

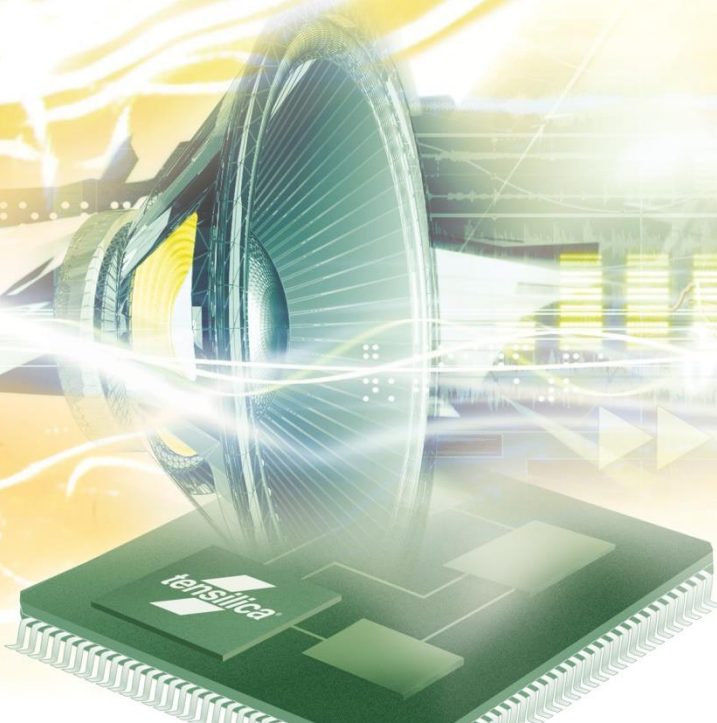


***Simulate dual core system software on
“Linux + XTSC HiFix with XAF”***

User's Guide

Version 0.1
February 2021

For HiFi DSPs



Cadence Design Systems, Inc.
2655 Seely Ave.

DRAFT

Contents

1. Introduction to dual core system software running on “Linux + XTSC HiFix with XAF”	6
1.1. Document Overview	6
1.2. Terminology	6
2. Dual core system software building blocks	7
2.1. About Host Application	7
2.2. Xtensa Remote Processing (XRP)	7
2.3. DSP software running on XTSC HiFix	8
2.4. Runtime interaction between Host and DSP	8
3. Run and Debug from Command Line Environment / LINUX	9
4. Usage of XAF on DSP (HiFi 5, HiFi 4, HiFi 3, etc.)	10
5. Running Host application	10
5.1. Single run mode	10
5.1.1. Single run mode with gain application	10
5.1.2. Single run mode with OGG-Vorbis file playback	11
5.2. Interactive mode	12
5.2.1. Steps to run Interactive mode	12
5.2.2. Interactive mode – user commands	14

Figures

Figure 1- System Architecture	7
Figure 2- Runtime Interaction	8
Figure 3- Single run mode with gain Application	11
Figure 4- Single run mode with OGG-Vorbis file playback	12
Figure 5- Running Host Application	13
Figure 6- Running DSP binary	13
Figure 7- User commands in Interactive mode	14

Tables

No table of figures entries found.

DRAFT

Document Change History

Version	Changes
0.1	Initial draft version

DRAFT

1. Introduction to dual core system software running on “Linux + XTSC HiFix with XAF”

1.1. Document Overview

This software package is designed to accelerate the integration of audio applications made by Xtensa audio framework (XAF) into a dual core heterogeneous system, where one core is ARM/X86 running with Linux operating system to offload the tasks to HiFix core running with XOS operating system using XRP IPC.

Host Processor: X86 running with Linux OS and XRP as IPC.

DSP: XTSC HiFi4 running with XOS, XAF (Xtensa Audio Framework) and XRP as IPC.

Audio application running on Host reads file I/O and/or buffer, then construct XRPM message and sends to HiFix DSP using Xtensa Remote Processing (XRP) to process data, then DSP sends back the processed data and/or acknowledge to Host.

1.2. Terminology

The following terms are used within this guide.

AP: Application Processor

DSP: Digital Signal Processor

IPC: Inter Processor Communication

OS: Operating System

XAF: Xtensa Audio Framework

XRP: Xtensa Remote Processing

XRPM: Xtensa Remote Processing Message

XTSC: Xtensa SystemC

XOS: Xtensa Operating System

2. Dual core system software building blocks

The following figure shows the various building blocks for dual core system software stack on “Linux + XTSC HiFix with XAF”.

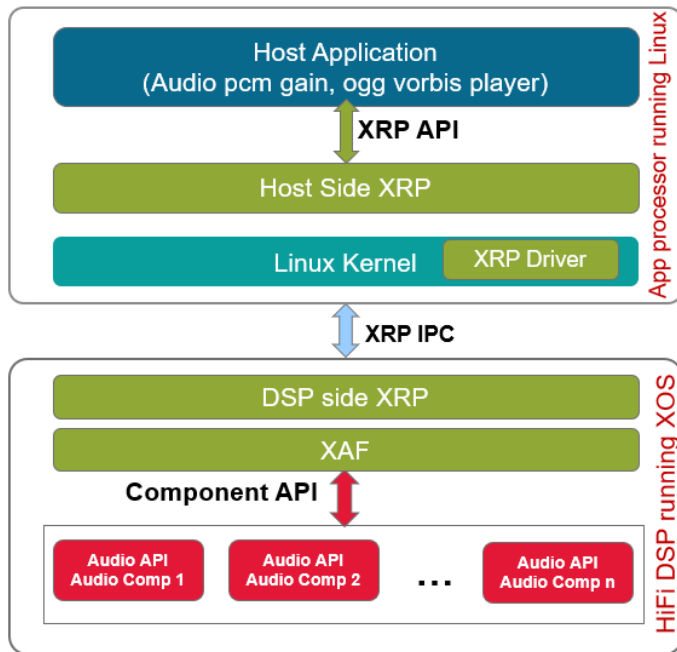


Figure 1- System Architecture

2.1. About Host Application

Host application comes with two threads each one specified with specific task.

shell_thread: Responsible for listening user input and parse user input for valid string. User can type "help" to see all the options supported. It checks for a valid command and passes the command to **host_dsp_thread** for further processing.

host_dsp_thread: Responsible for sending commands/data synchronously from Host to DSP and receiving acknowledgement and processed data from DSP.

2.2. Xtensa Remote Processing (XRP)

Xtensa Remote Processing (XRP) is a communication interface for Linux-based systems containing Xtensa processors. It allows Linux user space tasks to send messages to the firmware running on Xtensa processors. Its implementations support master-slave communication model with a single host node sending messages and one or more DSP nodes executing actions requested in received messages and returning the results. Message structure details are not defined by the XRP, but it is assumed that the message is a small structure that describes requested action accompanied by a vector of buffers with data or storage for that action.

Refer [here](#) for XRP user guide for more details.

2.3. DSP software running on XTSC HiFix

XRP running on HiFix DSP, polls for command and data from Host side. Once it gets XRP message, it passes the XRP message to XAF framework which in turn calls format specific CODEC to process it.

2.4. Runtime interaction between Host and DSP

At runtime, when AP and DSP are ready, the interaction is mainly about the data and control signals. The interaction can be synchronous and asynchronous. This software package currently supports synchronous IPC.

Cadence/Tensilica XRP IPC software is designed to handle the various data and control signal interactions between AP and DSP.

This software package uses XRP standalone implementation can run on a Linux-based system that communicates with Xtensa DSP code running in a simulator on the same Linux-based system.

It consists of the following two parts: the standalone Host implementation of the XRP interface, and the DSP implementation of the XRP interface.

- The standalone host XRP implementation is a library code that runs in a process that directly communicates through the shared memory with the Xtensa DSP side XRP code.
- The DSP implementation of the XRP interface is the same for both hosted and standalone host. The only difference is in the way the DSP firmware gets the shared memory address.

Figure - 1 and Figure - 2 shows the high-level interaction between AP and DSP from software perspective. Figure - 2 indicates a simple scenario, where AP sends command/data to DSP for processing. DSP reads data from IPC, process it and then sends back to AP using XRP IPC.

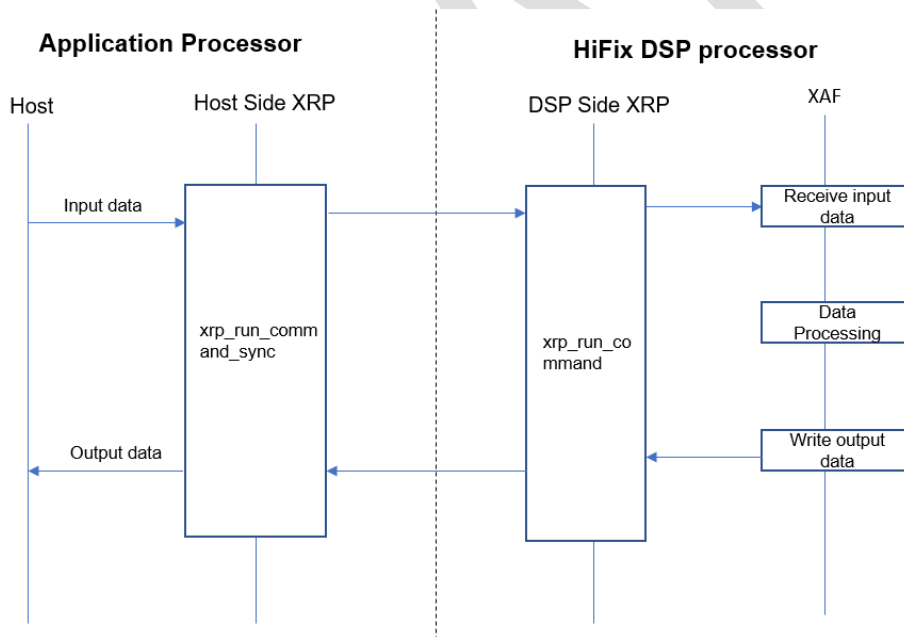


Figure 2- Runtime Interaction

3. Run and Debug from Command Line Environment / LINUX

Launch a terminal and change directory to the `xa_linux_xaf`: In order to use the command line Xtensa tools, some environment variables need to be setup that are used by the build scripts.

```
# Add Xtensa tools binaries to PATH and Setup Xtensa env vars needed for compile and linking
export PATH=$PATH:~/xtensa/XtDevTools/install/tools/RI-2019.2- linux/XtensaTools/bin
export XTENSA_SYSTEM=~/xtensa/XtDevTools/install/tools/RI-2019.2-linux/XtensaTools/config
export XTENSA_CORE= AE_HiFi4_LE5
export PATH=~/xtensa/XtDevTools/install/tools/RI-2019.2- linux/XtensaTools/Tools/bin:$PATH
```

```
# only set this below, if /usr/bin/gcc version is gretaer than ~/XtensaTools/Tools/bin/gcc
export PATH=/usr/bin:$PATH
```

```
# Install libfdt-dev on your local machine only if /usr/bin/gcc is used.
user@linux~ xa_linux_xaf$ sudo apt-get install libfdt-dev
```

```
#Build and run pcm gain audio application on the Dual core system (Linux + XTSC HiFix)
user@linux~ xa_linux_xaf$ cd build
user@linux~ xa_linux_xaf/build$ make clean
```

```
# Copies over the XRP host source files from XtensaTools
user@linux~ xa_linux_xaf/build$ make xrp_setup
```

```
#Generates the XTSC model and the MP LSPs
user@linux~ xa_linux_xaf/build$ make build
```

```
# Generates the XRP libraries for the host
user@linux~ xa_linux_xaf/build$ make stmp-xrp
```

```
#Build XAF library with pcm gain/ogg-vorbis components.
user@linux~ xa_linux_xaf/build$ cd ../dsp/xaf/build/
user@linux~ xa_linux_xaf/dsp/xaf/build$ make clean all
user@linux~ xa_linux_xaf/dsp/xaf/build $ cd ../../../build/
```

```
#Builds the binaries for the host and the single core DSP and runs them
user@linux~ xa_linux_xaf/build$ make run
```

Once this process has finished, you should see a series of files get compiled, followed by some logging output from a test, which should conclude with **PASSED - Output matches with reference**. If you see this, it means that a pcm gain audio application has been built and run buffer I/O on Host Linux and processed gain on HiFix DSP.

The following log shows gain application ran on Host and DSP.

```
user@linux~ xa_linux_xaf/build$ make run
Cadence Xtensa Audio Framework
Library Name   : Audio Framework (Hostless)
Library Version : 2.3
API Version    : 1.3
```

```
[DSP Gain] Audio Device Ready
[DSP Gain] Gainer component started
[DSP_ProcessThread] start
```

[DSP_ProcessThread] input over
[DSP_ProcessThread] Execution complete - exiting
[DSP_ProcessThread] exiting
[DSP Gain] Audio device closed

PASSED - Output matches with reference

Note: This software package is prepared only to run for **AE_HiFi4_LE5** core configuration. To work for other cores, modify `subsys.xld` and `subsys.yml` files located at `user@linux~ xa_linux_xaf/dsp/xaf/build$`

4. Usage of XAF on DSP (HiFi 5, HiFi 4, HiFi 3, etc.)

Xtensa Audio Framework (XAF) is a framework designed to accelerate the development of audio processing applications for the HiFi family of DSP cores. Application developers may choose components from the rich portfolio of audio and speech libraries already developed by Cadence and its ecosystem partners. In addition, customers may also package their proprietary algorithms and components and integrate them into the framework. Towards this goal, a simplified “Developer API” is defined, which enables application developers to rapidly create an end application and focus more on using the available components. XAF is designed to work on both the instruction set simulator as well as actual hardware.

Refer [here](#) for XAF user guide for more details.

5. Running Host application

Application on Host side can run in two modes. These two modes are Single run mode and Interactive mode.

5.1. Single run mode

In Single run mode application starts running with command line arguments pre provided by the user in `run_sim.sh` script file.

5.1.1. Single run mode with gain application

Refer [Section-3](#) on how to use “make run” command. By default, “make run” runs in Single run mode with gain application.

```
[
    build]$ make run
rm -rf run_sim.sh; cat run_prologue.sh > run_sim.sh; echo "./host_application_test gain & pids[0]=$!" >> run_sim.sh; echo
"sleep 1" >> run_sim.sh
chmod 755 run_sim.sh
./run_sim.sh
+ trap on_exit EXIT
+ trap on_exit INT
+ pids[0]=305477
+ sleep 1
+ ./host_application_test gain

=====
Linux + Xtensa HiFix with XAF version      : 0.1
Host process id                          : 305477

=====
+ xtsc-run --set_xtsc_parm=turbo=true --set_xtsc_parm=turbo_max_relaxed_cycles=10000000 --define=DSP0_BINARY=xa_dsp_firmware
--define=SHARED_RAM_NAME=SharedRAM_L_305477 --define=SYSTEMRAM_DELAY=1 --define=SYSTEMROM_DELAY=1 --include=sysbuild_dir/xts
c-run/SubSystem.inc

SystemC 2.3.1-Accellera --- Aug 27 2019 15:08:06
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED

Cadence Xtensa Audio Framework
Library Name : Audio Framework (Hostless)
Library Version : 2.3
API Version : 1.3

[DSP Gain] Audio Device Ready
[DSP Gain] Gainer component started
[DSP_ProcessThread] start
[DSP_ProcessThread] input over
[DSP_ProcessThread] Execution complete - exiting
[DSP_ProcessThread] exiting
[DSP Gain] Audio device closed
```

Figure 3- Single run mode with gain Application

5.1.2. Single run mode with OGG-Vorbis file playback

In order to use Single run mode with OGG-Vorbis file playback, replace 'gain' with 'file ../test/test_inp/hihat.ogg' in run_sim.sh as shown below.

Example: Launch a terminal and change directory to the xa_linux_xaf and then open run_sim.sh file from build folder.

```
user@linux:~/xa_linux_xaf$ vim ./build/run_sim.sh
```

```
./host_application_test gain & pids[0]=$!
```

Modify the above line from run_sim.sh file with the below line.

```
./host_application_test file ../test/test_inp/hihat.ogg & pids[0]=$!
```

```
[build]$ ./run_sim.sh
+ trap on_exit EXIT
+ trap on_exit INT
+ pids[0]=365496
+ sleep 1
+ ./host_application_test file ../test/test_inp/hihat.ogg

=====

Linux + XTSC HiFix with XAF version      : 0.1
Host process id                        : 365496

=====

File format: VORBIS
>>> xtsc-run --set_xtsc_parm=turbo=true --set_xtsc_parm=turbo_max_relaxed_cycles=10000000 --define=DSP0_BINARY=xa_dsp_firmwa
re --define=SHARED_RAM_NAME=SharedRAM_L.365496 --define=SYSTEMRAM_DELAY=1 --define=SYSTEMROM_DELAY=1 --include=sysbuild_dir/x
tsc-run/SubSystem.inc

SystemC 2.3.1-Accellera --- Aug 27 2019 15:08:06
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED

Cadence Xtensa Audio Framework
Library Name      : Audio Framework (Hostless)
Library Version   : 2.3
API Version       : 1.3

[DSP Codec] Audio Device Ready
[DSP Codec] Decoder created
[DSP Codec] Decoder component started
[DSP Codec] Setting decode playback format:
Decoder          : vorbis_dec
Sample rate      : 44100
Bit Width        : 16
Channels         : 2
[DSP Codec] Renderer component created
[DSP_BufferThread] start
```

Figure 4- Single run mode with OGG-Vorbis file playback

5.2. Interactive mode

Host Application and DSP binary are invoked in two different terminals. User can provide the inputs to run various audio applications interactively.

5.2.1. Steps to run Interactive mode

1. Open the terminal and set the Xtensa tools as described below and change directory to `xa_linux_xaf/build`

```
# Add Xtensa tools binaries to PATH and Setup Xtensa env vars needed for compile and linking
export PATH=$PATH:~/xtensa/XtDevTools/install/tools/RI-2019.2- linux/XtensaTools/bin
export XTENSA_SYSTEM=~/xtensa/XtDevTools/install/tools/RI-2019.2-linux/XtensaTools/config
export XTENSA_CORE= AE_HiFi4_LE5
export PATH=~/xtensa/XtDevTools/install/tools/RI-2019.2- linux/XtensaTools/Tools/bin:$PATH
```

```
#Change directory to build
user@linux~ xa_linux_xaf$ cd build
```

2. Complete [Section-3](#) steps and ensure that `host_application_test` and `xa_dsp_firmware` binaries are generated in the **build** folder (user@linux:~/xa_linux_xaf/build\$)

3. Run `host_application_test` from the terminal as shown in the below *Figure - 5*

#Run host applicaiton_test

`user@linux~ xa_linux_xaf/build$./host_application_test`

```
[build]$ ./host_application_test

=====
Linux + XTSC HiFix with XAF version      : 0.1
Host process id                          : 145725
=====

User commands:

help      : List of all commands,
exit      : Exit both host and dsp programs,
version   : Query DSP for XAF/Vorbis Library version,
gain      : Run PCM gain on DSP with I/O buffers from host,
file      : Perform audio file decode and playback on DSP,
           USAGE :file [list|stop]<audio_file>],
               file list <path>      : List audio files from the specified directory
                                   : By default if no path is provided, it list files from ../test/test_inp/ directory
               file <audio_file>    : Decode the audio_file and write into file output
               file stop             : Stop file decoding
>>
```

Figure 5- Running Host Application

4. Open another terminal and complete the step-1 as mentioned above (to set xtensa tool and change directory to build folder) and run DSP binary using `xtsc-run`. Get host process id of host application run (shown in the above *figure - 5*) and pass this id to the command line argument of DSP binary as shown in the below *figure - 6*.

#Execute the DSP binary

`user@linux~ xa_linux_xaf/build$ xtsc-run --set_xtsc_parm=turbo=true --set_xtsc_parm=turbo_max_relaxed_cycles=10000000 --define=DSP0_BINARY=xa_dsp_firmware --define=SHAREDGRAM_NAME=SharedGRAM_L.145725 --define=SYSTEMRAM_DELAY=1 --define=SYSTEMROM_DELAY=1 --include=sysbuild_dir/xtsc-run/SubSystem.inc`

```
[build]$ xtsc-run --set_xtsc_parm=turbo=true --set_xtsc_parm=turbo_max_relaxed_cycles=10000000 --define=DSP0_BINARY=xa_dsp_firmware --define=SHAREDGRAM_NAME=SharedGRAM_L.145725 --define=SYSTEMRAM_DELAY=1 --define=SYSTEMROM_DELAY=1 --include=sysbuild_dir/xtsc-run/SubSystem.inc

SystemC 2.3.1-Accellera --- Aug 27 2019 15:08:06
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED

Cadence Xtensa Audio Framework
Library Name   : Audio Framework (Hostless)
Library Version : 2.3
API Version    : 1.3
```

Figure 6- Running DSP binary

5.2.2. Interactive mode – user commands

Once application is started, observe terminal output with shell prompt. User commands are listed in the shell. See the below *figure - 7*.

```
>>version
Linux + XTSC HiFix with XAF version : 0.1
Component versions from DSP:
Audio Framework version 2.3
Audio Framework API version 1.3
VORBIS Decoder Lib version 1.12

>>gain

PASSED - Output matches with reference

>>file list ./
Available audio files:
File 1:: chirp_1ch_16b_192kbps.mp3
File 2:: hihat.ogg

>>file hihat.ogg
File format: VORBIS
>>
DSP file playback complete

>>file chirp_1ch_16b_192kbps.mp3
File format: MP3
>>file stop
STOP Command success!!!!

DSP file playback complete

>>exit
HOST :: EXIT COMMAND ACK FROM DSP:
```

Figure 7- User commands in Interactive mode