FlexRAN 5G New Radio Field Programmable Gate Array

User Guide

April 2019

Revision 3.0

Intel Confidential

Document Number: 603578, Rev. 3.0

Contents

Figures

Tables

Revision History

Revision	Description	Date
3.0	Changed Inbound Rx data format for FlexRAN Software Release v19.03	April 2019
2.0	Added Front Haul FPGA description for Release v18.12	December 2018
1.0	Initial release for FlexRAN Software Release v18.09	October 2018

§

1.0 Introduction

FlexRAN 5G New Radio (NR) Forward Error Correction (FEC) Field Programmable Gate Array (FPGA) is on a Peripheral Component Interconnect-express* (PCIe*) card as part of the FlexRAN 5G NR platform and implements encoding and decoding.

FlexRAN 5G NR Front Haul (FH) Field Programmable Gate Array (FPGA) is on a Peripheral Component Interconnect-express* (PCIe*) card as part of the FlexRAN 5G NR platform and implements Orthogonal Frequency Division Multiplexing (OFDM) signal generation, degeneration and front end processing for received Physical Random Access Channel (PRACH) signals.

1.1 Scope

This document explains how to set up, run, and test the FlexRAN 5G NR FEC FPGA and FlexRAN 5G NR FH FPGA Environment.

1.2 Intended Audience

The intended audiences are designers and engineers working with the 5G NR FPGA.

1.3 Terminology

Table 1. Terminology

Term	Description
NR	New Radio
FEC	Forward Error Correction
FH	Front Haul
FPGA	Field-Programmable Gate Array
JTAG	Joint Test Action Group
OFDM	Orthogonal Frequency Division Multiplexing
PCIe*	Peripheral Component Interconnect-express*
PRACH	Physical Random Access Channel
SOF	SRAM Object File
SRAM	Static Random Access Memory

1.4 References and Resources

Table 2. Reference Documents and Resources

Document or Reference	Document No./Location
DE5a-Net Arria® FPGA Development Kit User Manual	https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=231&No=970&PartNo=4
TR10a-HL FPGA Development Kit User Manual	https://www.intel.com/content/dam/www/programmable/us/en/portal/dsn/42/doc-us-dsnbk-42-5703181507555-tr10a-hl-user-manual.pdf
Download Center for FPGAs	https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html

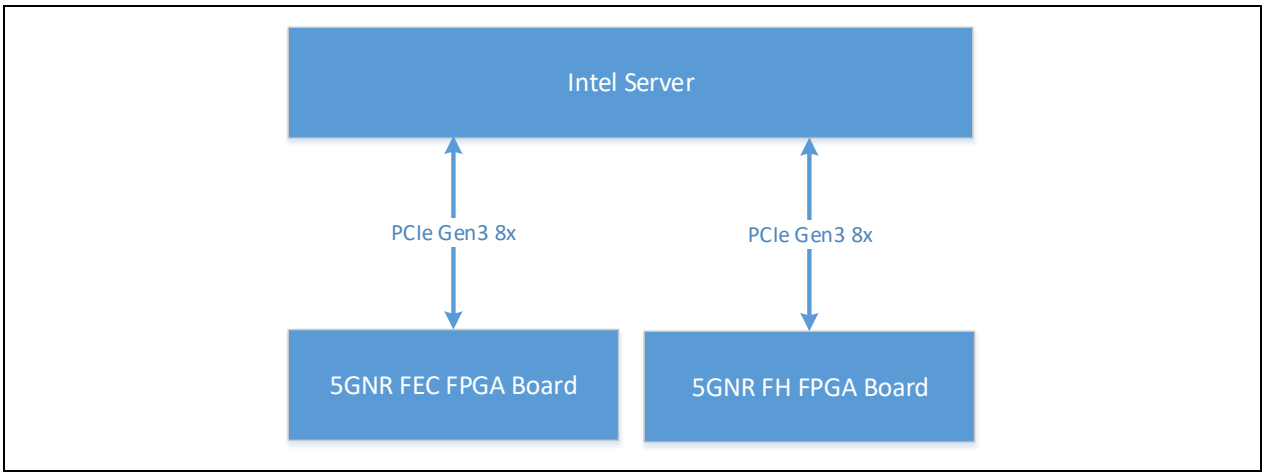
§

2.0 System Overview

Figure 1 shows the FlexRAN system setup. The 5G NR FEC FPGA and 5G NR FH FPGA cards are used as Intel® Server Platform peripheral boards.

The boards are supported by both Terasic* DE5a-NET Arria® 10 FPGA and Terasic* `TR10A_HL FPGA` kits.

Figure 1. System Setup – HW Board



The Intel® Server Platform runs 5G NR software for the FEC FPGA and FH FPGA test. The server and the FPGA card transfer data through the PCIe* Gen3 x8 interfaces.

§

3.0 Hardware Configuration

To run NR FEC FPGA or NR FH FPGA, the following configuration and procedure must be followed.

3.1 Terasic* Board Hardware Setup

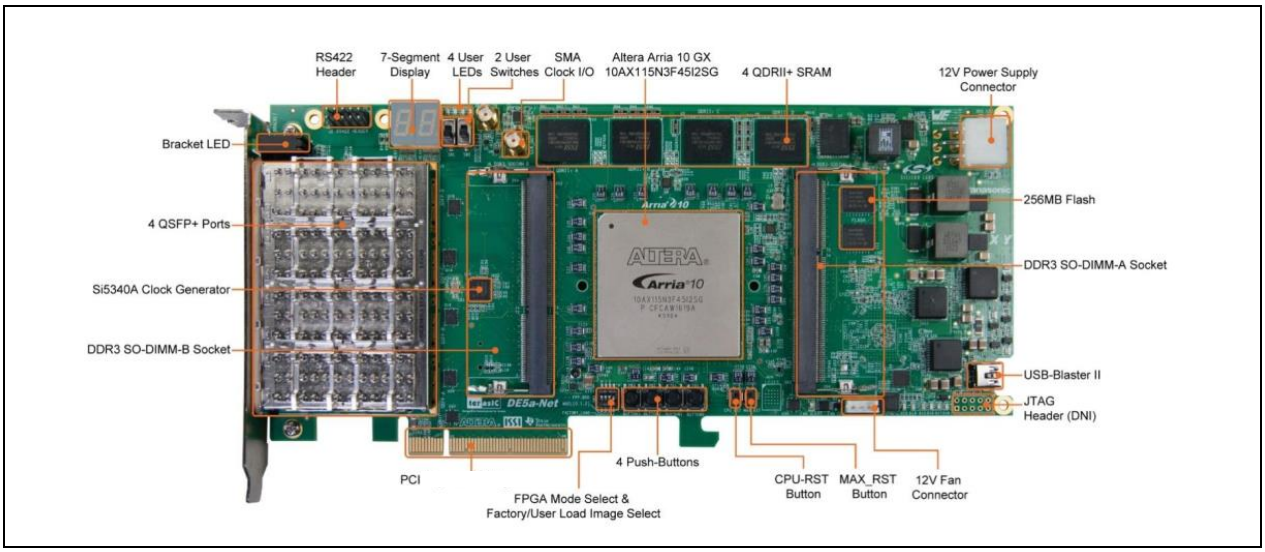
In this section, differences between the DE5a-Net and TR10a-HL Intel® Arria® 10 FPGA Development Kit are described and their configurations addressed.

3.1.1 DE5a-Net Arria® 10 FPGA Development Kit

Set Pin4 of "FPGA mode select" switch to **0** (the default position is **1**) to activate FPGA's User Image Load mode. Then, the Flash* image will be loaded automatically by the FPGA in this mode. Refer to Figure 2 for the switch position on the board, indicators **0** and **1** are printed on the board.

Use USB-Blaster II* to connect the PC server to the FPGA card to program or flash the FPGA image.

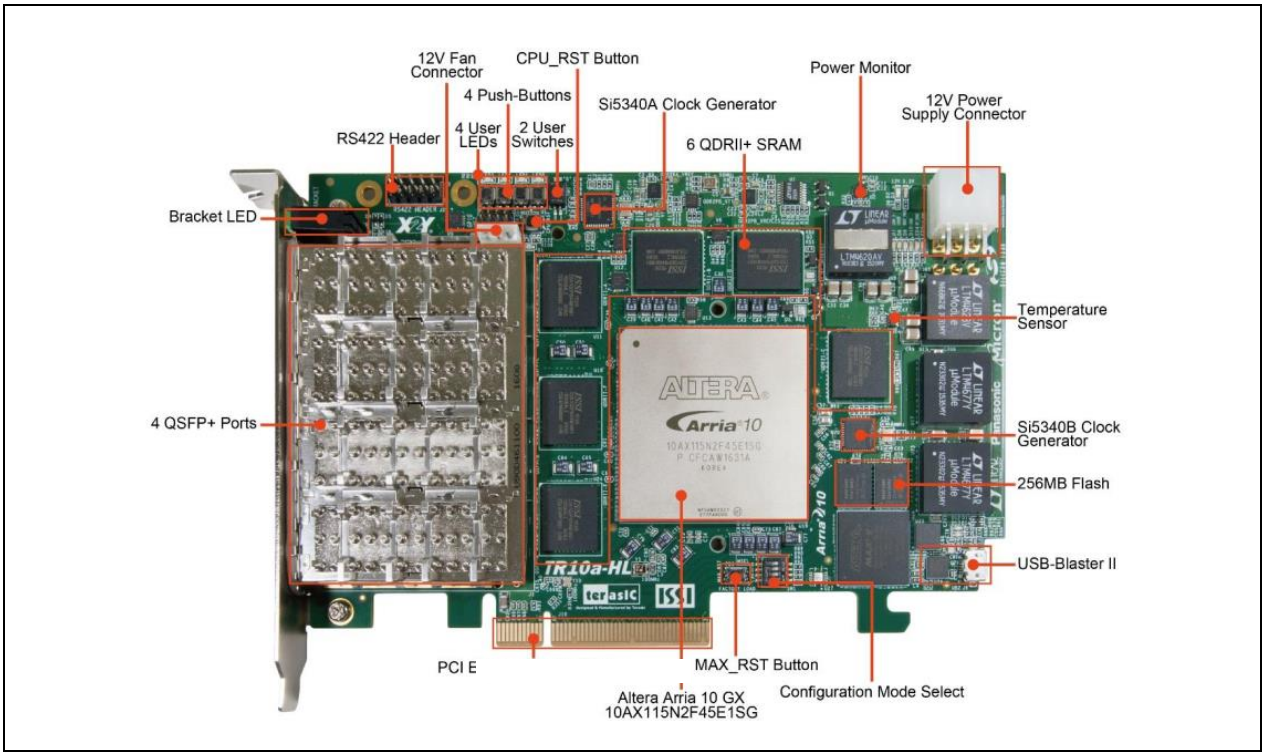
Figure 2. Terasic* DE5a-Net



3.1.2 Terasic* TR10a-HL Development Kit

Set Pin4 of the "FPGA mode select" switch to **0** (the default position is **1**) to activate FPGA's User Image Load mode. Then, the Flash* image will be loaded automatically by the FPGA in this mode. Refer to Figure 3 for the "FPGA Mode Select" switch position on the board, indicators **0** and **1** are printed on the board. Then, use a USB cable to connect the PC server to the FPGA card to program / flash the FPGA image.

Figure 3. Terasic* TR10a-HL Board



§

Appendix A Updating FPGA image

For updating the 5G NR FEC FPGA and 5G NR FH FPGA images, use the Intel® Quartus® Prime software v16.1 (or later versions) and follow the instructions in this section. Refer to *Download Center for FPGAs* in Table 2 for Intel® Quartus® Prime software and documents.

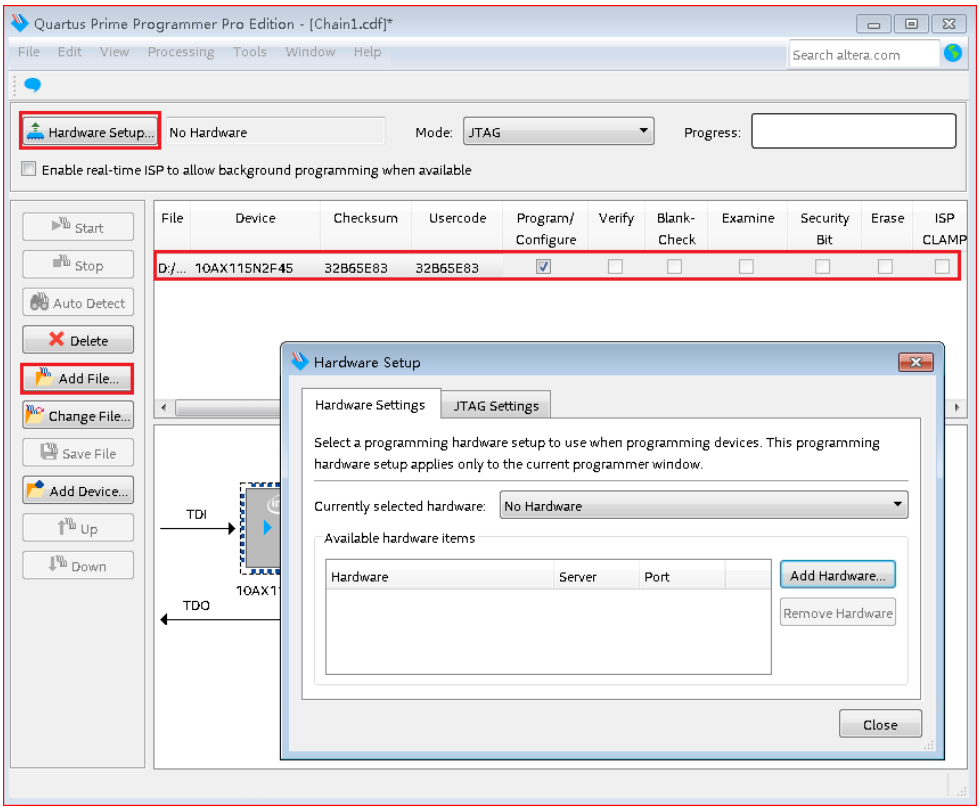
There are two ways to update the FPGA image, directly download the FPGA image to the FPGA or download the FPGA image to Flash*. If you are using the direct download method, the FPGA image will be lost when the server powers off. However, if downloaded to Flash, the FPGA will automatically load the image from Flash every time the server power cycles.

A.1 Downloading the FPGA Image

The 5G NR FPGA images can be downloaded directly to the FPGA board by using the Intel® Quartus® programmer's tool and the USB-Blaster II* cable, which is provided with the FPGA card (refer to Figure 4). FPGA images for the DE5a and TR10a board are different. Confirm you are using the correct images for the target boards.

1. Connect the USB-Blaster II cable to the FPGA card's USB-Blaster II port shown in Figure 2 and Figure 3.
2. Open the Intel® Quartus® Prime Programmer Pro Edition software.
3. Select **Hardware Setup**.
4. Select the current hardware.
5. Select **Add File**.
6. Select the new FPGA Image. Notice the columns Device, Checksum, and Usercode of the new image (refer to Figure 4).
7. Select **Start** to download the FPGA image. This may take up to 10 to 15 seconds. When the new image download is complete, reboot the server.

Figure 4. Intel® Quartus® Prime Programmer Download FPGA Image



A.2 Program the FPGA Image to Flash*

The flash burning package in this release is needed to update the FPGA image in Flash. Each type of card has its own package. There is one shell file and one batch file in the package. The shell file is called by the batch files. These two files convert the original FPGA image (`.sof`) to Flash (`.flash`) and burn the Flash file into Flash.

When the new FPGA image file is needed to be updated into Flash, follow the steps to modify the two files and update the Flash:

1. Confirm the FPGA is set to User Image Mode (refer to Section 3.1.1).
2. Connect USB-Blaster II* cable to the FPGA card's USB-Blaster II port as described in A.1, Downloading the FPGA Image.
3. Open `flash_program.sh`, modify the released FPGA image `sof` file name to `nr_fec.sof`:

```
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" sof2flash --input=nr_fec.sof --
output=flash_hw.flash --offset=0x02B40000 --pfl --optionbit=0x00030000 --
programmingmode=PS
```

4. Run the Windows batch file `flash_program_xxx.bat`, which downloads `DE5a_NET_PFL.sof` or `TR10A_HL_PFL.sof` and calls the shell script `flash_program.sh` to generate the `*.flash` files and program them into Flash.

a. For the DE5a_NET Development Kit, name the loader program **`DE5a_NET_PFL.sof`.**

```
%QUARTUS_BIN%\\quartus_pgm.exe -m jtag -c 1 -o "p;DE5a_NET_PFL.sof"

@ set SOPC_BUILDER_PATH=%SOPC_KIT_NIOS2%+%SOPC_BUILDER_PATH%

@ "%QUARTUS_BIN%\\cygwin\bin\bash.exe" --rcfile
".\flash_program_bts.sh"
```

b. For TR10a-HL Board, name the loader program **`TR10A_HL_PFL_sof`.**

```
%QUARTUS_BIN%\\quartus_pgm.exe -m jtag -c 1 -o "p; TR10A_HL_PFL.sof"

@ set SOPC_BUILDER_PATH=%SOPC_KIT_NIOS2%+%SOPC_BUILDER_PATH%

@ "%QUARTUS_BIN%\\cygwin\bin\bash.exe" --rcfile
".\flash_program_bts.sh"
```

Then confirm the .sh file name is correct and the Joint Test Action Group (JTAG) number matches with the FPGA board, refer to Figure 5.

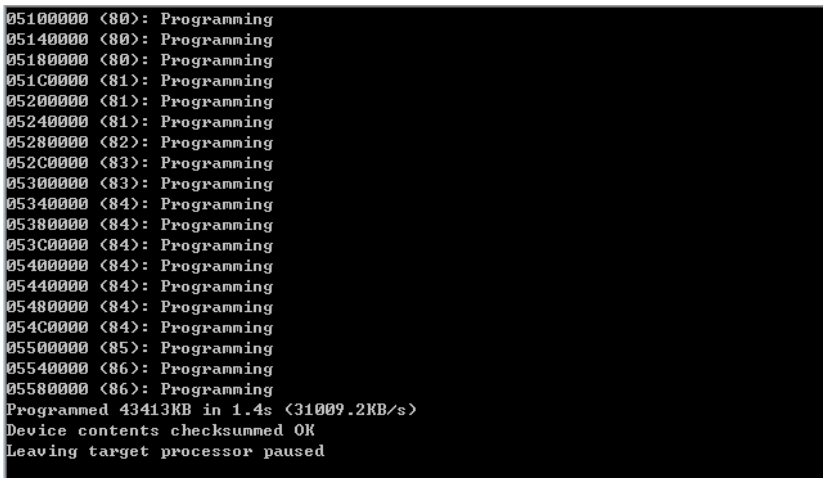
Figure 5. Intel® Quartus® Prime Programmer Flash Start Log

```
Info: ************************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 16.1.1 Build 200 11/30/2016 SJ Pro Edition
    Info: Copyright (C) 2016  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel MegaCore Function License Agreement, or other
    Info: applicable license agreement, including, without limitation,
    Info: that your use is for the sole purpose of programming logic
    Info: devices manufactured by Intel and sold by Intel or its
    Info: authorized distributors.  Please refer to the applicable
    Info: agreement for further details.
    Info: Processing started: Fri Mar 03 16:46:14 2017
Info: Command: quartus_pgm -m jtag -c 1 -o p;DE5a_NET_PFL.sof
Info (213045): Using programming cable "DE5 [USB-1]"
Info (213011): Using programming file DE5a_NET_PFL.sof with checksum 0x30CA4992
for device 10AX115N2F45@1
Info (209060): Started Programmer operation at Fri Mar 03 16:47:02 2017
Info (209016): Configuring device index 1
Info (209017): Device 1 contains JTAG ID code 0x02E660DD
Info (209007): Configuration succeeded -- 1 device(s) configured
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Fri Mar 03 16:47:17 2017
```

5. Wait for 30-50 minutes until completion.

6. Power off and power on again. The FPGA will get the image from Flash at User Image Load mode. Refer to Figure 6.

Figure 6. Intel® Quartus® Programmer Flash Completion Log



A.2.1 Batch File For Flash Burning

In the batch file for flash burning, a general Flash loader FPGA image named `DE5a_NET_PFL.sof` or `TR10A_HL_PFL.sof` should be programmed to the FPGA board through the USB cable:

```
@ REM ####################################
@ REM # Variable to ignore <CR> in DOS
@ REM # line endings
@ set SHELLOPTS=igncr

@ REM ####################################
@ REM # Variable to ignore mixed paths
@ REM # i.e. G:/$SOPC_KIT_NIOS2/bin
@ set CYGWIN=nodosfilewarning

@set QUARTUS_BIN=%QUARTUS_ROOTDIR%\\bin
@if exist %QUARTUS_BIN%\\quartus_pgm.exe (goto DownLoad)

@set QUARTUS_BIN=%QUARTUS_ROOTDIR%\\bin64
@if exist %QUARTUS_BIN%\\quartus_pgm.exe (goto DownLoad)

:: Prepare for future use (if exes are in bin32)
@set QUARTUS_BIN=%QUARTUS_ROOTDIR%\\bin32
```

```
:DownLoad
%QUARTUS_BIN%\\quartus_pgm.exe -m jtag -c 1 -o "p;TR10A_HL_PFL.sof"
@ set SOPC_BUILDER_PATH=%SOPC_KIT_NIOS2%+%SOPC_BUILDER_PATH%
@ "%QUARTUS_BIN%\\cygwin\bin\bash.exe" --rcfile ".\flash_program.sh"


Pause
```

A.2.2 Shell Script File for Flash Burning

In the flash burning package, the shell script file is called by the batch file. The shell script file transforms `nr_fec.sof` into `flash_hw.flash`, running the following code:

```
# file: nios2_sdk_shell_bashrc


#conver to .flash as factory image
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" sof2flash --input=nr_fec.sof --
output=flash_hw.flash --offset=0x02B40000 --pfl --optionbit=0x00030000 --
programmingmode=PS
#"$SOPC_KIT_NIOS2/nios2_command_shell.sh" elf2flash --base=0x0 --end=0x0FFFFFFF --
reset=0x05E40000 --input=HELLO_NIOS.elf --output=flash_sw.flash --
boot=$SOPC_KIT_NIOS2/components/altera_nios2/boot_loader_cfi.srec


#Programming with .flash
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0
flash_hw.flash
#"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0
flash_sw.flash
```

§

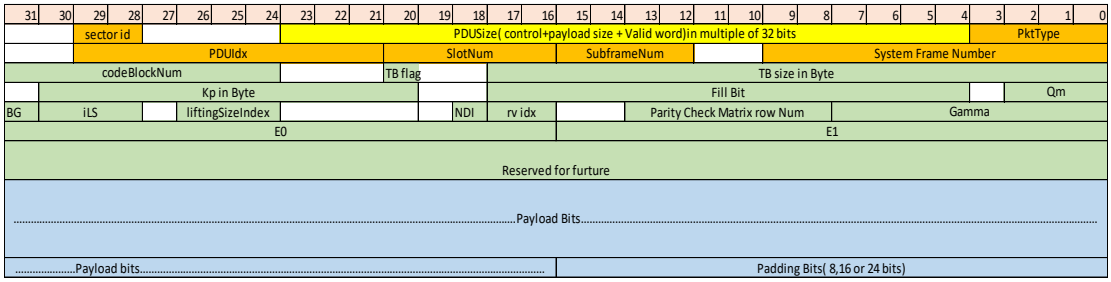
Appendix B NR FEC FPGA Interface Data Formats

The following shows the data formats of FEC interface from the FPGA point of view.

B.1 Inbound TX Data Formats

Inbound TX data is received from the Host software for encoding (refer to Table 3 for display formatting).

Table 3 Inbound TX Data Formats



Field Description:

`PktType` -> 4 bits. 0010: TX TB packet, the payload is the raw bit data sent by the CPU to the FPGA for encoding.

`sector id` =>2 bits. Local cell id. Range: 0-3

`PDUIdx` => 9 bits. 0-511(256 USER x 2 TB)

`System Frame Number` => 10 bits. Range: 0-1023

`Subframe Num` => 4 bits. Range: 0-9

`SlotNum` => 5 bits. Range: 0-31

`TB size in Byte`=> 18 bits. Range: 0-262143

`codeBlockNum` => 8 bits. Range: 0-255

`Gamma` => 8 bits. Range: 0-255

`Kp in byte`=> 11 bits. Range: 0-1055

`Fill Bit Num`=> 14 bits. Range: 0-8447

Qm=> 3 bits. 0-> BPSK, 1-> QPSK, 2-> 16QAM, 3-> 64QAM, 4-> 256QAM

BG=>1 bits. 0-> BaseGraph1, 1->BaseGraph2

iLS=>3 bits. Lifting set Index, Range: 0-7

liftingSizeIndex=>3 bits. Lifting size Index, Range: 0 to 7

NDI=>1 bit. New data indicator

rv idx=>2 bits. Range: 0-3. Redundancy version index

Parity Check Matrix row Num=>BG 1 range: 4-46, BG 2 range: 6-42

TB flag=> 1 bit. Indicate last TB of slot

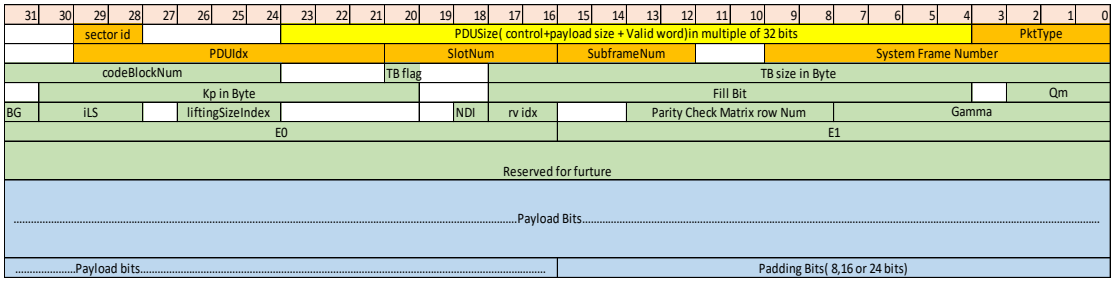
E0=>16 bits. Range: 0-65535

E1=>16 bits. Range: 0-65535

B.2 Outbound TX Data Formats

Outbound TX data is the result data encoded by the FPGA. The Host software will retrieve the data from the FPGA (refer to Table 4 for display format).

Table 4. Outbound TX Data Formats



Field Description:

PktType => 4 bits. 0011: TX Symbol Packet, FPGA send encoded bit to CPU

sector ID =>2 bits. Local cell id. Range: 0-3

PDUIdx => 9 bits. 0-511 (256 USER x2TB)

System Frame Number => 10 bits. Range: 0-1023

Subframe Num => 4 bits. Range: 0-9

SlotNum => 5 bits. Range: 0-31

`TB size in Byte`=> 18 bits. Range: 0-262143

`codeBlockNum` => 8 bits. Range: 0-255

`Gamma` => 8 bits. Range: 0-255

`Kp in byte`=> 11 bits. Range: 0-1055

`Fill Bit Num`=> 14 bits. Range: 0-8447

`Qm`=> 3 bits. 0-> BPSK, 1-> QPSK, 2-> 16QAM, 3-> 64QAM, 4-> 256QAM

`BG`=>1 bit. 0-> BaseGraph1, 1->BaseGraph2

`iLS`=>3 bits. Lifting set Index, Range: 0 to 7

`liftingSizeIndex`=>3bits. Lifting Size Index. Range: 0-7

`NDI`=>1 bit. New data indicator

`rv idx`=>2 bits. Range: 0-3. Redundancy version index

`Parity Check Matrix row Num`=> BG 1 range: 4-46, BG 2 range: 6-42

`TB flag`=> 1 bit. Indicate last TB of slot

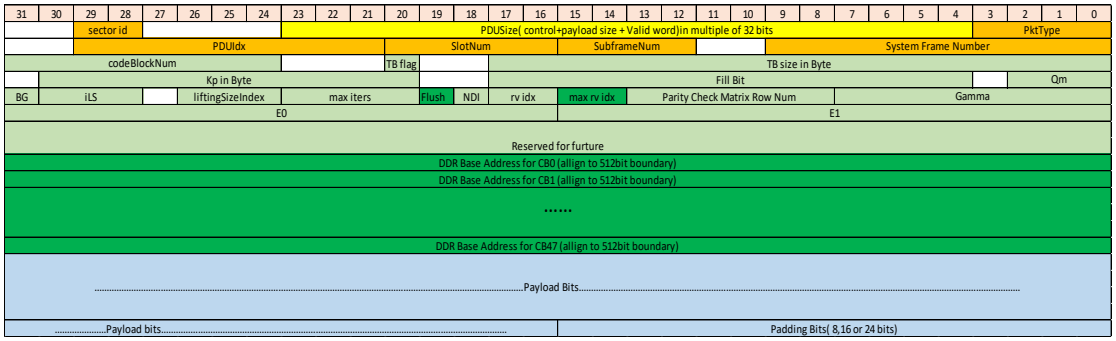
`E0`=>16 bits. Range: 0-65535

`E1`=>16 bits. Range: 0-65535

B.3 Inbound RX Data Formats

Inbound RX data is received from the Host software for decoding. Refer to Table 5 for display format.

Table 5. Inbound RX Data Formats



Field Description:

`PktType` => 4 bits. 0101: RX Symbol Packet, CPU send descrambled to FPGA for decoding

`sector ID` =>2 bits. local cell id. Range: 0-3

`PDUIdx` => 9 bits. 0-511(256 USER x 2TB)

`System Frame Number` => 10 bits. Range: 0-1023

`Subframe Num` => 4 bits. Range: 0-9

`SlotNum` => 5 bits. Range: 0-31

`TB size in Byte`=> 18 bits. Range: 0-262143

`codeBlockNum` => 8 bits. Range: 0-255

`Gamma` => 8 bits. Range: 0-255

`Fill Bit Num`=> 14 bits. Range: 0-8447

`Qm`=> 3 bits. 0-> BPSK, 1-> QPSK, 2-> 16QAM, 3->64QAM, 4->256QAM

`E0`=> 16 bits. Range: 0-65535

`E1`=> 16 bits. Range: 0-65535

`BG`=>1 bit. 0-> BaseGraph1, 1->BaseGraph2

`iLS`=>3 bits. Lifting set Index. Range: 0-7

`max iteration`=> 4 bits. Decorder max iterations. Range: 0-15

`liftingSizeIndex`=> 3 bits. Lifting size Index, Range: 0-7

`NDI`=>1 bit. New data indicator

`rv idx`=>2 bits. Range: 0-3. Redundancy version index

`Parity Check Matrix row Num`=>6 bits. Range: 4-46 for BG1, 6-42 for BG2

`Flush`=>1 bit. If this bit is set to 1, FPGA will clear the previous data in DDR

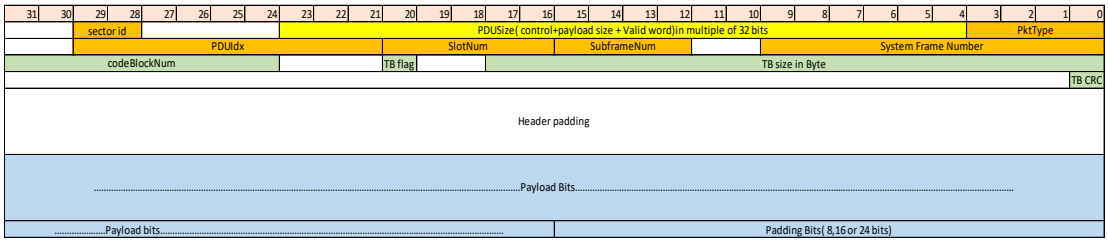
`max rv idx`=> 2 bits. The max redundant version index in previous retransmissions, which is used to decide how many data in DDR need to be read out of DDR memory

`DDR Base Address for CBn`=> The base address in DDR memory where encoder should store CBn's data

B.4 Outbound RX Data Formats

Outbound RX data is the result data decoded by the FPGA. Host software will retrieve data from the FPGA. Table 6 shows the display format.

Table 6. Outbound RX Data Formats



Field Description:

`PktType` => 4 bits. 0111: RX TB packet, FPGA send RX TB data to CPU

`sector id` =>2 bits. Range: 0-3

`PDUIdx` => 9 bits. 0-511(256 USER x 2TB)

`System Frame Number` => 10 bits. Range: 0-1023

`Subframe Num` => 4 bits. Range: 0-9

`SlotNum` => 5 bits. Range: 0-31

`TB size in Byte`=> 18 bits. Range: 0-262143

`codeBlockNum` => 8 bits. Range: 0-255

`TB Flag`=> 1 bit. Range: 0, not last TB, 1: last TB

`TB CRC`=>1 bit. Range: 0, no pass, 1: Pass

§

Appendix C NR FH FPGA Interface Data Formats

The following shows the data formats of FH interface from the FPGA point of view.

C.1 Inbound TX Symbol Data Formats

Inbound TX Symbol Data is received by the FH FPGA from the CPU to generate OFDM symbols and send it to the radio unit (refer to Table 7).

Table 7. Inbound TX Symbol Data Formats for FH FPGA

256-bit Word	31 30 29 28	27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
0	Sector ID (0~2)	Antenna ID (0~3)	PDUSize(control+payload size + Valid word)in multiple of 32 bits			PktType
	Reserved		frame index	Subframe Index (0~9)	Slot Index (0~31)	**Symbol Index (0~13)**
	Reserved					
	Reserved					
	Reserved					
	Reserved					
	Reserved					
	Reserved					
1	Q0 [15:0]			I0 [15:0]		
	Q1 [15:0]			I1 [15:0]		
	Q2 [15:0]			I2 [15:0]		
	Q3 [15:0]			I3 [15:0]		
	Q4 [15:0]			I4 [15:0]		
	Q5 [15:0]			I5 [15:0]		
	Q6 [15:0]			I6 [15:0]		
	Q7 [15:0]			I7 [15:0]		
...	...					
99	Q784 [15:0]			I784 [15:0]		
	Q785 [15:0]			I785 [15:0]		
	Q786 [15:0]			I786 [15:0]		
	Q787 [15:0]			I787 [15:0]		
	Q788 [15:0]			I788 [15:0]		
	Q789 [15:0]			I789 [15:0]		
	Q790 [15:0]			I790 [15:0]		
	Q791 [15:0]			I791 [15:0]		

`PktType` => 4bits. 1001: TX FH Packet

`PDUSize` => 20bits. Fixed to 800 (66*12 + 8) for mmWave system; fixed to 3288 (273*12 + 4 + 8) for the sub6 GHz system

`Frame Index` => 10 bits. Range: 0-99

Note: The Q index will range from 0 to 3275 for the sub6 GHz system.

C.2 Outbound RX Symbol Data Formats

The FH FPGA receives RX data from the radio unit and sends it to the CPU as Outbound RX Symbol Data after the OFDM symbol degeneration process (refer to Table 8).

Table 8. Outbound RX Symbol Data Formats for FH FPGA

256-bit Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Sector ID (0~2)				Antenna ID (0~3)				PDUSize(control+payload size + Valid word)in multiple of 32 bits																				PktType			
	Reserved								frame index															Subframe Index (0~9)					Slot Index (0~31)				Symbol Index (0~13)
	Reserved																															
	Reserved																															
	Reserved																															
	Reserved																															
	Reserved																															
1	Q0 [15:0]																I0 [15:0]															
	Q1 [15:0]																I1 [15:0]															
	Q2 [15:0]																I2 [15:0]															
	Q3 [15:0]																I3 [15:0]															
	Q4 [15:0]																I4 [15:0]															
	Q5 [15:0]																I5 [15:0]															
	Q6 [15:0]																I6 [15:0]															
	Q7 [15:0]																I7 [15:0]															
...	...																															
99	Q784 [15:0]																I784 [15:0]															
	Q785 [15:0]																I785 [15:0]															
	Q786 [15:0]																I786 [15:0]															
	Q787 [15:0]																I787 [15:0]															
	Q788 [15:0]																I788 [15:0]															
	Q789 [15:0]																I789 [15:0]															
	Q790 [15:0]																I790 [15:0]															
	Q791 [15:0]																I791 [15:0]															

`PktType` => 4 bits. 1010: RX FH Packet

`PDUSize` => 20 bits. Fixed to 800 (66*12 + 8) for mmWave system; fixed to 3288 (273*12 + 4 + 8) for the sub6 GHz system.

`Frame Index` => 10 bits. Range: 0-99

Note: The Q index will range from 0 to 3275 for the sub6 GHz system.

C.3 Outbound RX PRACH Data Formats

The FH FPGA receives RX data from the radio unit, and then extracts the PRACH data from it, completing a set of front end processing. The results are sent to the CPU as Outbound RX PRACH Data (refer to Table 9).

Table 9. Outbound RX PRACH Data Formats

256-bit Word	31 30 29 28	27 26 25 24	23	22 21 20 19 18 17 16 15 14 13	12 11 10 9	8 7 6 5 4 3 2 1 0
0	Sector ID (0~2)	Antenna ID (0~3)	Rsv	Frame index	Subframe Index (0~9)	Slot Index (0~31) PktType
	Occassion Num	Reserved	Sequence Num	Reserved	Freq Offset	
	Reserved					
	Reserved					
	Reserved					
	Reserved					
	Reserved					
	Reserved					
1	Q0 [15:0]				I0 [15:0]	
	Q1 [15:0]				I1 [15:0]	
	Q2 [15:0]				I2 [15:0]	
	Q3 [15:0]				I3 [15:0]	
	Q4 [15:0]				I4 [15:0]	
	Q5 [15:0]				I5 [15:0]	
	Q6 [15:0]				I6 [15:0]	
	Q7 [15:0]				I7 [15:0]	
...	...					
N	Q(8*N-8) [15:0]				I(8*N-8) [15:0]	
	Q(8*N-7) [15:0]				I(8*N-7) [15:0]	
	Q(8*N-6) [15:0]				I(8*N-6) [15:0]	
	Q(8*N-5) [15:0]				I(8*N-5) [15:0]	
	Q(8*N-4) [15:0]				I(8*N-4) [15:0]	
	Q(8*N-3) [15:0]				I(8*N-3) [15:0]	
	Q(8*N-2) [15:0]				I(8*N-2) [15:0]	
	Q(8*N-1) [15:0]				I(8*N-1) [15:0]	

`PktType` => 4 bits. 1100: PRACH RX Data Packet

`Freq Offset` => 16 bits. The total frequency offset of PRACH RX data

`Sequence Num` => 4 bits. Range: 1-12

`Occasion Num` => 4 bits. Range: 1-7

`N` => the number of 256 bits of RX Data for one PRACH Slot, N = 18 * `Sequence_Num` * `Occasion_Num`

Note: Each sequence contains 18 words of 256 bits, corresponding to 144 elements of 32 bits of IQ data.

C.4 Inbound RX PRACH CFG Formats

The inbound RX PRACH Configuration (which is shown as CFG) is an interface used by the CPU to set the parameters for the FH FPGA's PRACH RX processing (refer to Table 10).

`PktType` => 4 bits. 1011: PRACH RX CFG Packet.

`k1'` => frequency offset without DC shift, the value is k1+N_gridsize*12/2. for `mmWave`, `N_gridsize` = 66

`k'` => subcarrier offset. For short sequence, k' = 2

`Config Index` => PRACH configure index. Range: 0 to 255

Table 10. Inbound RX PRACH CFG Formats

256-bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																													PktType		
0	Config Index								k'								k1'															
	Reserved																															
	Reserved																															
	Reserved																															
	Reserved																															
	Reserved																															
	Reserved																															

§