++ source files with the -R2018a API. MathWorks recommends that you create MEX files and update existing MEX files to use the interleaved complex API. Alternatively, use the MX_HAS_INTERLEAVED_COMPLEX macro to apply the desired behaviour across versions of MATLAB.

8.6 MEX Command

Step 1: First integrate the pcap by Mex command.

Mex command: - mex -R2018a test.cpp C-model / liblibrary.so -lpcap

-**R2018a:** Specifies the MATLAB release version (R2018a in this case) for compatibility. This ensures that the MEX file is compiled using the specified MATLAB version's libraries and includes.

parser_test.cpp: This is the source file (parser_test.cpp) that want to compile into a MEX file. MEX files are typically C++ source files that contain functions callable from MATLAB.

C-model /liblibrary.so: This specifies a pre-compiled shared library (liblibrary.so) located in the directory C-model. This library can be linked with your MEX file during compilation.

-lpcap: This option specifies that the MEX file should be linked with the libpcap library (-l indicates a library) during compilation. libpcap is a library used for packet capture.

Step 2: Then call the parser_test.cpp source files that contain functions.

For example, in parser.m mex command is executed as follow,

mex-R2018a generation_scripts/coupling_pcap_parser/parser_test.cpp generation_scripts/coupling_pcap_parser/liblibrary.so -lpcap

8.7 MEX_Wrapper Functions

8.7.1 Parser Function

Function Syntax: function [ifft_in] = parser(pcap_file)

p_file: input pcap file

ifft_in: parser output data of size [917280 X 8]

1. Execute the Mex command with respected arguments of the function to integrate the pcap Parser.
2. In that mex function collects the input buffer, input1, input2, input3 and input4, output data are passed to the test function to extract the Pcap.
   - **Function Syntax:** test (input_buf, input1, input2, input3, input4 output_f).
     - **input_buf:** A string data is copied from prhs [0] into a C string input_ buf.
     - **input1, input2, input3 & input4:** Input1, Input2, Input3 and Input 4 are copied into prhs[1], prhs[2],prhs[3] and prhs[4]. A pointer is created to the real data in the input matrix.
     - **output:** Output matrix is created.

       plhs [0] = mxCreateDoubleMatrix (6552, 2600, mxREAL)
   - In the test function there is another function called pcap_extraction to extract data from pcap file.
   - After the pcap extraction, the pcap parser integration is done. The extracted data is rearranged and saved.
3. Following are the input and output variables provided to parser_test.cc to extract data from pcap,
   - input_buf: pcap_file (tc_c_u_coupling_case_1_compress.pcap file).
   - input1 : number of axc present in pcap (8/16).
   - input2 :compression_type enabled in pcap('9'-compression , '16'-without compression).
   - input3 :fs_offset enabled in pcap (fs_offset value configured in pcap).
   - input4 : eaxc-id configuration in pcap ('1'-eaxc id enabled in pcap, '0'-eaxc-id not enabled in pcap).
   - Output :ifft_in (extracted data from pcap)

9.0 Document Revision History

Table 9-1. Revision History

Date	Version	Changes
2025-05-22	0.1	Initial Release