Transceiver Facility FPGA PSM Specification

Transceiver Facility
where software defines the radio

Document WINNF-TS-0008-App03

Version V2.1.1

22 January 2022

TERMS, CONDITIONS & NOTICES

This document has been prepared by the work group of WInnF project SCA-2015-001 “Transceiver Next” to assist The Software Defined Radio Forum Inc. (or its successors or assigns, hereafter “the Forum”). It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the Transceiver Next work group.

Contributors to this document that have submitted copyrighted materials (the Submission) to the Forum for use in this document retain copyright ownership of their original work, while at the same time granting the Forum a non-exclusive, irrevocable, worldwide, perpetual, royalty-free license under the Submitter’s copyrights in the Submission to reproduce, distribute, publish, display, perform, and create derivative works of the Submission based on that original work for the purpose of developing this document under the Forum's own copyright.

Permission is granted to the Forum’s participants to copy any portion of this document for legitimate purposes of the Forum. Copying for monetary gain or for other non-Forum related purposes is prohibited.

THIS DOCUMENT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS DOCUMENT.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

This document was developed following the Forum's policy on restricted or controlled information (Policy 009) to ensure that that the document can be shared openly with other member organizations around the world. Additional Information on this policy can be found here: http://www.wirelessinnovation.org/page/Policies_and_Procedures

Although this document contains no restricted or controlled information, the specific implementation of concepts contain herein may be controlled under the laws of the country of origin for that implementation. Readers are encouraged, therefore, to consult with a cognizant authority prior to any further development.

Wireless Innovation Forum ™ and SDR Forum ™ are trademarks of the Software Defined Radio Forum Inc.

Table of Contents

List of Figures

List of Tables

Contributors

The following individuals and their organization of affiliation are credited as Contributors to development of the specification, for having been involved in the work group that developed the draft then approved by WInnF member organizations:

- Eric Nicollet, Thales,
- Claude Bélisle, VIAVI Solutions,
- David Hagood, Cynosure,
- Steve Bernier, VIAVI Solutions,
- Marc Adrat, Fraunhofer FKIE,
- Guillaume Delbarre, DGA,
- Olivier Kirsch, KEREVAL,
- Charles Linn, L3Harris,
- David Murotake, HiKE,
- Kevin Richardson, MITRE.
- Frédéric Le Roy, ENSTA,
- Dmitri Zvernick, NordiaSoft.

Transceiver Facility FPGA PSM Specification

1 Introduction

This document WINNF-TS-0008-App03 is the *FPGA PSM specification* of *Transceiver Facility* V2.1.0.

It derives from *Transceiver Facility PIM Specification* [Ref1] in accordance with *Principles for WInnForum Facility Standards* [Ref2].

It addresses the FPGA programming paradigm, applying the mapping rules of the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3] and specifically reporting any deviation to those rules.

The following figure positions the interfaces addressed by the *FPGA PSM specification*:

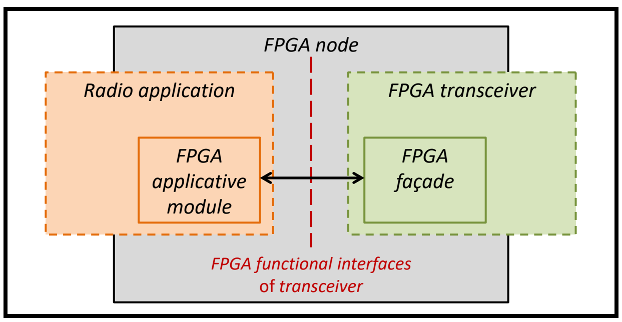

Figure 1 -Positioning of FPGA PSM interfaces

As depicted, the *FPGA PSM specification* addresses the *FPGA functional interfaces* of *transceivers*, positioned, within an *FPGA node*, between the *FPGA applicative module*s of *radio applications* and *FPGA façades* of *transceiver* instances.

The *FPGA PSM specification* considers RTL (Register-Transfer Level) [Ref4] interfaces, specifying RTL signals and the associated chronogram, independently from the used programming language (e.g., VHDL or Verilog).

It also provides normative source files applicable in case of VHDL programming.

1.1 Reference definitions

The *Transceiver Facility FPGA PSM specification* applies the following definitions from *Transceiver Facility PIM Specification* [Ref1]:

Topic	Used definitions
Transceiver concepts	*transceiver*, *Transceiver Facility*, *Tx channel, Rx channel*

Table 1 Definitions from *Transceiver Facility PIM Specification*

The *Transceiver Facility FPGA PSM specification* applies the following definitions from *Principles for WInnForum Facility Standards* [Ref2]:

Topic	Used definitions
Base concepts	*radio application*
Architecture concepts	*Façade*
WInnForum facilities	*facility*, *PIM specification*, *PSM specification*
Services	*Service*
Primitives	*primitive*, *parameter*, *exception*, *type*

Table 2 Definitions from *Principles for WInnForum Facility Standards*

The *Transceiver Facility FPGA PSM specification* applies the following definitions from *WInnForum Facilities PSMs Mapping Rules* [Ref3]:

Topic	Used definitions
Interfaces	*FPGA PSM specification*, *FPGA functional interfaces*
Software architecture	*FPGA node*, *FPGA façade*, *FPGA applicative module*
RTL signals origin	*origin, caller, callee*
Base RTL signals	*primitive prefix, structural RTL signals*, *semantics RTL signals*, *parameters RTL signals*, *exception RTL signals*

Table 3 Definitions from *WInnForum Facilities PSMs Mapping Rules*

The term "*unspecified*" indicates an aspect explicitly left to implementer's decisions.

1.2 Conformance

1.2.1 Radio platform items

An *FPGA façade* of a *transceiver* implementation **is conformant with** the *Transceiver Facility FPGA PSM specification* if it provides an FPGA implementation of related *primitives*.

An *FPGA transceiver* **is defined as** a *transceiver* implementation with all of its *FPGA façades* being conformant with the *FPGA PSM specification*.

1.2.2 Radio application items

An *FPGA applicative module* of a *radio application* **is conformant with** the *Transceiver Facility FPGA PSM specification* if it can use *FPGA façades* conformant with the *FPGA PSM specification* without using any non-standard *primitive* for the *transceiver*.

1.3 Document structure

Section 2 specifies the normative content for the *FPGA functional interfaces*.

Section 3 specifies the normative content for FPGA constants.

Section 4 specifies the normative files to be used for VHDL programming.

2 FPGA functional interfaces

This normative section specifies the *FPGA functional interfaces* for *transceiver*, according to the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

2.1 Specification approach

2.1.1 Base RTL signals

The following base RTL signals from the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3] are used:

Base RTL signal	Used definitions
Structural RTL signals	**CLK**, **RST**
Semantics RTL signals	**EN**, **RDY**
Parameters RTL signals	**EN_IN**, **DATA_IN**, **EN_OUT**, **DATA_OUT**
Exception RTL signals	**IRQ**

Table 4 Base RTL signals from *WInnForum Facilities PSMs Mapping Rules*

For each base RTL signal, the complete RTL signal name, its *origin* (using the concept of *caller* and *callee*) and its format are specified.

2.1.2 Primitive prefixes

The *primitive prefixes* for *transceiver* follow the related indications of *WInnForum Facilities PSMs Mapping Rules*, with the *transceiver*-specific addition of **<CHAN>_** field.

A *primitive prefix* concatenates:

- The **XCVR_** field, for *transceiver*,
- The **<instNum>_** field, optionally numbering instances of a *transceiver* in case there are more than one (starting count from **1**),
- The **<CHAN>_** field indicating if an RTL signal is attached to *Tx channels* or *Rx channels* with, for **SamplesTransmission** and **SamplesReception**, the *Tx channel* or *Rx channel* number,
- The **<PRIM_NAME>_** field, identifying the *primitive* using a screaming snake case transcription of the *PIM specification* name.

The value of **<CHAN>** **is specified by** the following table:

<CHAN> **value**	**Case**
TX	For *Services* attached to *Tx channels*. Including **SamplesTransmission** if only one *Tx channel* is used.
RX	The *Services* attached to *Rx channels*. Including **SamplesReception** if only one *Rx channel* is used.
TX<channelNumber>	For **SamplesTransmission** if several *Tx channels* are used, starting at 1.
RX<channelNumber>	For **SamplesReception** if several *Rx channels* are used, starting at 1.

Table 5 Possible values for <CHAN>

2.1.3 Stream-oriented design

A stream-oriented design, keeping the notion of baseband samples block and completely suppressing the notion of baseband samples packet, is introduced for **SamplesReception** (see section 2.2.8) and **SamplesTransmission** (see section 2.2.9) *Services*.

The PIM *primitives* *pushTxPacket()* and *pushRxPacket()* are replaced by the substitute *primitives* *pushTxBlock()* and *pushRxBlock()*, which are then mapped to an FPGA interface:

- One-by-one transfers of baseband samples are realized over the length of the block,
- The main parameter becomes ***in*** *BasebandBlock* ***sequence<BasebandSamples>***,
- Parameter *endOfBlock* is suppressed.

The **RxPacketsLengthControl** service is not mapped, implying:

- Operation *setRxPacketsLength()* is not mapped,
- Type ***PacketsLength*** is not mapped.

2.2 Interfaces specification

2.2.1 Transceiver::Management::Reset

2.2.1.1 reset()

The RTL signals for *reset()* **are specified by** the following table:

RTL signal name **XCVR_{<instNum>_}_<CHAN>_RESET_** +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
EN_OUT	*FPGA façade*	1-bit signal

Table 6 RTL signals for *reset()*

The dynamic behavior for *reset()* **is specified by** the following chronogram:

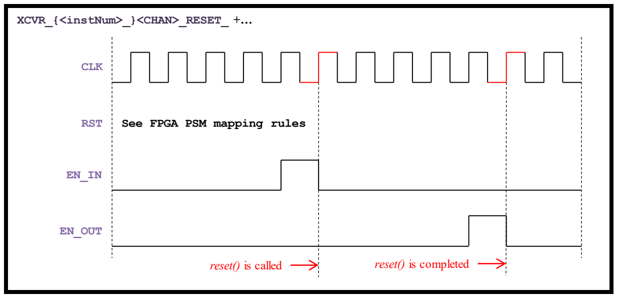

Figure 2 Dynamic behavior for *reset()*

2.2.2 Transceiver::Management::RadioSilence

2.2.2.1 startRadioSilence()

The RTL signals for *startRadioSilence()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_START_RADIO_SILENCE_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 7 RTL signals for *startRadioSilence()*

The dynamic behavior for *startRadioSilence()* **is specified by** the following chronogram:

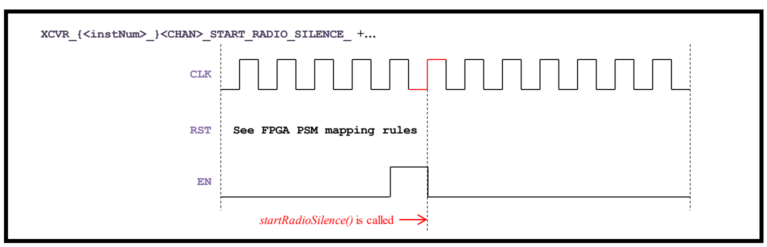

Figure 3 Dynamic behavior for *startRadioSilence()*

2.2.2.2 stopRadioSilence()

The RTL signals for *stopRadioSilence()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_STOP_RADIO_SILENCE_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 8 RTL signals for *stopRadioSilence()*

The dynamic behavior for *stopRadioSilence()* **is specified by** the following chronogram:

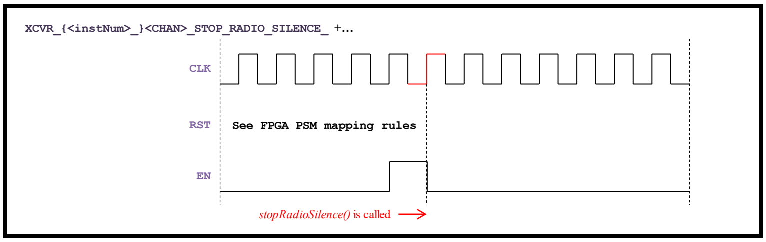

Figure 4 Dynamic behavior for *stopRadioSilence()*

2.2.3 Transceiver::BurstControl::DirectCreation

2.2.3.1 startBurst()

The RTL signals for *startBurst()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_START_BURST_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_length	*FPGA app module*	32-bit vector
START_BURST_RDY	*FPGA façade*	1-bit signal

Table 9 RTL signals for *startBurst()*

The dynamic behavior for *startBurst()* **is specified by** the following chronogram:

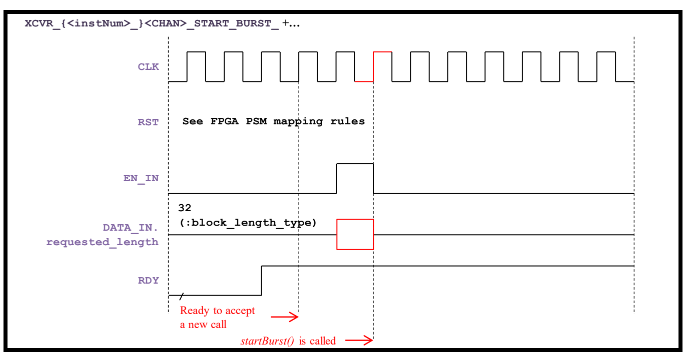

Figure 5 Dynamic behavior for *startBurst()*

2.2.4 Transceiver::BurstControl::RelativeCreation

2.2.4.1 scheduleRelativeBurst()

The RTL signals for *scheduleRelativeBurst()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_SCHEDULE_RELATIVE_BURST_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_alternate	*FPGA app module*	1-bit signal
DATA_IN.requested_delay	*FPGA app module*	32-bit or 64-bit vector
DATA_IN.requested_length	*FPGA app module*	32-bit vector
RDY	*FPGA façade*	1-bit signal

Table 10 RTL signals for *scheduleRelativeBurst()*

The dynamic behavior for *scheduleRelativeBurst()* **is specified by** the following chronogram:

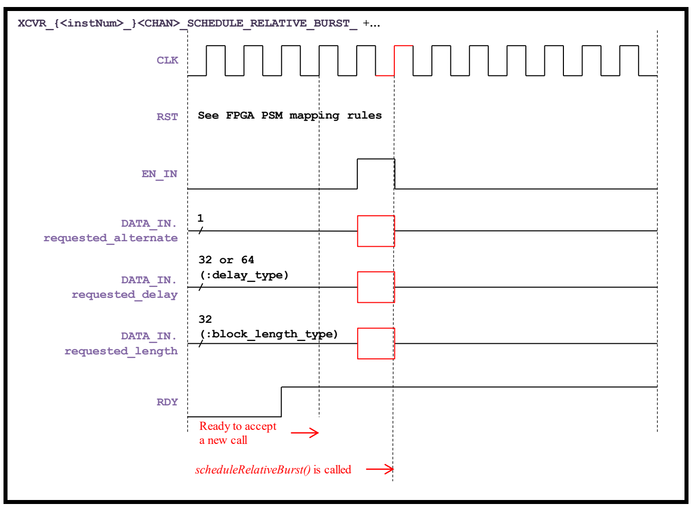

Figure 6 Dynamic behavior for *scheduleRelativeBurst()*

2.2.5 Transceiver::BurstControl::AbsoluteCreation

2.2.5.1 scheduleAbsoluteBurst()

The RTL signals for *scheduleAbsoluteBurst()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_SCHEDULE_ABSOLUTE_BURST_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_start_time	*FPGA app module*	Two 32-bit vectors
DATA_IN.requested_length	*FPGA app module*	32-bit vector
RDY	*FPGA façade*	1-bit signal

Table 11 RTL signals for *scheduleAbsoluteBurst()*

The dynamic behavior for *scheduleAbsoluteBurst()* **is specified by** the following chronogram:

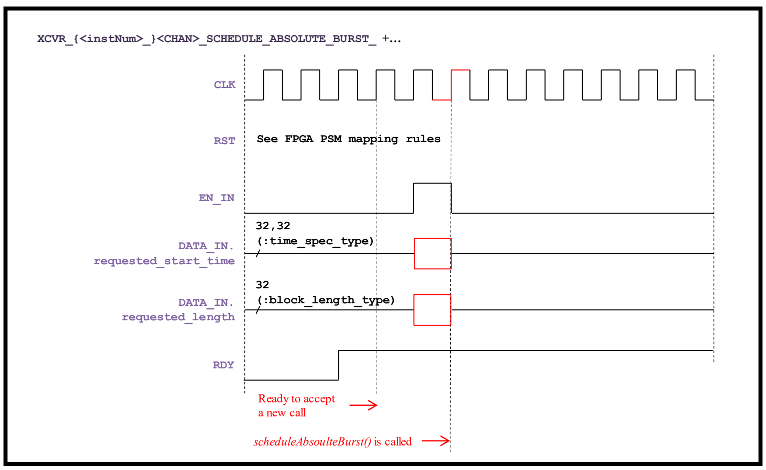

Figure 7 Dynamic behavior for *scheduleAbsoluteBurst()*

2.2.6 Transceiver::BurstControl::StrobedCreation

2.2.6.1 scheduleStrobedBurst()

The RTL signals for *scheduleStrobedBurst()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_SCHEDULE_STROBED_BURST_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_strobe_source	*FPGA app module*	16-bit vector
DATA_IN.requested_delay	*FPGA app module*	32-bit or 64-bit vector
DATA_IN.requested_length	*FPGA app module*	32-bit vector
RDY	*FPGA façade*	1-bit signal

Table 12 RTL signals for *scheduleStrobedBurst()*

The dynamic behavior for *scheduleStrobedBurst()* **is specified by** the following chronogram:

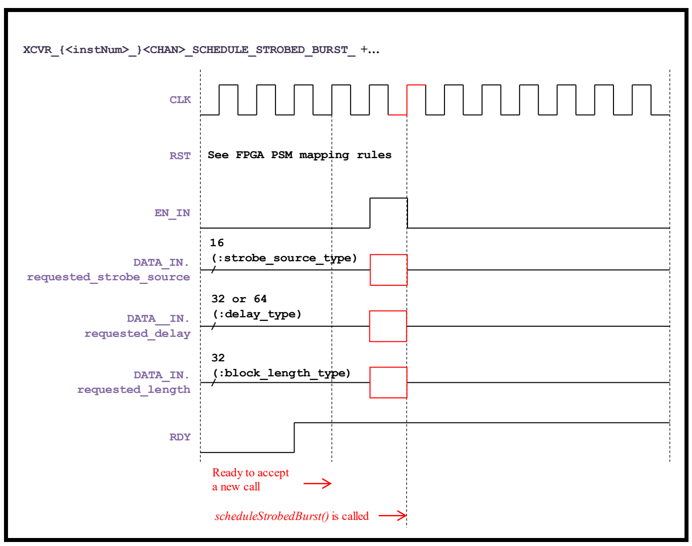

Figure 8 Dynamic behavior for *scheduleStrobedBurst()*

2.2.7 Transceiver::BurstControl::Termination

2.2.7.1 setBlockLength()

The RTL signals for *setBlockLength()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_SET_BLOCK_LENGTH_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_length	*FPGA app module*	32-bit vector

Table 13 RTL signals for *setBlockLength()*

The dynamic behavior for *setBlockLength()* **is specified by** the following chronogram:

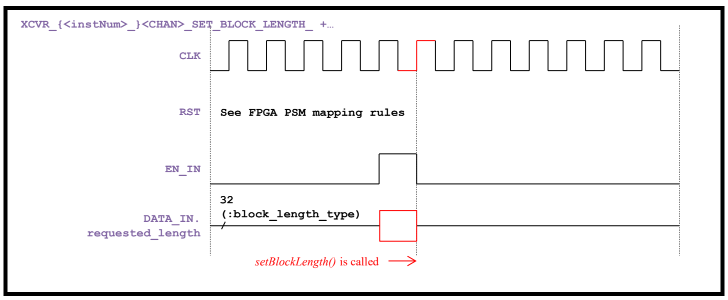

Figure 9 Dynamic behavior for *setBlockLength()*

2.2.7.2 stopBurst()

The RTL signals for *stopBurst()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_STOP_BURST_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 14 RTL signals for *stopBurst()*

The dynamic behavior for *stopBurst()* **is specified by** the following chronogram:

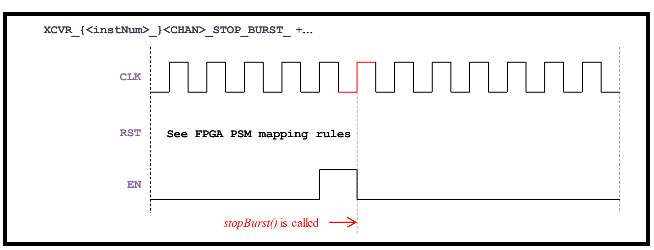

Figure 10 Dynamic behavior for *stopBurst()*

2.2.8 Transceiver::BasebandSignal::SamplesReception

2.2.8.1 pushRxPacket()/pushRxBlock()

The stream-oriented design explained in section 2.1.3 is applied, with the PIM *primitive* *pushRxPacket()* being replaced by *pushRxBlock()*, with a **sequence** of baseband samples as the only **in** *parameter*, no **out** *parameter*.

No flow control mechanism is required for reception.

The transmission of meta-data, as specified by value of **RX_META_DATA**, is directly using transmission of data of type ***rx_meta_data_type***.

The RTL signals for *pushRxBlock()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_PUSH_RX_BLOCK_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
BASEBAND_SAMPLE_FIRST	*FPGA façade*	1-bit signal
BASEBAND_SAMPLE_LAST	*FPGA façade*	1-bit signal
BASEBAND_SAMPLE_EN	*FPGA façade*	1-bit signal
BASEBAND_SAMPLE_DATA	*FPGA façade*	Two 16-bit or 32-bit vectors
EN_IN If RX_META_DATA is equal to TRUE.	*FPGA app module*	1-bit signal
DATA_IN.rx_meta_data If RX_META_DATA is equal to TRUE.	*FPGA app module*	User-defined

Table 15 RTL signals for *pushRxBlock()*

The dynamic behavior for *pushRxBlock()* **is specified by** the following chronogram:

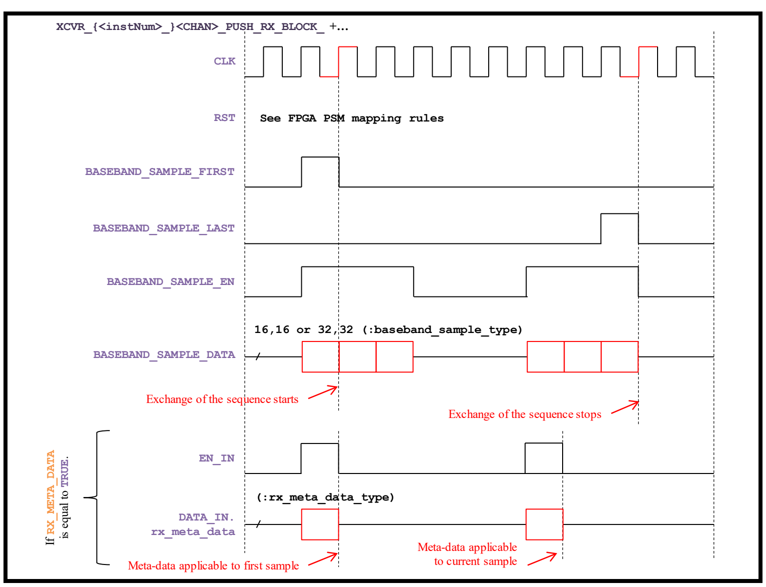

Figure 11 Dynamic behavior for *pushRxBlock()*

2.2.9 Transceiver::BasebandSignal::SamplesTransmission

2.2.9.1 pushTxPacket()/pushTxBlock()

The stream-oriented design explained in section 2.1.3 is applied, with the PIM *primitive* *pushTxPacket()* being replaced by *pushTxBlock()*, with a **sequence** of baseband samples as the only **in** *parameter*, no **out** *parameter*.

Flow control signaling is added to ensure transmission integrity.

The transmission of meta-data, as specified by value of **TX_META_DATA**, is directly using transmission of data of type ***tx_metadata_type***.

The RTL signals for *pushTxBlock()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_PUSH_TX_BLOCK_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
BASEBAND_SAMPLE_FIRST	*FPGA app module*	1-bit signal
BASEBAND_SAMPLE_LAST	*FPGA app module*	1-bit signal
BASEBAND_SAMPLE_EN	*FPGA app module*	1-bit signal
BASEBAND_SAMPLE_DATA	*FPGA app module*	2 16-bit or 32-bit vectors
BASEBAND_SAMPLE_RDY	*FPGA façade*	1-bit signal
EN_IN If TX_META_DATA is equal to TRUE.	*FPGA app module*	1-bit signal
DATA_IN If TX_META_DATA is equal to TRUE.	*FPGA app module*	User-defined

Table 16 RTL signals for *pushTxBlock()*

The dynamic behavior for *pushTxBlock()* **is specified by** the following chronogram:

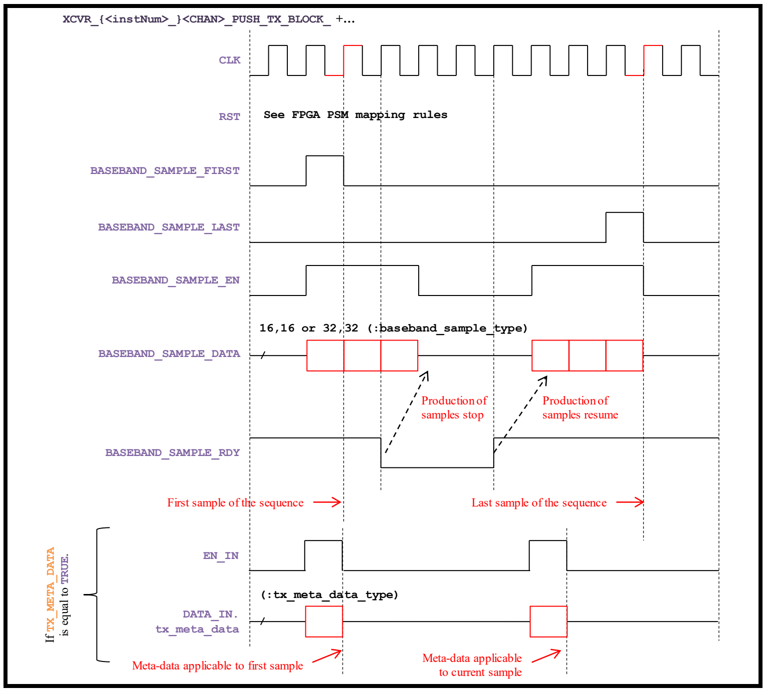

Figure 12 Dynamic behavior for *pushTxBlock()*

2.2.10 Transceiver::BasebandSignal::RxPacketsLengthControl

Not mapped.

2.2.11 Transceiver::Tuning::InitialTuning

2.2.11.1 setTuning()

The RTL signals for *setTuning()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_SET_TUNING_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_preset	*FPGA app module*	16-bit vector
DATA_IN.requested_frequency	*FPGA app module*	32-bit or 64-bit vector
DATA_IN.requested_gain	*FPGA app module*	16-bit vector – signed
DATA_IN.requested_burst_number	*FPGA app module*	16-bit vector
RDY	*FPGA façade*	1-bit signal

Table 17 RTL signals for *setTuning()*

The dynamic behavior for *setTuning()* **is specified by** the following chronogram:

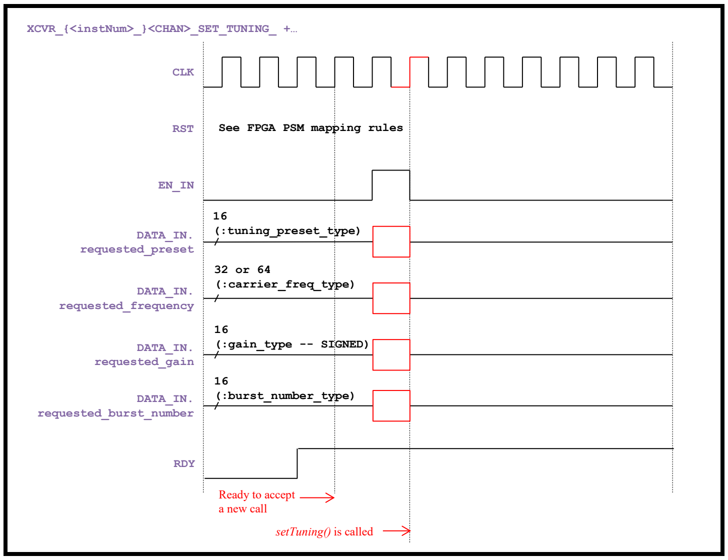

Figure 13 Dynamic behavior for *setTuning()*

2.2.12 Transceiver::Tuning::Retuning

2.2.12.1 retune()

The RTL signals for *retune()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_RETUNE_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA app module*	1-bit signal
DATA_IN.requested_frequency	*FPGA app module*	32-bit or 64-bit vector
DATA_IN.requested_gain	*FPGA app module*	16-bit vector
DATA_IN.requested_delay	*FPGA app module*	32-bit or 64-bit vector

Table 18 RTL signals for *retune()*

The dynamic behavior for *retune()* **is specified by** the following chronogram:

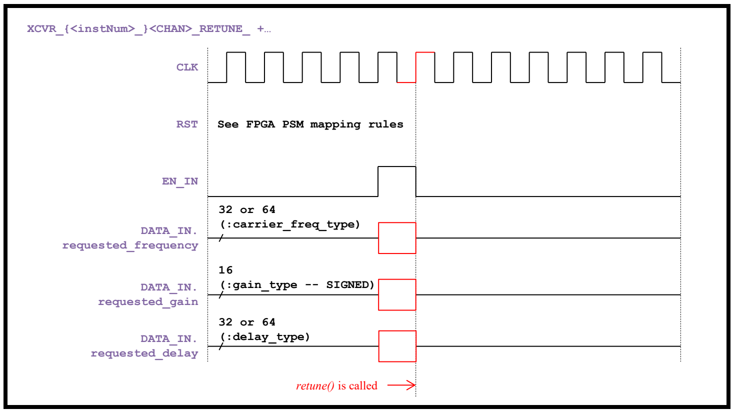

Figure 14 Dynamic behavior for *retune()*

2.2.13 Transceiver::Notifications::Events

2.2.13.1 notifyEvent()

The RTL signals for *notifyEvent()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_NOTIFY_EVENT_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.notified_event	*FPGA façade*	3-bit vector

Table 19 RTL signals for *notifyEvent()*

The dynamic behavior for *notifyEvent()* **is specified by** the following chronogram:

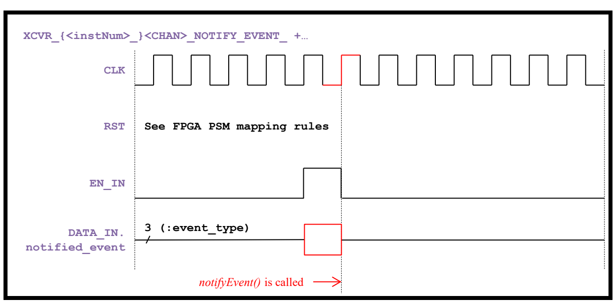

Figure 15 Dynamic behavior for *notifyEvent()*

2.2.14 Transceiver::Notifications::Errors

2.2.14.1 notifyError()

The RTL signals for *notifyError()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_NOTIFY_ERROR_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.notified_error	*FPGA façade*	4-bit vector

Table 20 RTL signals for *notifyError()*

The dynamic behavior for *notifyError()* **is specified by** the following chronogram:

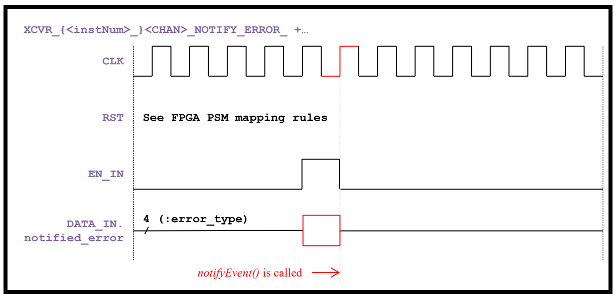

Figure 16 Dynamic behavior for *notifyError()*

2.2.15 Transceiver::GainControl::GainChanges

2.2.15.1 indicateGain()

The RTL signals for *indicateGain()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_INDICATE_GAIN_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.new_gain	*FPGA façade*	16-bit vector -- signed
DATA_IN.first_valid_sample	*FPGA façade*	32-bit vector

Table 21 RTL signals for *indicateGain()*

The dynamic behavior for *indicateGain()* **is specified by** the following chronogram:

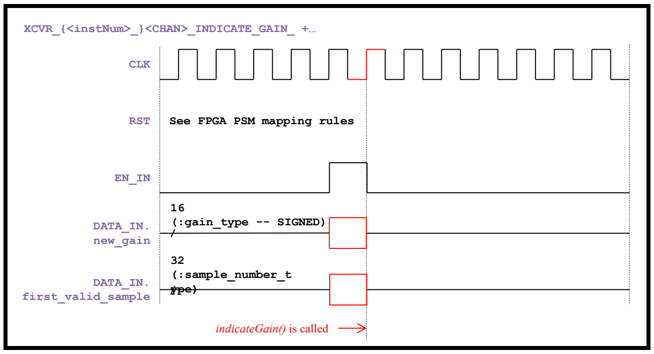

Figure 17 Dynamic behavior for *indicateGain()*

2.2.16 Transceiver::GainControl::GainLocking

2.2.16.1 lockGain()

The RTL signals for *lockGain()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_LOCK_GAIN_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 22 RTL signals for *lockGain()*

The dynamic behavior for *lockGain()* **is specified by** the following chronogram:

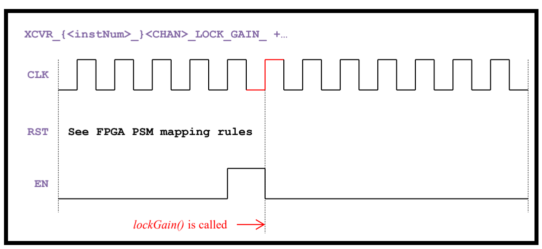

Figure 18 Dynamic behavior for *lockGain()*

2.2.16.2 unlockGain()

The RTL signals for *unlockGain()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_UNLOCK_GAIN_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 23 RTL signals for *unlockGain()*

The dynamic behavior for *unlockGain()* **is specified by** the following chronogram:

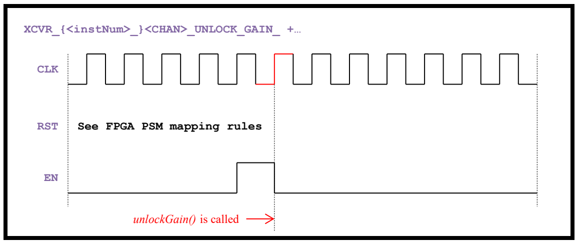

Figure 19 Dynamic behavior for *unlockGain()*

2.2.17 Transceiver::TransceiverTime::TimeAccess

2.2.17.1 getCurrentTime()

The RTL signals for *getCurrentTime()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_GET_CURRENT_TIME_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.current_time	*FPGA app module*	Two 32-bit vectors

Table 24 RTL signals for *getCurrentTime()*

The dynamic behavior for *getCurrentTime()* **is specified by** the following chronogram:

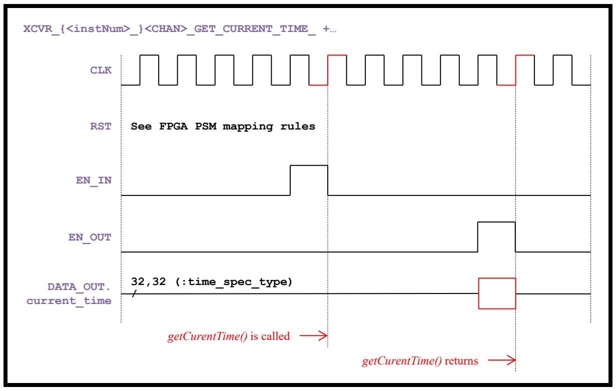

Figure 20 Dynamic behavior for *getCurrentTime()*

2.2.17.2 getLastStartTime()

The RTL signals for *getLastStartTime()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_GET_LAST_START_TIME_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.last_start_time	*FPGA app module*	Two 32-bit vectors
DATA_OUT.last_burst_number	*FPGA app module*	32-bit vector

Table 25 RTL signals for *getLastStartTime()*

The dynamic behavior for *getLastStartTime()* **is specified by** the following chronogram:

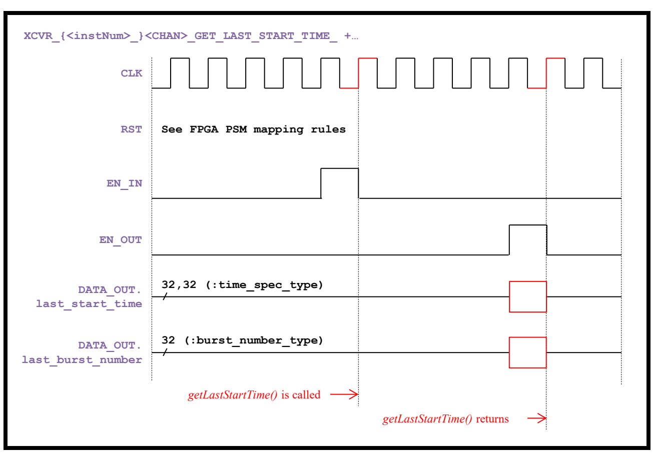

Figure 21 Dynamic behavior for *getLastStartTime()*

2.2.18 Transceiver::Strobing::ApplicationStrobe

2.2.18.1 triggerStrobe()

The RTL signals for *triggerStrobe()* **are specified by** the following table:

RTL signal name XCVR_{<instNum>_}_<CHAN>_TRIGGER_STROBE_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN	*FPGA app module*	1-bit signal

Table 26 RTL signals for *triggerStrobe()*

The dynamic behavior for *triggerStrobe()* **is specified by** the following chronogram:

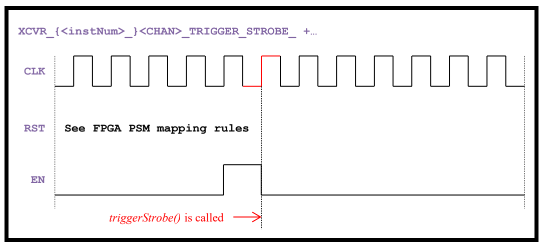

Figure 22 Dynamic behavior for *triggerStrobe()*

3 FPGA PSM constants

This normative section specifies FPGA PSM constants.

3.1 PIM version

In accordance with the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3], the `XCVR_PIM_VERSION` constant **is equal to** `0x020100`.

4 VHDL programming

This section specifies additional normative concepts supported by *FPGA transceivers* programmed using VHDL [Ref5], essentially through specification of VHDL packages to be used by conformant *FPGA façades* and *FPGA applicative modules* (see section 1.2).

The supported VHDL versions are vhdl-93 and all subsequent versions.

The specified VHDL packages have been successfully compiled using:

- Modelsim 10.4a,
- Syntax Checker of Xilinx Vivado 2021.1.

4.1 VHDL library

The specified packages need to be compiled in **xcvr_api** library.

4.2 pkg_xcvr_interface_declaration_properties.vhd

The **pkg_xcvr_interface_declaration_properties.vhd** file **is defined as** the standard VHDL package for handling of interface declaration properties.

The content of **pkg_xcvr_interface_declaration_properties.vhd** **is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Interface declaration properties ([PIM] §4.5)
package pkg_xcvr_interface_declaration_properties is

  -- 1) Variable types properties (CARRIER_FREQ_TYPE, CARRIER_FREQ_TYPE,
CARRIER_FREQ_TYPE)
  -- Enumerated values constants
  -- int16
  constant C_INT16 : std_logic_vector( 1 downto 0) := "00";
  -- int32
  constant C_INT32 : std_logic_vector( 1 downto 0) := "01";
  -- int64
  constant C_INT64 : std_logic_vector( 1 downto 0) := "10";
  -- float32: not mapped
```

```
  -- Setting of enumerated values
  -- enum CARRIER_FREQ_TYPE {int32, int64};
  constant C_CARRIER_FREQ_TYPE : std_logic_vector( 1 downto 0) :=
<C_INT32|C_INT64>; -- USER SELECTION REQUIRED

  -- enum DELAY_TYPE {int32, int64};
  constant C_DELAY_TYPE : std_logic_vector( 1 downto 0) := <C_INT32|C_INT64>;
-- USER SELECTION REQUIRED

  -- enum IQ_TYPE {int16, int32, float32};
  constant C_IQ_TYPE : std_logic_vector( 1 downto 0) := <C_INT16|C_INT32>; --
USER SELECTION REQUIRED

  -- enum CARRIER_FREQ_TYPE {int32, int64};
  constant C_CARRIER_FREQ_VECTOR_SIZE : natural;

  -- enum DELAY_TYPE {int32, int64};
  constant C_DELAY_VECTOR_SIZE : natural;

  -- enum IQ_TYPE {int16, int32, float32};
  constant C_IQ_VECTOR_SIZE : natural;
  -- float32: not mapped

  -- 2) Optional meta-data properties
  -- boolean TX_META_DATA
  constant C_TX_META_DATA : std_logic := '<1|0>'; -- USER SELECTION REQUIRED
  -- boolean RX_META_DATA
  constant C_RX_META_DATA : std_logic := '<1|0>'; -- USER SELECTION REQUIRED

end package pkg_xcvr_interface_declaration_properties;

package body pkg_xcvr_interface_declaration_properties is
  function set_carrier_freq_vector_size(constant_type : in std_logic_vector(1
downto 0)) return natural is
    begin
    if (constant_type = C_INT32) then
      return 32;
    elsif (constant_type = C_INT64) then
      return 64;
    else
      assert false report "<C_INT32|C_INT64>" severity error;
      return 32;
    end if;
  end set_carrier_freq_vector_size;

  function set_delay_vector_size(constant_type : in std_logic_vector(1 downto
0)) return natural is
  begin
    if (constant_type = C_INT32) then
      return 32;
    elsif (constant_type = C_INT64) then
      return 64;
    else
      assert false report "<C_INT32|C_INT64>" severity error;
      return 32;
    end if;
  end set_delay_vector_size;
```

```
  function set_iq_vector_size(constant_type : in std_logic_vector(1 downto
0)) return natural is
  begin
    if (constant_type = C_INT16) then
      return 16;
    elsif (constant_type = C_INT32) then
      return 32;
    else
      assert false report "<C_INT16|C_INT32>" severity error;
      return 16;
    end if;
  end set_iq_vector_size;

  -- Resulting vector sizes computations
  -- enum CARRIER_FREQ_TYPE {int32, int64};
  constant C_CARRIER_FREQ_VECTOR_SIZE : natural :=
set_carrier_freq_vector_size( C_CARRIER_FREQ_TYPE);

  -- enum DELAY_TYPE {int32, int64};
  constant C_DELAY_VECTOR_SIZE : natural := set_delay_vector_size(
C_DELAY_TYPE);

  -- enum IQ_TYPE {int16, int32, float32};
  constant C_IQ_VECTOR_SIZE : natural:= set_iq_vector_size( C_IQ_TYPE);
  -- float32: not mapped

end package body pkg_xcvr_interface_declaration_properties;
```

The user has to adapt **`pkg_xcvr_interface_declartion_properties.vhd`** in selecting the property value in lines commented with `"-- USER SELECTION REQUIRED"`.

4.3 pkg_xcvr_api_types.vhd

The **pkg_xcvr_api_types.vhd** file **is defined as** the standard VHDL package for the *type*s of the *FPGA PSM specification*.

The content of **pkg_xcvr_api_types.vhd** **is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;

library xcvr_api;
use xcvr_api.pkg_xcvr_interface_declaration_properties.all;

package pkg_xcvr_api_types is

  -- Constant reflecting value of XCVR_PIM_VERSION property
  constant C_XCVR_PIM_VERSION : std_logic_vector( 23 downto 0):= X"020100";

  -- BasebandPacket ([PIM] §3.4.2) : the FPGA PSM does not use the concept of
  -- packets for baseband samples exchange; each baseband sample is
transferred
  -- individually within the digital stream

  -- BlockLength ([PIM] §3.4.3)
  -- typedef unsigned long BlockLength;

  subtype block_length_type is std_logic_vector( 31 downto 0);
  constant C_UNDEFINED_BLOCK_LENGTH : block_length_type := (others => '1');

  -- IQ ([PIM] §3.4.11)
  -- typedef <short|long|float> IQ
  subtype iq_type is std_logic_vector( C_IQ_VECTOR_SIZE-1 downto 0);

  -- BasebandSample ([PIM] §3.4.4)
  -- struct BasebandSample {IQ valueI, IQ valueQ};
  type baseband_sample_type is record
    valueI : iq_type;
    valueQ : iq_type;
  end record;

  -- BurstNumber ([PIM] §3.3.5)
  -- typedef unsigned long BurstNumber;
  subtype burst_number_type is std_logic_vector( 15 downto 0);

  -- CarrierFreq ([PIM] §3.4.6), in Hz
  -- typedef <unsigned long|unsigned long long> CarrierFreq
  subtype carrier_freq_type is std_logic_vector( C_CARRIER_FREQ_VECTOR_SIZE-1
downto 0);

  -- const CarrierFreq UndefinedCarrierFreq = <0xFFFFFFFF |
0xFFFFFFFFFFFFFFFF>
  constant C_UNDEFINED_CARRIER_FREQ : carrier_freq_type := (others => '1');

  -- Delay ([PIM] §3.4.7), in ns
  -- typedef <unsigned long|unsigned long long> Delay
  subtype delay_type is std_logic_vector( C_DELAY_VECTOR_SIZE-1 downto 0);

  -- const Delay UndefinedDelay = <0xFFFFFFFF | 0xFFFFFFFFFFFFFFFF>
  constant C_UNDEFINED_DELAY : delay_type := (others => '1');
```

```
  -- Error ([PIM] §3.4.8)
  -- enum Error { errorDelayedTuning, ...
  subtype error_type is std_logic_vector( 3 downto 0);
  constant C_DELAYED_TUNING_ERROR            : error_type := X"0";
  constant C_TUNING_TIMEOUT_ERROR            : error_type := X"1";
  constant C_DELAYED_FIRST_SAMPLE_ERROR      : error_type := X"2";
  constant C_FIRST_SAMPLE_TIMEOUT_ERROR      : error_type := X"3";
  constant C_TRANSMISSION_UNDERFLOW_ERROR    : error_type := X"4";
  constant C_RECEPTION_OVERFLOW_ERROR        : error_type := X"5";
  constant C_SHORTER_TRANSMITTED_BLOCK_ERROR : error_type := X"6";
  constant C_LONGER_TRANSMITTED_BLOCK_ERROR  : error_type := X"7";

  -- Event ([PIM] §3.4.9)
  -- enum Event { eventProcessingStart, ...
  subtype event_type is std_logic_vector( 2 downto 0);
  constant C_PROCESSING_START_EVENT : event_type := "000";
  constant C_PROCESSING_STOP_EVENT  : event_type := "001";
  constant C_SILENCE_START_EVENT    : event_type := "010";
  constant C_SILENCE_STOP_EVEVNT    : event_type := "011";

  -- Gain ([PIM] §3.4.10), in 1/10 dB
  -- typedef short Gain;
  subtype gain_type is std_logic_vector( 15 downto 0); -- signed

  -- const Gain UndefinedGain = 0xFFFF;
  constant C_UNDEFINED_GAIN : gain_type := (others => '1');

  -- MetaData ([PIM] §3.4.12) : specified by pkg_xcvr_metadata_types.vhd

  -- PacketLength ([PIM] §3.4.13): not mapped

  -- SampleNumber ([PIM] §3.4.14)
  -- typedef unsigned long SampleNumber;
  subtype sample_number_type is std_logic_vector( 31 downto 0);

  -- StrobeSource ([PIM] §3.4.15)
  -- enum StrobeSource { ApplicationStrobe,...
  subtype strobe_source_type is std_logic_vector( 3 downto 0);
  constant C_APPLICATION_STROBE : strobe_source_type := X"0";
  constant C_TIME_REF_PPS       : strobe_source_type := X"1";
  constant C_GNSS_PPS           : strobe_source_type := X"2";
  constant C_USER_STROBE_1      : strobe_source_type := X"3";
  constant C_USER_STROBE_2      : strobe_source_type := X"4";
  constant C_USER_STROBE_3      : strobe_source_type := X"5";
  constant C_USER_STROBE_4      : strobe_source_type := X"6";

  -- TimeSpec ([PIM] §3.4.16)
  -- struct TimeSpec { unsigned long seconds, unsigned long nanoseconds};
  type time_spec_type is record
    seconds     : std_logic_vector( 31 downto 0);   -- in seconds
    nanoseconds : std_logic_vector( 31 downto 0);   -- in ns (<1.000.000.000)
  end record;

  -- const TimeSpec UndefinedTimeSpec = {0xFFFFFFFF, 0xFFFFFFFF};
  constant C_UNDEFINED_TIME_SPEC : time_spec_type := (others => (others =>
'1'));
```

```
  -- TuningPreset ([PIM] §3.4.17)
  -- typedef unsigned short TuningPreset;
  subtype tuning_preset_type is std_logic_vector( 15 downto 0);

  -- const TuningPreset UndefinedTuningPreset = 0xFFFF;
  constant C_UNDEFINED_TUNING_PRESET : tuning_preset_type := (others => '1');

end package pkg_xcvr_api_types;
```

4.4 pkg_xcvr_metadata_types.vhd

The file **`pkg_xcvr_metadata_types.vhd`** **is defined as** the standard VHDL package used to declare meta-data *types*.

It is applicable if **`TX_META_DATA`** or **`RX_META_DATA`** is equal to **`TRUE`**.

The content of **`pkg_xcvr_metadata_types.vhd`** **is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;

library xcvr_api;
use xcvr_api.pkg_xcvr_interface_declaration_properties.all;
use xcvr_api.pkg_xcvr_api_types.all;

package pkg_xcvr_metadata_types is

  -- MetaData ([PIM] §3.4.12)
  -- typedef struct TxMetaData (keep if C_TX_META_DATA = '1')
  type tx_metadata_type is record
    <user-defined>;   -- USER-DEFINED FIELDS
  end record;

  -- typedef struct RxMetaData (keep if C_RX_META_DATA = '1')
  type rx_metadata_type is record
    <user-defined>;   -- USER-DEFINED FIELDS
  end record;

end package pkg_xcvr_metadata_types;
```

The user has to adapt **`pkg_xcvr_metadata_types.vhd`** to declare the fields composing ***`tx_metadata_type`*** and/or ***`rx_metadata_typelines`*** using the lines identified with the comment `"-- USER-DEFINED FIELDS"`.

4.5 pkg_xcvr_primitives_parameters.vhd

The file **pkg_xcvr_primitives_parameters.vhd** **is defined as** the standard VHDL package for the *parameters* of the *FPGA PSM specification primitives*.

For *primitives* with optional implementation of *exceptions* specified by the *PIM specification*, the corresponding dedicated 1-bit RTL signal is not specified in the VDHL file.

The content of **pkg_xcvr_primitives_parameters.vhd** **is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;

library xcvr_api;
use xcvr_api.pkg_xcvr_api_types.all;
use xcvr_api.pkg_xcvr_metadata_types.all;

package pkg_xcvr_primitives_parameters is

  -- Management::Reset ([PIM] §2.4.1.1)
  -- reset() has no parameters
  -- PIM Exceptions: none

  -- Management::RadioSilence ([PIM] §2.4.1.2)
  -- startRadioSilence() and stopRadioSilence() have no parameters
  -- PIM Exceptions: none

  -- BurstControl::DirectCreation ([PIM] §2.4.2.1)
  -- startBurst()
  type start_burst_in is record
    requested_length: block_length_type;
  end record;
  -- PIM Exceptions: MinBlockLength, MaxBlockLength

  -- BurstControl::RelativeCreation ([PIM] §2.4.2.2)
  -- scheduleRelativeBurst()
  type schedule_relative_burst_in_type is record
    requested_alternate : std_logic;
    requested_delay : delay_type;
    requested_length : block_length_type;
  end record;
  -- PIM Exceptions: NoAlternateReferencing, MinFromPrevious,
MaxFromPrevious,
  -- MinBlockLength, MaxBlockLength, RelativeMILT

  -- BurstControl::AbsoluteCreation ([PIM] §2.4.2.3)
  -- scheduleAbsoluteBurst()
  type schedule_absolute_burst_in_type is record
    requested_start_time : time_spec_type;
    requested_length : block_length_type;
  end record;
  -- PIM Exceptions: MaxNanoseconds, MinBlockLength, MaxBlockLength,
  -- AbsoluteMILT
```



```
-- BurstControl::StrobedCreation ([PIM] §2.4.2.4)
-- scheduleStrobedBurst()
type schedule_strobed_burst_in_type is record
  requested_strobe_source : strobe_source_type;
  requested_delay : delay_type;
  requested_length : block_length_type;
end record;
-- PIM Exceptions: StrobeSource, MinFromStrobe, MaxFromStrobe,
-- MinBlockLength, MaxBlockLength

-- BurstControl::Termination ([PIM] §2.4.2.5)
-- setBlockLength()
type set_block_length_in_type is record
  requested_length : block_length_type;
end record;
-- PIM Exceptions: NoOngoingProcessing, MinBlockLength, MaxBlockLength

-- stopBurst() has no parameter
-- PIM Exceptions: NoOngoingProcessing

-- BasebandSignal::SamplesReception ([PIM] §2.4.3.1)
-- pushRxBlock(), replacing pushRxPacket()
type push_rx_block_in_sample_type is record
  rx_baseband_sample : baseband_sample_type;
end record;
type push_rx_block_in_metadata_type is record
  rx_meta_data : rx_metadata_type;
end record;
-- PIM Exceptions: none

-- BasebandSignal::SamplesTransmission ([PIM] §2.4.3.2)
-- pushTxBlock(), replacing pushTxPacket()
type push_tx_block_in_sample_type is record
  tx_baseband_sample : baseband_sample_type;
end record;
type push_tx_block_in_metadata_type is record
  tx_meta_data : tx_metadata_type;
end record;
-- PIM Exceptions: MaxTxPacketsLength, TxPacketsMILT

-- BasebandSignal::RxPacketsLengthControl ([PIM] §2.4.3.3): not mapped
-- setRxPacketsLength(): not mapped

-- Tuning::InitialTuning ([PIM] §2.4.4.1)
-- setTuning()
type set_tuning_in_type is record
  requested_preset : tuning_preset_type;
  requested_frequency : carrier_freq_type;
  requested_gain : gain_type;
  requested_burst_number : burst_number_type;
end record;
-- PIM Exceptions: MaxTuningPreset, MinCarrierFreq, MaxCarrierFreq,
-- MinGain, MaxGain, TuningMILT
```

```
  -- Tuning::Retuning ([PIM] §2.4.4.2)
  -- retune()
  type retune_in_type is record
    requested_frequency : carrier_freq_type;
    requested_gain : gain_type;
    requested_delay : delay_type;
  end record;
  -- PIM Exceptions: NoOngoingProcessing, MinCarrierFreq, MaxCarrierFreq,
  -- MinGain, MaxGain, MinFromOnGoing, MaxFromOnGoing, RetuningMILT

  -- Notifications::Events ([PIM] §2.4.5.1)
  -- notifyEvent()
  type notify_event_in_type is record
    notified_event : event_type;
  end record;
  -- PIM Exceptions: none

  -- Notifications::Errors ([PIM] §2.4.5.2)
  -- notifyError()
  type notify_error_in_type is record
    notified_error : error_type;
  end record;
  -- PIM Exceptions: none

  -- GainControl::GainChanges ([PIM] §2.4.6.1)
  -- indicateGain()
  type indicate_gain_in_type is record
    new_gain : gain_type;
    first_valid_sample : sample_number_type;
  end record;
  -- PIM Exceptions: none

  -- GainControl::GainLocking ([PIM] §2.4.6.2)
  -- lockGain() and unlockgain() have no parameter
  -- PIM Exceptions: NoOngoingProcessingException

  -- TransceiverTime::TimeAccess ([PIM] §2.4.7.1)
  -- getCurrentTime()
  type get_current_time_out_type is record
    current_time : time_spec_type;
  end record;
  -- PIM Exceptions: none

  -- getLastStartTime()
  type get_last_start_time_out_type is record
    last_start_time : time_spec_type;
    last_burst_number : burst_number_type;
  end record;
  -- PIM Exceptions: none

  -- Strobing::ApplicationStrobe ([PIM] §2.4.8.1)
  -- triggerStrobe() has no parameter
  -- PIM Exceptions: none

end package pkg_xcvr_primitives_parameters;
```

5 References

5.1 Referenced documents

[Ref1] *Transceiver Facility PIM Specification*, The Wireless Innovation Forum, WINNF-TS-0008 V2.1.1, 20 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Specifications/WINNF-TS-0008-V2.1.1.pdf

[Ref2] *Principles for WInnForum Facility Standards*, The Wireless Innovation Forum, WINNF-TR-2007, V1.0.0, 13 October 2020
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2007-V1.0.0.pdf

[Ref3] *WInnForum Facilities PSMs Mapping Rules*, The Wireless Innovation Forum, WINNF-TR-2008, V1.0.1, 18 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2008-V1.0.1.pdf

[Ref4] *Register-transfer level*, Wikipedia
URL: https://en.wikipedia.org/wiki/Register-transfer_level

[Ref5] Standard VHDL Language Reference Manual, IEEE, 2002
URL: http://ieeexplore.ieee.org/document/1003477/

The URLs above were successfully accessed at release date.

END OF THE DOCUMENT