HPU Core 4.0 – User Guide (Rev. B)

Author: Francesco Diotalevi, Maurizio Casti

Company: Istituto Italiano di Tecnologia

History of HPU_Core IP:

Ver.	Rev.	Date	Author	Description
1.0		Sep 19th 2016	Francesco Diotalevi	• First Release (DRAFT)
1.1		Jun 14th 2017	Francesco Diotalevi	• Modified the AXIstream module and added some debug ports. Added Reset_DMA_stream bit into CTRL_REG.
2.0		Nov 15th 2017	Francesco Diotalevi	• Bug fixed for the HSSAER Channel Enable. Added AUX Threshold Error register and AUX Rx Counter registers. Some Fixes in HDL code.
2.1		Jun 15th 2018	Francesco Diotalevi	• Enlarged to 24 the SSAER data transfer. Different header coding.
3.0		Aug 9th 2018	Maurizio Casti	• Added the SpiNNlink interface capability
3.1		Aug 24th 2018	Maurizio Casti	• Added the START/STOP and Data Mask Feature to SpiNNlink
3.2		Oct 24th 2018	Francesco Diotalevi	• Splitted FlushFifos in FlushRXFifo and FlushTXFifos. • Modifications in axistream module to premature end a burst transfer. • Added register that counts data in AXI stream bus. • Modified reset value of RX PAER Configuration register (RX_PAER_CFNG_REG). • Enlarged DMA length field to 16 bits.
3.3		Oct 30th 2018	Francesco Diotalevi	• DMA register (DMA_REG) has thi bit 0 fixed to 0. It can be written only with even values.
3.4		Nov 16th 2018	Maurizio Casti	• Absolute Timing Feature for Transmission • Changed some feature about START/STOP Command • Added SpiNNlink Control Register and Status Register
3.5		May 20th 2020	Francesco Diotalevi	• Added Synchronization Fifos to make independent the differential clocks with the Core clock
3.6	**1**	Jan 28th 2021	Maurizio Casti	• Added interception ports (for algorithm insertion) • Updated the list of tags for external sensor (IMU and Cochlea) - see AERSensorsMap.xlxs • GUI renewed
4.0	**18**		Maurizio Casti	• Compatibility extended to ZynqUplus • GTP TX/RX Interface (Serie7 architecture) • GTH RX Interface (Ultrascale architecture) • Timestamp removable from data streaming • AXI Stream dedicate clock domain

History of HPU_Core Documentation:			
Revision	**Date**	**Moified by**	**Description**
--	Omissis	F.Diotalevi, M.Casti	Previous not tracked versions
A	Mar 26th 2021	M.Casti	Table of VHDL Generics updated Figure 0.1 HPU Core block updated
B	Jul 4th 2024	M.Casti	Figure 0.2 HPUCore architecture block diagram updated Figure 0.3 - HPU Core block updated Table 0.1 - updated

Contents

Glossary

Acronym	Meaning
TBD	To Be Defined
TBT	To Be Tested
TBI	To Be Implemented

1 HPUCore implementation parameters (VHDL Generics)

Name	Description	Default	Format	Range
Customization Parameters				
C_FPGA_FAM	FPGA Family ("zynq", "zynquplus") Note: for Artix use "zynq"	"zynquplus"	String	
C_PAER_DSIZE	Size of PAER address	24	Integer	1 - 32
C_RX_HAS_PAER	If true (checked), the RX PAER interface is exposed	true	Boolean	
C_RX_HAS_HSSAER	If true (checked), the RX HSSAER interface is exposed	true	Boolean	
C_RX_HSSAER_N_CHAN	The number of RX HSSAER channels	3	Boolean	1 - 4
C_RX_HAS_GTP	If true (checked), the RX GTP interface is exposed	false	Boolean	
C_RX_HAS_SPNNLNK	If true (checked), the RX SpiNNlink interface is exposed	true	Boolean	
C_TX_HAS_PAER	If true (checked), the TX PAER interface is exposed	true	Boolean	
C_TX_HAS_HSSAER	If true (checked), the TX HSSAER interface is exposed	true	Boolean	
C_TX_HSSAER_N_CHAN	The number of TX HSSAER channels	3	Integer	1 - 4
C_TX_HAS_GTP	If true (checked), the TX GTP interface is exposed	false	Boolean	
C_TX_HAS_SPNNLNK	If true (checked), the TX SpiNNlink interface is exposed	true	Boolean	
C_PSPNNLNK_WIDTH	Size of SpiNNaker parallel data interface	32	Integer	1 - 32
C_DEBUG	If true (checked), Debug Ports are enabled	false	Boolean	
C_S_AXI_DATA_WIDTH	AXI4 Lite Slave Data width	32	Integer	
C_S_AXI_ADDR_WIDTH	AXI4 Lite Slave Address width	8	Integer	
C_RX_PAER_L_SENS_ID	Left Rx PAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER0_L_SENS_ID	Ch0 Left Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER1_L_SENS_ID	Ch1 Left Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER2_L_SENS_ID	Ch2 Left Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER3_L_SENS_ID	Ch3 Left Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_PAER_R_SENS_ID	Right Rx PAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER0_R_SENS_ID	Ch0 Right Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER1_R_SENS_ID	Ch1 Right Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER2_R_SENS_ID	Ch2 Right Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_SAER3_R_SENS_ID	Ch3 Right Rx HSSAER Sensor Type	"000"	BitString	3 bit
C_RX_PAER_A_SENS_ID	Aux Rx PAER Sensor Type	"001"	BitString	3 bit
C_RX_SAER0_A_SENS_ID	Ch0 Aux Rx HSSAER Sensor Type	"001"	BitString	3 bit
C_RX_SAER1_A_SENS_ID	Ch1 Aux Rx HSSAER Sensor Type	"001"	BitString	3 bit
C_RX_SAER2_A_SENS_ID	Ch2 Aux Rx HSSAER Sensor Type	"001"	BitString	3 bit
C_RX_SAER3_A_SENS_ID	Ch3 Aux Rx HSSAER Sensor Type	"001"	BitString	3 bit
C_RX_LEFT_INTERCEPTION	Left RX port interception	false	Boolean	
C_RX_RIGHT_INTERCEPTION	Right RX port interception	false	Boolean	
C_RX_AUX_INTERCEPTION	Aux RX port interception	false	Boolean	
C_HAS_DEFAULT_LOOPBACK	Default loopback settings ports	false	Boolean	
Hidden Parameters				
C_SLV_DWIDTH	CTRL and CTRG Debug port width	32	Integer	

Table 1.1 - HPU_Core implementation parameters (VHDL generics)

2 Introduction

The Head Processor Unit Core (HPU Core) is an AXI peripheral used to manage different input AER or SpiNNlink streaming and transfer the acquired data into memory through DMA interface or by reading registers with Host CPU.
It is also Transmission capable, and permits to send AER or SpiNNlink streaming to external devices.

It has an Axi4 lite bus I/f for writing/reading internal registers and delivers two AXI stream bus @ 32bit (Read and Write lanes). It can be configured with up to 4 input channels and 1 output channel, and each channel can manage PAER, SAER, Gigabit Transceiver, SpiNNlink flows.

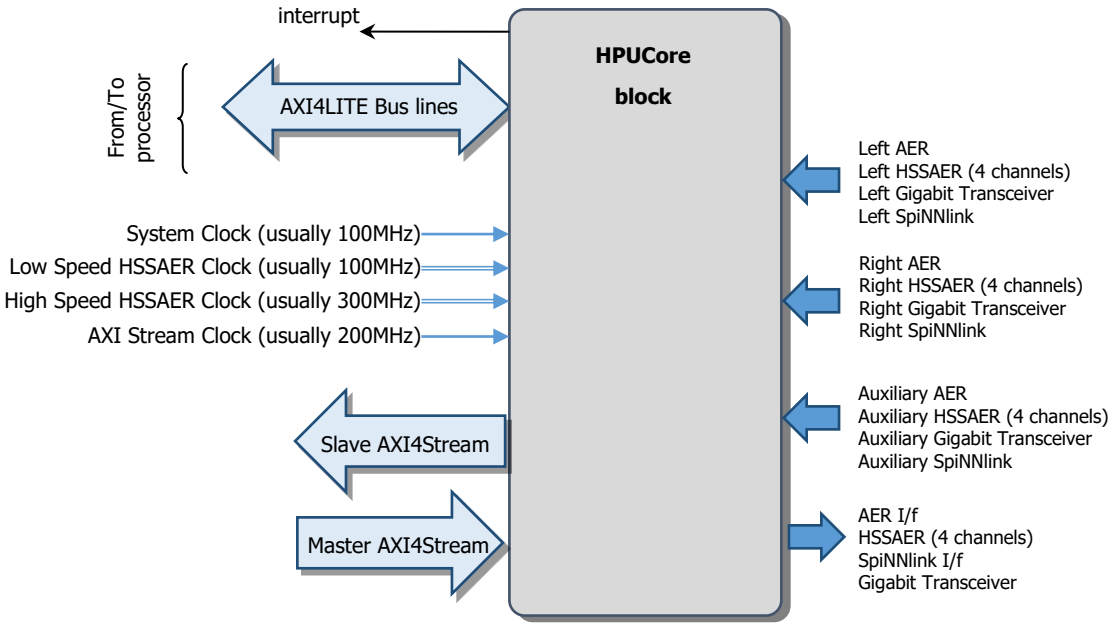


Figure 2.1 - HPUCore block

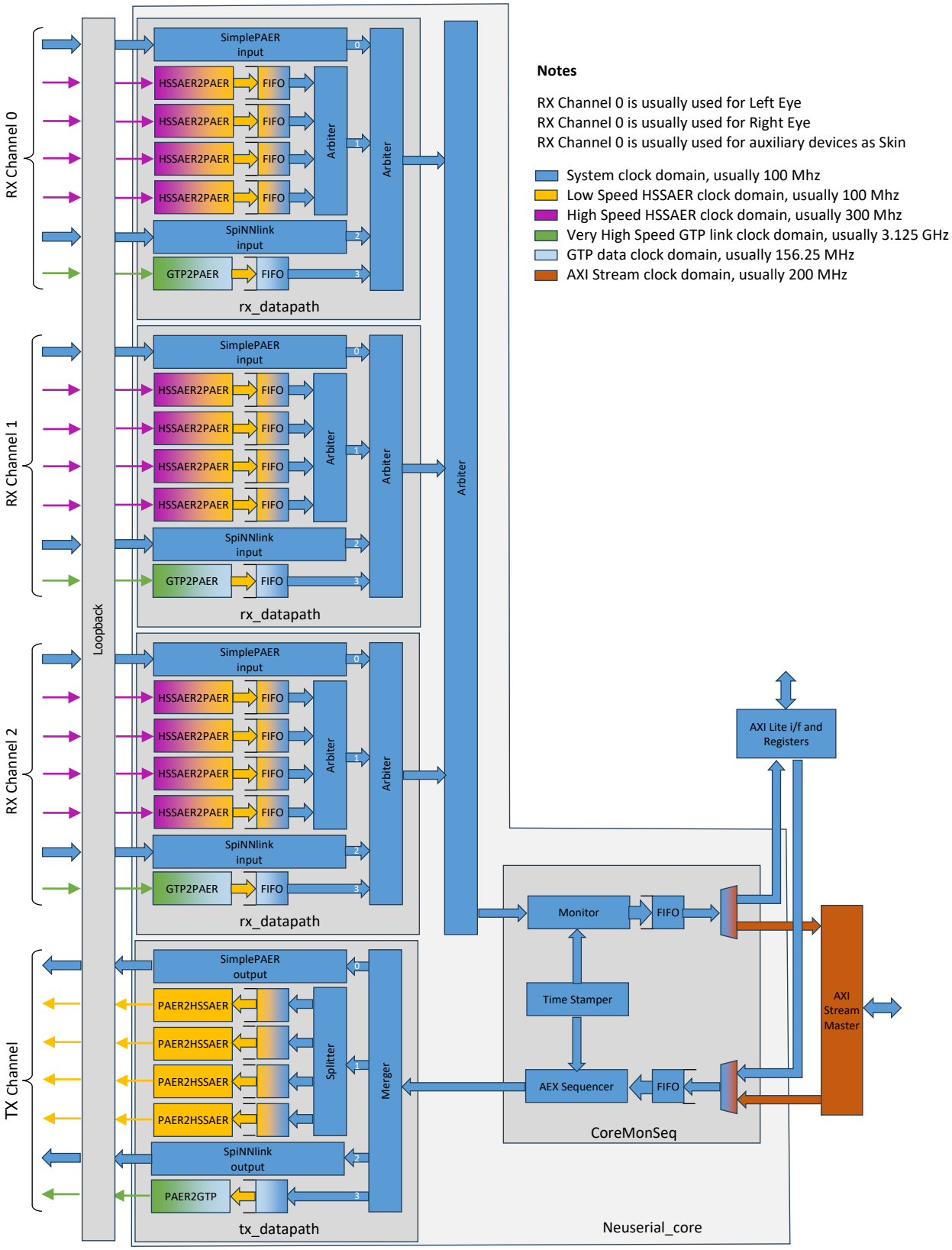
a list of the ports and their description is shown in following Table 2.1.

Comment	Port name	Width	Dir	Description
Interrupt	Interrupt	1	O	Level interrupt active high signal
AXI4 Lite Bus lines	S_AXI_ACLK	1	I	AXI Clock, System clock line
	S_AXI_ARESETN	1	I	AXI Reset active low line
	S_AXI_AWADDR	32	I	AXI Write address
	S_AXI_AWVALID	1	I	Write address valid
	S_AXI_WDATA	32	I	Write data
	S_AXI_WSTRB	4	I	Write strobes
	S_AXI_WVALID	1	I	Write valid
	S_AXI_BREADY	1	I	Response ready
	S_AXI_ARADDR	32	I	Read address
	S_AXI_ARVALID	1	I	Read address valid
	S_AXI_RREADY	1	I	Read ready
	S_AXI_ARREADY	1	O	Read address ready
	S_AXI_RDATA	32	O	Read data
	S_AXI_RRESP	2	O	Read response

Comment	Port name	Width	Dir	Description
	S_AXI_RVALID	1	O	Read valid
	S_AXI_WREADY	1	O	Write ready
	S_AXI_BRESP	2	O	Write response
	S_AXI_BVALID	1	O	Write response valid
	S_AXI_AWREADY	1	O	Write address ready
RX Left Eye I/F	LRx_PAER_Addr	18	I	Parallel AER address
	LRx_PAER_Req	1	I	Parallel AER request
	LRx_PAER_Ack	1	O	Parallel AER acknowledge
	LRx_HSSAER	4	I	4 channels High Speed Serial AER signal
	LRx_data_2of7_from_spinnaker	7	I	SpiNNlink input data line
	LRx_ack_to_spinnaker_o	1	O	SpiNNlink acknowledge
RX Right Eye I/F	RRx_PAER_Addr	18	I	Parallel AER address
	RRx_PAER_Req	1	I	Parallel AER request
	RRx_PAER_Ack	1	O	Parallel AER acknowledge
	RRx_HSSAER	4	I	4 channels High Speed Serial AER signal
	RRx_data_2of7_from_spinnaker	7	I	SpiNNlink input data line
	RRx_ack_to_spinnaker_o	1	O	SpiNNlink acknowledge
RX Auxiliary I/F	AuxRx_PAER_Addr	18	I	Parallel AER address
	AuxRx_PAER_Req	1	I	Parallel AER request
	AuxRx_PAER_Ack	1	O	Parallel AER acknowledge
	AuxRx_HSSAER	4	I	4 channels High Speed Serial AER signal
	AuxRx_data_2of7_from_spinnaker	7	I	SpiNNlink input data line
	AuxRx_ack_to_spinnaker_o	1	O	SpiNNlink acknowledge
Slave Axi stream I/f	S_AXIS_TREADY	1	O	Tready
	S_AXIS_TDATA	32	I	Data bus
	S_AXIS_TLAST	1	I	Last signal
	S_AXIS_TVALID	1	I	Valid signal
Master Axi stream I/f	M_AXIS_TREADY	1	I	Tready
	M_AXIS_TDATA	32	O	Data bus
	M_AXIS_TLAST	1	O	Last signal
	M_AXIS_TVALID	1	O	Valid signal
TX I/F	Tx_PAER_Addr	18	O	Parallel AER address
	AuxRx_PAER_Req	1	O	Parallel AER request
	AuxRx_PAER_Ack	1	I	Parallel AER acknowledge
	AuxRx_HSSAER	3	O	3 channels High Speed Serial AER signal
System signals	HSSAER_ClkLS_p, HSSAER_ClkLS_n	2	I	Differential Low Speed clock (usually 100MHz)
	HSSAER_ClkHS_p, HSSAER_ClkHS_n	2	I	Differential High Speed clock. It must be 3x ClkLS (usually 300MHz)
	nSyncReset	1	I	Active low Synchronous reset
	DefLocFarLpbk	1	I	Default Far loopback value
	DefLocNearLpbk	1	I	Default Near loopback value

Table 2.1 - HPUCore interface signals description

3 HPUCore Block Diagram



3.1 Understanding the HPU Core in SynthTactAER and UZCB Application

The HPU Core can have until 3 different AER generator connected to him.

In the HPU Core implementation for SynthTactAER design the AER sources are configured as Serial AER lines. The Serial AER lines are LVDS signals and each source has 3 different channels. The 3 interfaces are:

- Left ATIS camera
- Right ATIS camera
- AUXiliary interface

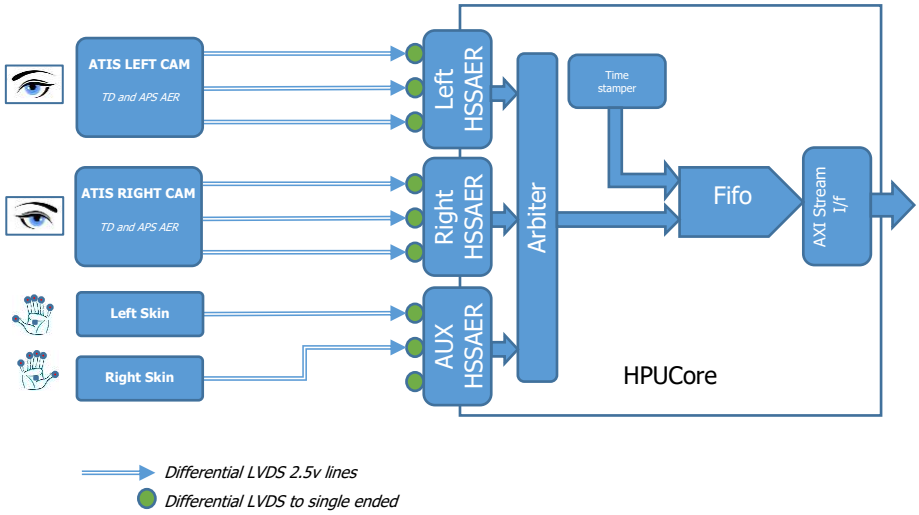


Figure 3.2 - Simplified block diagram of the HPUCore - SynthTactAER.

In the HPU Core implementation for UZCB design the AER sources are configured as GTP or Serial AER lines. The GTP are Gigabit Transceiver (differential lines) and the Serial AER lines (HSSAER) are LVDS signals. The 3 interfaces are:

- Left ATIS GEN3 camera
- Right ATIS GEN3 camera
- AUXiliary interface

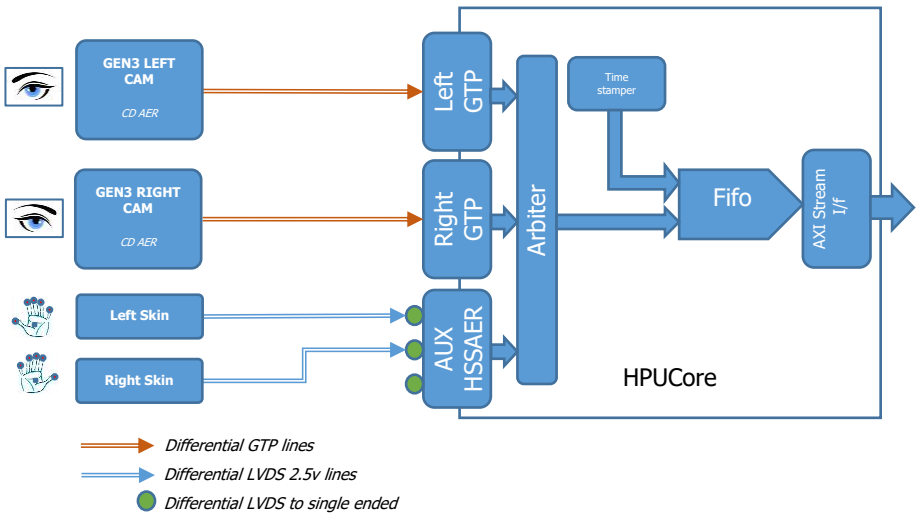


Figure 3.3 - Simplified block diagram of the HPUCore - UZCB

TIMESTAMP (31 downto 0) when CTRLReg.15 is '0'

TIME ID (Reserved) *31 downto 24*	Payload *23 downto 0*
10000000	Time value

TIMESTAMP (31 downto 0) when CTRLReg.15 is '1'

Payload *31 downto 0*
Time value

Note: One-unit difference into Time value means 80ns.

According to the AERsensorsMap.xlsx (svn version r12867) the data are packed as in the tables that follow:

DATA (31 downto 0) coming from Left/Right eyes channels

31	30	29	28	27	26	25	24	Payload *23 downto 0*
0	Reserved				0	0	0	Data

DATA (31 downto 0) coming from Aux channel

31	30	29	28	27	26	25	24	Payload *23 downto 0*
0	Reserved				Copy of data[21:19]			Data

The AXI stream I/f is connected to a DMA used to perform slave to memory transfers.

Reading from `/dev/iit-hpu` we obtain couples of 32bit wide data. The first data is a timestamp while the second one is the data associated to the timestamp.

The **Table 3.1** shows the meaning of both timestamp and data values.

Then a typical acquired sequence is as in the table below:

Example of acquired sequence	Events Notes: *All events in the table come from Left Eye through SAER interface*
T: 0x80FFFD1D --> TD: 0x040132E5	The payload for TD event is 0x132E5
T: 0x80FFFD9E --> TD: 0x040132E6	The payload for TD event is 0x132E6
T: 0x80FFFE1F --> TD: 0x040132E7	The payload for TD event is 0x132E7
T: 0x80FFFEA0 --> TD: 0x040132E8	The payload for TD event is 0x132E8
T: 0x80FFFEB8 --> APS: 0x0405C600	This is a APS event because of the 18th bit is high. The payload for APS event is 0x1C600
T: 0x80FFFF21 --> TD: 0x040132E9	The payload for TD event is 0x132E9
T: 0x80FFFFA2 --> TD: 0x040132EA	The payload for TD event is 0x132EA
T: 0x80000023 --> TD: 0x040132EB	Here the time stamp has wrapped incrementing the wrap value. The payload for TD event is 0x132EB

Table 3.1 - Meaning of Timestamp and Data values.

4 HPUCore internal registers

In this Section a detailed view of the registers internal to the HPUCore module is given.

The HPUCore block has an Axi Light Slave interface [1] to interface the registers with the hosting processor.

AXI is part of ARM AMBA, a family of micro controller buses first introduced in 1996. The first version of AXI was first included in AMBA 3.0, released in 2003. AMBA 4.0, released in 2010, includes the second version of AXI, AXI4. There are three types of AXI4 interfaces:

- AXI4—for high-performance memory-mapped requirements.
- AXI4-Lite—for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- AXI4-Stream—for high-speed streaming data.

Xilinx introduced these interfaces in the ISE® Design Suite, release 12.3.

In the following the complete list of accessible HPUCore registers.

#	Offset	Mnemonic	Description	Type	Reset Value
0	0x00	CTRL_REG	Control register	R/W	0x00000000
1	0x04	LPBK_LR_CNFG_REG	Loopback LR Configuration register	R/W	0x00000000
2	0x08	RXData_REG	RX Data Buffer	R/O	0x00000000
3	0x0C	RXTime_REG	RX Time Buffer	R/O	0x00000000
4	0x10	TXData_REG	TX Data Buffer	R/W	0x00000000
5	0x14	DMA_BREG	DMA Burst Register	R/W	0x00000000
6	0x18	STAT_RAW_REG	Status RAW register	R/O	0x00000000
7	0x1C	IRQ_REG	IRQ register	R/C	0x00000000
8	0x20	MSK_REG	Mask register for the IRQ_REG register	R/W	0x00000000
10	0x28	WRAPTimeStamp_REG	Wrapping TimeStamp Register	R/C	0x00000000
13	0x34	HSSAER_STAT	HSSAER status register	R/O	0x00000000
14	0x38	HSSAER_RX_ERR	HSSAER RX Error register	R/O	0x00000000
15	0x3C	HSSAER_RX_MSK	HSSAER RX Mask register	R/W	0x00000000
16	0x40	RX_CTRL_REG	RX Control register	R/W	0x00000000
17	0x44	TX_CTRL_REG	TX Control register	R/W	0x00000000
18	0x48	RX_PAER_CNFG_REG	RX PAER configuration register	R/W	0x01010100
19	0x4C	TX_PAER_CFNG_REG	TX PAER Configuration register	R/W	0x00000000
20	0x50	IP_CFNG_REG	IP implemented configuration register	R/O	0x0000????
21	0x54	FIFO_THRS_REG	FIFO threshold value register	R/W	0x00000000
22	0x58	LPBK_AUX_CNFG_REG	Loopback AUX Configuration register	R/W	0x00000000
23	0x5C	ID_REG	ID Register	R/O	0x48505535
24	0x60	AUX_CTRL_REG	Auxiliary interface Control register	R/W	0x00000000
25	0x64	HSSAER_AUX_RX_ERR	HSSAER AUX RX Error register	R/O	0x00000000
26	0x68	HSSAER_AUX_RX_MSK	HSSAER AUX RX Mask register	R/W	0x00000000

#	Offset	Mnemonic	Description	Type	Reset Value
27	06C	HSSAER_AUX_RX_ERR_THR_REG	HSSAER AUX RX error threshold register	R/W	0x10101010
28	0x70	HSSAER_AUX_RX_ERR_CH0_REG	HSSAER AUX RX error counter register for Channel 0	R/C	0x00000000
29	0x74	HSSAER_AUX_RX_ERR_CH1_REG	HSSAER AUX RX error counter register for Channel 1	R/C	0x00000000
30	0x78	HSSAER_AUX_RX_ERR_CH2_REG	HSSAER AUX RX error counter register for Channel 2	R/C	0x00000000
31	0x7C	HSSAER_AUX_RX_ERR_CH3_REG	HSSAER AUX RX error counter register for Channel 3	R/C	0x00000000
32	0x80	SPNN_START_KEY_REG	SpiNNlink Start command key	R/W	0x80000000
33	0x84	SPNN_STOP_KEY_REG	SpiNNlink Stop command key	R/W	0x40000000
34	0x88	SPNN_TX_MASK_REG	SpiNNlink TX Data Mask	R/W	0x00FFFFFF
35	0x8C	SPNN_RX_MASK_REG	SpiNNlink Stop command key	R/W	0x00FFFFFF
36	0x90	SPNN_CTRL_REG	SpiNNaker Control Register	R/W	0x00000000
37	0x94	SPNN_STATUS_REG	SpiNNaker Status Register	R/O	0x00000000
38	0xA0	TlastTimeOut_REG	TLAST Time Out register	R/W	0x00010000
39	0xA4	TlastConter_REG	TLAST Counter register	R/O	0x00000000
40	0xA8	TdataCounter_REG	TData Counter register	R/O	0x00000000
41	0xAC	Reserved			
42	0xB0	GTRX_LEFT_REG	Gigabit Transceiver receiver - left	R/W	
43	0xB4	GTRX_RIGHT_REG	Gigabit Transceiver receiver - right	R/W	
44	0xB8	GTRX_AUX_REG	Gigabit Transceiver receiver - aux	R/W	
45	0xBC	GTTX_REG	Gigabit Transceiver transmitter	R/W	

4.1 Control register (CTRL_REG)

This register is used to control the behaviour of the HPUCore block.

CTRL_REG (HPUCore Base + 0x00) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LocFar RPAER Lpbk	LocFar LPAER Lpbk	LocFar RSAER Lpbk	LocFar LSAER Lpbk	LocFar AuxPA ERLpb k	LocFar AuxSA ERLpb k	Loc Near Lpbk	Remot e Lpbk	LocFar SpinnL pbkSel (1)	LocFar SpinnL pbkSel (0)	Reserved					
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Full Time stam p	Only Events TX	Only Events RX	Reset Stream	Reserved		AXIStre am Latency	FlushTX FIFO	AuxRxPa er FIFO FLush	RRxPaer FIFO FLush	LRxPaer FIFO FLush	FlushRX FIFO	Rese rved	EN INT	EN DMA	DMA running
r/w							s/c	s/c	s/c	s/c	s/c		r/w	r/w	ro

- DMA running
  - When '1' it shows that the DMA transfer is on going
  - When '0' it shows that no DMA transfer is active
- EN DMA is the DMA interface Enable
  - When '1' the DMA I/f is enabled
  - When '0' the DMA I/f is disabled
- Enable Interrupt
  - When '1' the Interrupt is enabled
  - When '0' the Interrupt never rises up
- FlushRXFIFO
  - When set to '1' the RX FIFO of the HPUCore is flushed. This bit is automatically cleared.
- LRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the Left PAER interface are flushed. This bit is automatically cleared.
- RRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the Right PAER interface are flushed. This bit is automatically cleared.
- AuxRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the AUX PAER interface are flushed. This bit is automatically cleared.
- FlushTXFIFO
  - When set to '1' the TX FIFO of the HPUCore is flushed. This bit is automatically cleared.
- AXIStream Latency
  - When set to 1 the HPU based on the TLAST TIMEOUT Register (TLASTTO_REG) can have a maximum latency equal to the value of the register multiplied by the period of the system clock
- Only Events RX
  - When set to 1, the timestamp of received events is NOT sent to DMA. That permits a doubling of events throughput
- Only Events TX
  - When set to 1, the timestamp of transmitting events is NOT present in DMA data, and events are transmitted as soon as possible. That permits a doubling of events throughput
- Fulltimestamp
  - When set to '1' the Timestamp is 32 bit wide, when set to '0' the time stamp is 24 bit wide and the higher part is equal to 0x80.
- Local Far SpinnLink Loopback selection (for further details, look at the RTL code):
  - When '00' No Loopback
  - When '01' Tx is sent to "LEFT" Rx
  - When '10' Tx is sent to "RIGHT" Rx
  - When '11' Tx is sent to "AUX" Rx
- Remote Loopback enabling (for further details, look at the RTL code)

- Local Near Loopback enabling (for further details, look at the RTL code)
- Local Far PAER/SAER Loopback enabling (for further details, look at the RTL code)

4.2 Loopback LR Configuration register (LPBK_LR_CNFG_REG)

This register contains the configuration for Left and Right loopback.

LPBK_LR_CNFG_REG (HPUCore Base + 0x04) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Right RX chan 3 LPB cnfg				Right RX chan 2 LPB cnfg				Right RX chan 1 LPB cnfg				Right RX chan 0 LPB cnfg			
r/w				r/w				r/w				r/w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left RX chan 3 LPB cnfg				Left RX chan 2 LPB cnfg				Left RX chan 1 LPB cnfg				Left RX chan 0 LPB cnfg			
r/w				r/w				r/w				r/w			

The register is used in debug to test the connection. For further details, look at the RTL code.

4.3 RX Data Buffer register (RXDATA_REG)

This register contains the data (read from the INFIFO) coming from the selected by N_MuxAddr NMC. The format of the register is depicted into the figure below.

RXDATA_REG (HPUCore Base + 0x08) Reset Value: **0x00000000**

The meaning of this register is as explained in **Table 3.1**.

NOTE: The reading of this register must follow the reading of the RX Time Buffer register (RXTIME_REG).

4.4 RX Time Buffer register (RXTIME_REG)

This register contains the time stamp associated to the received data (see Loopback LR Configuration register (LPBK_LR_CNFG_REG)

This register contains the configuration for Left and Right loopback.

LPBK_LR_CNFG_REG (HPUCore Base + 0x04) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Right RX chan 3 LPB cnfg				Right RX chan 2 LPB cnfg				Right RX chan 1 LPB cnfg				Right RX chan 0 LPB cnfg			
r/w				r/w				r/w				r/w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left RX chan 3 LPB cnfg				Left RX chan 2 LPB cnfg				Left RX chan 1 LPB cnfg				Left RX chan 0 LPB cnfg			
r/w				r/w				r/w				r/w			

The register is used in debug to test the connection. For further details, look at the RTL code.

RX Data Buffer register (RXDATA_REG)) from the INFIFO.

RXTIME_REG (HPUCore Base + 0x0C) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
10000000								Time stamp							
ro															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Time stamp															
ro															

The Time Stamp value read from this register is the Time Stamp that the HPUCore sticks to the Received data available into the RX Data Buffer register (RXDATA_REG).

4.5 TX Data Buffer register (TXDATA_REG)

This register is used to fill the OUTFIFO.

TXDATA_REG (HPUCore Base + 0x10) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXData															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXData															
rw															

When writing to this register, keep in mind that it is used by the internal hw as follows:

The register needs to be written twice to enable the correct behaviour.

The first data written into the register represents the time, elapsed which, the second data written into the register is delivered to the *loopback* module, routing it according to its MSB bit:

00 => the packet is sent to the parallel AER interface
01 => the packet is sent to the HSSAER interface
10 => the packet is sent to the SpiNNlink interface ---------- it was: GTP driver interface
11 => the packet is sent to all the interfaces: it is acknowledged only when all the interfaces have acknowledged the transfer

4.6 DMA register (DMA_REG)

This register is used set the behaviour of the Axistream interface. It represents the number of data (32 bit size length) sent to the DMA interface and "closed" by a TLAST on the axistream interface.

Please note that this value has bit 0 set to 0, this means that it cannot be written with an odd value.

NOTE: This register can be written only if CTRL_REG.ENDMA='0'.

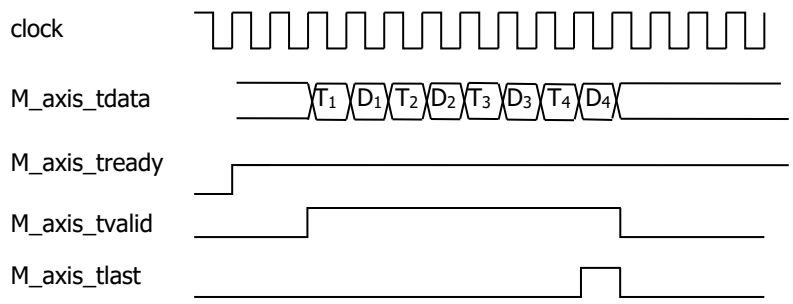
DMA_REG (HPUCore Base + 0x14) Reset Value: **0x00000100**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															DMA_test mode
															r/w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA Length[15:1]															0
r/w															

The DM_test_mode set to '1' enables the DMA to write consecutive incremental values at high rates.

For example, if it is set to 8, the burst from/to the DMA I/f will be in terms of 8 data length.



4.7 RAW Status Register (STAT_RAW_REG)

When read, this register gives a snapshot of the status of warning or errors signals. It is a Read Only register.

STAT_RAW_REG (HPUCore Base + 0x18) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			GT disaligned	GT PLL alarm	AuxSpinnRxErr	RSpinnRxErr	LSpinnRxErr	AuxSpinnParityErr	RSpinnParityErr	LSpinnParityErr	TxSpinnDump	GlblRXerr_of	GlblRXerr_to	GlblRXerr_tx	GlblRXerr_ko
					ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFO > THRS	AUX RX PAER FIFO FULL	RRX PAER FIFO FULL	LRX PAER FIFO FULL	Reserved		RX FIFO Not Empty	RxBufferReady	Time Stamp Wrapped	Reserved	TX Data Full	TX Data Almost Full	TX Data Empty	RX Data Full	RX Data Almost Empty	RX Data Empty
	ro	ro	ro			ro	ro	ro		ro	ro	ro	ro	ro	ro

- RxDataEmpty
  - When '0', the INFIFO is not empty
  - When '1' the INFIFO is empty
- RxDataAlmostEmpty
  - When '1' the INFIFO has 1 or 0 data to be read.
  - When '0' the INFIFO has more or equal two data to be read.
- RxDataFull
  - When '1' the INFIFO is full.
  - When '0' the INFIFO is not full.
- TxDataEmpty
  - When '0', the OUTFIFO is not empty
  - When '1' the OUTFIFO is empty
- TxDataAlmostFull
  - When '1' the OUTFIFO has 2047 or 2048 data within himself.
  - When '0' the OUTFIFO has less than 2047 data within himself.
- TxDataFull
  - When '1' the OUTFIFO is full.
  - When '0' the OUTFIFO is not full.
- Bias Finished
  - When '1' the Bias signals have been latched
  - When '0' no Bias signals have been latched
- Time stamp wrapped (this bit is high for one clock period only, when the counter wraps its value)
  - When '1' the counter inside the TimeStamp module has wrapped its value.
  - When '0' the counter inside the TimeStamp module has not yet wrapped its value
- RXBufferReady
  - When '1' the Rx Fifo has at least DMA_REG value of data available
  - When '0' the Rx Fifo has less than DMA_REG value of data available
- RXFifoNotEmpty
  - When '1' the RX Fifo is not empty.
  - When '0' the RX Fifo is empty
- LRXPaerFifoFull
  - When '1' the Left RX Fifo is not empty.
  - When '0' the Left RX Fifo is empty
- RRXPaerFifoFull

  - When '1' the Right RX Fifo is not empty.
  - When '0' the Right RX Fifo is empty
- AuxRXPaerFifoFull
  - When '1' the Aux RX Fifo is not empty.
  - When '0' the Aux RX Fifo is empty
- RXfifo>Threshold
  - When '1' the Infifo has more elements with respect the value written into Fifo Threshold register (FIFO_THRSH_REG).
  - When '0' the Infifo has lesse elements with respect the value written into Fifo Threshold register (FIFO_THRSH_REG.
- Glbl Rx err ko
  - Global Rx err ko. (see [3] for further details)

    It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER_AUX_RX_ERR_THR_REG).

- Glbl Rx err rx
  - Global Rx err rx. (see [3] for further details)

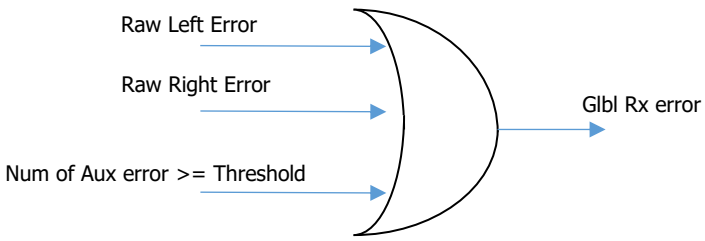
    It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER_AUX_RX_ERR_THR_REG).

- Glbl Rx err to
  - Global Rx err to. (see [3] for further details)

    It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER_AUX_RX_ERR_THR_REG).

- Glbl Rx err of
  - Global Rx err of. (see [3] for further details)

    It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER_AUX_RX_ERR_THR_REG).

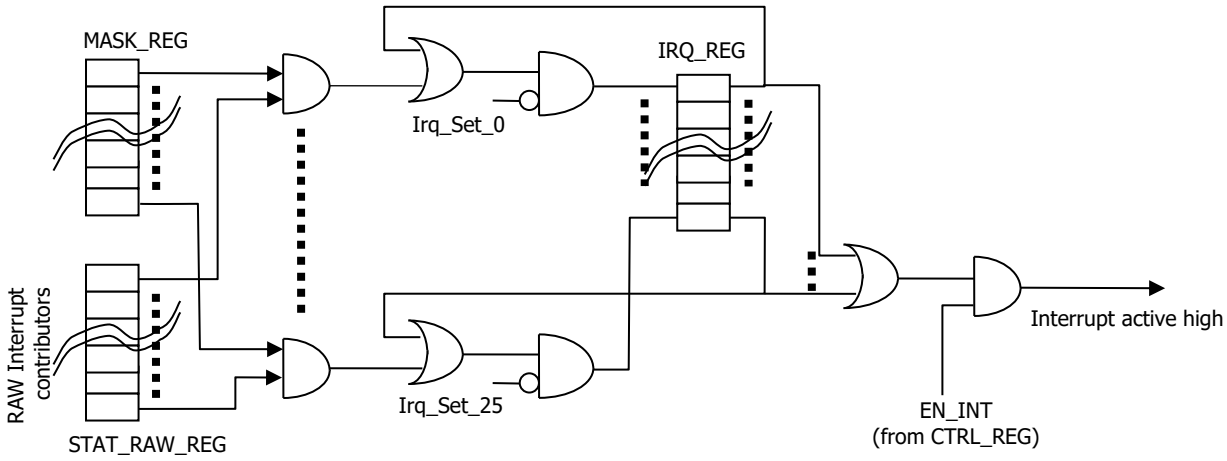


- TxSpinnDump
  - When '0', the SpiNNlink TX is working
  - When '1', the SpiNNlink TX is "dumping" data
- LSpinnRxErr
  - When '0', the Left SpiNNlink RX is working
  - When '1', the Left SpiNNlink RX is receiving wrong symbols
- RSpinnRxErr
  - When '0', the Right SpiNNlink RX is working
  - When '1', the Right SpiNNlink RX is receiving wrong symbols
- AuxSpinnRxErr
  - When '0', the Aux SpiNNlink RX is working
  - When '1', the Aux SpiNNlink RX is receiving wrong symbols
- LSpinnParityErr
  - When '0', the Left SpiNNlink RX is working
  - When '1', the Left SpiNNlink RX is receiving packets with wrong parity

- RSpinnParityErr
  - When '0', the Right SpiNNlink RX is working
  - When '1', the Right SpiNNlink RX is receiving packets with wrong parity
- AuxSpinnParityErr
  - When '0', the Aux SpiNNlink RX is working
  - When '1', the Aux SpiNNlink RX is receiving packets with wrong parity

4.8 IRQ Register (IRQ_REG)

When read, this register gives the status of the collected warning or errors signals. It is a Read/Set register, i.e., to clear the warning/error bit the user has to write '1' on the corresponding bit position.



IRQ_REG (HPUCore Base + 0x1C) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			GT disaligned	GT PLL alarm	AuxSpinn RxErr	RSpinnRx Err	LSpinnRx Err	AuxSpinn ParityErr	RSpinnParityErr	LSpinnParityErr	TxSpinnDump	Glbl RX err_of	Glbl RX err_to	Glbl RX err_rx	Glbl RX err_ko
			r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c	r/c

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFO > THRS	AUX RX PAER FIFO FULL	RRX PAER FIFO FULL	LRX PAER FIFO FULL	Reserved		RX FIFO Not Empty	RxBufferReady	Time Stamp Wrapped	Reserved	TX Data Full	TX Data Almost Full	TX Data Empty	RX Data Full	RX Data Almost Empty	RX Data Empty
r/c	r/c	r/c	r/c			r/c	r/c	r/c		r/c	r/c	r/c	r/c	r/c	r/c

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT_RAW_REG).

4.9 Mask Register (MSK_REG)

This is the Mask register used to mask the contributors for the interrupt signal.

MSK_REG (HPUCore Base + 0x20) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												GlblRXerr_of	GlblRXerr_to	GlblRXerr_tx	GlblRXerr_ko
												r/w	r/w	r/w	r/w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFO > THRS	AUX RX PAER FIFO FULL	RRX PAER FIFO FULL	LRX PAER FIFO FULL	Reserved		RX FIFO Not Empty	RxBufferReady	Time Stamp Wrapped	Reserved	TX Data Full	TX Data Almost Full	TX Data Empty	RX Data Full	RX Data Almost Empty	RX Data Empty
r/w	r/w	r/w	r/w			r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT_RAW_REG).

4.10 Wrapping TimeStamp Register (WRAPTimeStamp_REG)

This register is used to read how many times the internal 32bit counter of the TimeStamp module has wrapped its value.

In case the user writes any value in this register, it will be cleared and also the internal 32bit counter of the TimeStamp module will be cleared.

WRAPTIMESTAMP_ REG (NEUELAB Base + 0x28) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Wrapping times															
r/c															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wrapping times															
r/c															

4.11 HSSAER STATus register (HSSAER_STAT_REG)

This is the HSSAER Status register.

HSSAER_STAT_REG (HPUCore Base + 0x34) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				Chan3 Aux run	Chan2 Aux run	Chan1 Aux run	Chan0 Aux run	Reserved				Chan3 TX run	Chan2 TX run	Chan1 TX run	Chan0 TX run
				ro	ro	ro	ro					ro	ro	ro	Ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				Chan3 R RX run	Chan2 R RX run	Chan1 R RX run	Chan0 R RX run	Reserved				Chan3 L RX run	Chan2 L RX run	Chan1 L RX run	Chan0 L RX run
				ro	ro	ro	ro					ro	ro	ro	ro

The user can read the status of the 4 channels of Left Rx Eye, Right Rx Eye, Aux Rx or Tx channel.

4.12 HSSAER RX Error register (HSSAER_RX_ERR_REG)

This is the HSSAER Rx error register.

HSSAER_RX_ERR_REG (HPUCore Base + 0x38) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chan3 R RX err_of	Chan3 R RX err_to	Chan3 R RX err_rx	Chan3 R RX err_ko	Chan2 R RX err_of	Chan2 R RX err_to	Chan2 R RX err_rx	Chan2 R RX err_ko	Chan1 R RX err_of	Chan1 R RX err_to	Chan1 R RX err_rx	Chan1 R RX err_ko	Chan0 R RX err_of	Chan0 R RX err_to	Chan0 R RX err_rx	Chan0 R RX err_ko
ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	Ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chan3 L RX err_of	Chan3 L RX err_to	Chan3 L RX err_rx	Chan3 L RX err_ko	Chan2 L RX err_of	Chan2 L RX err_to	Chan2 L RX err_rx	Chan2 L RX err_ko	Chan1 L RX err_of	Chan1 L RX err_to	Chan1 L RX err_rx	Chan1 L RX err_ko	Chan0 L RX err_of	Chan0 L RX err_to	Chan0 L RX err_rx	Chan0 L RX err_ko
ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro

The user can read the error contributors for Left and Right 4 channels. See [3].

4.13 HSSAER RX MSK register (HSSAER_RX_MSK_REG)

This is the HSSAER Rx mask register.

HSSAER_RX_MSK_REG (HPUCore Base + 0x3C) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Chan3 R RX err_of	Chan3 R RX err_to	Chan3 R RX err_rx	Chan3 R RX err_ko	Chan2 R RX err_of	Chan2 R RX err_to	Chan2 R RX err_rx	Chan2 R RX err_ko	Chan1 R RX err_of	Chan1 R RX err_to	Chan1 R RX err_rx	Chan1 R RX err_ko	Chan0 R RX err_of	Chan0 R RX err_to	Chan0 R RX err_rx	Chan0 R RX err_ko
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Chan3 L RX err_of	Chan3 L RX err_to	Chan3 L RX err_rx	Chan3 L RX err_ko	Chan2 L RX err_of	Chan2 L RX err_to	Chan2 L RX err_rx	Chan2 L RX err_ko	Chan1 L RX err_of	Chan1 L RX err_to	Chan1 L RX err_rx	Chan1 L RX err_ko	Chan0 L RX err_of	Chan0 L RX err_to	Chan0 L RX err_rx	Chan0 L RX err_ko
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

The user can mask (writing 0) or not (writing 1) the corresponding contributors of error register. See [3].

4.14 RX Control register (RX_CTRL_REG)

This is the HSSAER Left and Right Rx control register.

RX_CTRL_REG (HPUCore Base + 0x40) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				RRX HSSAER Channel En				Reserved				RRX SpNNInkEn	RRX GTP En	RRX PAER En	RRX HSSAER En
				Channel 3	Channel 2	Channel 1	Channel 0								
				rw								rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LRX HSSAER Channel En				Reserved				LRX SpNNInkEn	LRX GTP En	LRX PAER En	LRX HSSAER En
				Channel 3	Channel 2	Channel 1	Channel 0								
				rw								rw	rw	rw	rw

Where:

- LRX HSSAER Enable
  - When '0', the Left HSSAER interface is not enabled
  - When '1', the Left HSSAER interface is enabled
- LRX PAER Enable
  - When '0', the Left PAER interface is not enabled
  - When '1', the Left PAER interface is enabled
- LRX GTP Enable
  - When '0', the Left GTP interface is not enabled
  - When '1', the Left GTP interface is enabled
- LRX SpNNInk Enable
  - When '0', the Left SpiNNlink interface is not enabled
  - When '1', the Left SpiNNlink interface is enabled
- LRX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it
- RRX HSSAER Enable
  - When '0', the Right HSSAER interface is not enabled
  - When '1', the Right HSSAER interface is enabled
- RRX PAER Enable
  - When '0', the Right PAER interface is not enabled
  - When '1', the Right PAER interface is enabled
- RRX GTP Enable
  - When '0', the Right GTP interface is not enabled
  - When '1', the Right GTP interface is enabled
- RRX SpNNInk Enable
  - When '0', the Right SpiNNlink interface is not enabled
  - When '1', the Right SpiNNlink interface is enabled
- RRX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

4.15 TX Control register (TX_CTRL_REG)

This is the HSSAER Tx control register.

TX_CTRL_REG (HPUCore Base + 0x44) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										TX Timestamp Mask Selection		TX Timing resync timeout Selection			
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX timeout counter rearm/status	TX Timing resync retrig/ status	TX Timing Mode		TX HSSAER Channel En					Tx Dest Switch En	Tx Dest Switch (1)	Tx Dest Switch (0)	TX SpNNInkEn	TX GTP En	TX PAER En	TX HSSAER En
				Channel 3	Channel 2	Channel 1	Channel 0								
rw	rw	rw		rw					rw	rw	rw	rw	rw	rw	rw

Where:

- TX Destination Switch Enable
  - When '0' the message is routed according to its two MSB bits:
    - 00 => the packet is sent to the parallel AER interface
    - 01 => the packet is sent to the HSSAER interface
    - 10 => the packet is sent to the SpiNNlink interface ---------- it was: GTP driver interface
    - 11 => the packet is sent to all the interfaces: it is acknowledged
  - When '1' the message is routed according to TxDestSwitch(1:0)

    Note: tied to '0' if IP doesn't have any TX interface
- TX Destination Switch
  - if TX Destination Switch Enable = '1',
    when
    - 00 => the packet is sent to the parallel AER interface
    - 01 => the packet is sent to the HSSAER interface
    - 10 => the packet is sent to the SpiNNlink interface ---------- it was: GTP driver interface
    - 11 => the packet is sent to all the interfaces: it is acknowledged
  - if TX Destination Switch Enable = '0', it doesn't have effect

    Note: tied to '0' if IP doesn't have any TX interface
- TX HSSAER Enable
  - When '0', the HSSAER interface is not enabled
  - When '1', the HSSAER interface is enabled
- TX PAER Enable
  - When '0', the PAER interface is not enabled
  - When '1', the PAER interface is enabled
- TX GTP Enable
  - When '0', the GTP interface is not enabled
  - When '1', the GTP interface is enabled
- TX SpiNNlink Enable
  - When '0', the SpiNNlink interface is not enabled
  - When '1', the SpiNNlink interface is enabled
- TX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

- TX Timing Mode
  - 00 : *Delta Time mode*: timestamp provided with data is the *Delta Time* since the last
  - 01 : *ASAP Mode*: timestam is ignored, and events are sent as-soon-as-possible
  - 10 : *Absolute Time Mode*: timestamp provided with data is "absolute"
  - 11 : Available for future use (if selected now, TX is stopped)
- TX Timing resync Retrig/Status
  - When read, it reports the Status of Resynchronization:
    - 1 = TX Timer is ready for resynchronization with new data;
    - 0 = TX Timer is evolving by its own;
  - Writing "1" force the TX timer to be ready for resynchronization, regardless timeout was or not reached;
  - Writing "0" does not have any effect.
- TX Timeout counter Rearm/Status
  - When read, it reports the Status of Timeout Counter:
    - 1 = Counting (it means that last event was sent and internal TX FIFO is void)
    - 0 = Stopped (it means that TX engine is processing data);
  - Writing "1" force a rearming of Timeout Counter (like an event was sent);
  - Writing "0" does not have any effect.
- TX Timing resync timeout selection
  - It permits to select the timeout value for a resynchronisation or disable the timeout counter
    - 0 = 1 ms
    - 1 = 5 ms
    - 2 =10 ms
    - 3 = 50 ms
    - 4 = 100 ms
    - 5 = 500 ms
    - 6 = 1 s
    - 7 = 2.5 s
    - 8 = 5 s
    - 9 = 10 s
    - A = 25 s
    - B = 50 s
    - C = 100 s (1m 40s)
    - D = 250 s (4m 10s)
    - E = 500 s (8m 20)
    - F = **Timer disabled**
- TX Timestamp Mask Selection
  - It permits to select a Mask for TX internal Timestamp (be sure that it is big enough to contain the maximum delay between events, and lower than the mask of Data Timestamp)
    - 0 = 0x000FFFFF (20 bits)
    - 1 = 0x00FFFFFF (24 bits)
    - 2 = 0x0FFFFFFF (28 bits)
    - 3 = 0xFFFFFFFF (32 bits)

4.16 RX PAER Configuration register (RX_PAER_CFNG_REG)

This is the RX PAER configuration register.

RX_PAER_CFNG_REG (HPUCore Base + 0x48) Reset Value: **0x02000100**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXPAER Ack release Delay								RXPAER Ack Set Delay							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXPAER Data Sample Delay								Reserved			RXPAER ignore Fifo Full	Reserved		RXPAER Ack Active Lvl	RXPAER Req Active Lvl
rw											rw			rw	rw

Where:

- RXPAER Req Active level
  - When '0', the Request signal is active low
  - When '1', the Request signal is active high
- RXPAER Ack Active level
  - When '0', the Acknowledge signal is active low
  - When '1', the Acknowledge signal is active high
- RX PAER ignore FIfo Full
  - When '0', the Fifo Full stops the acknowledge signal
  - When '1', the Fifo Full doesn't stop the acknowledge signal by acknowledging the request
- RXPAER Data Sample Delay
  - This is the number of system clock used to sample the PAER address.
- RXPAER Ack Set Delay
  - This is the number of system clock used to set the ACK signal after that the request becomes active.
- RXPAER Ack Release Delay
  - This is the number of system clock used to release the ACK signal after that the request becomes active.

Please note that *RXPAER Data Sample Delay*, *RXPAER Ack Set Delay* and *RXPAER Ack Release Delay* must be different in values.

Please note that this register must be written before enabling the interface in RX Control register (RX_CTRL_REG) and/or AUXiliary RX Control register (AUX_RX_CTRL_REG)

4.17 TX PAER Configuration register (TX_PAER_CFNG_REG)

This is the TX PAER configuration register.

TX_PAER_CFNG_REG (HPUCore Base + 0x4C) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TXPAER Ack Active Lvl	TXPAER Req Active Lvl
														rw	rw

Where:

- TXPAER Req Active level
  - When '0', the Request signal is active low
  - When '1', the Request signal is active high
- TXPAER Ack Active level
  - When '0', the Acknowledge signal is active low
  - When '1', the Acknowledge signal is active high

Please note that this register must be set before to enabling the PAER interface (i.e.: before writing the TX Control register (TX_CTRL_REG)

4.18 IP Configuration register (IP_CFNG_REG)

This is the HPUCore configuration register. It shows how the HPUCore has been implemented in terms of features.

IP_CFNG_REG (HPUCore Base + 0x50) Reset Value: **0x0000????**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TX HSSAER #channels -1		HAS TX SpNNlink	HAS TX GTP	Has TX PAER	Has TX HSSAER	Reserved		RX HSSAER #channels -1		HAS RX SpNNlink	HAS RX GTP	Has RX PAER	Has RX HSSAER
													ro	ro	ro

Where:

- Has RX HSSAER
  - When '0', the IP doesn't have the RX HSSAER I/f
  - When '0', the IP has the RX HSSAER I/f
- Has RX PAER
  - When '0', the IP doesn't have the RX PAER I/f
  - When '0', the IP has the RX PAER I/f
- Has RX GTP
  - When '0', the IP doesn't have the RX GTP I/f
  - When '0', the IP has the RX GTP I/f
- Has RX SpiNNlink
  - When '0', the IP doesn't have the RX SpiNNlink I/f
  - When '0', the IP has the RX SpiNNlink I/f
- RX HSSAER #channels -1
  - This shows the number of channels of RX HSSAER I/f. For instance, if it is 2'b10, it means that the RX HSSAER interface has 3 channels
- Has TX HSSAER
  - When '0', the IP doesn't have the TX HSSAER I/f
  - When '0', the IP has the TX HSSAER I/f
- Has TX PAER
  - When '0', the IP doesn't have the TX PAER I/f
  - When '0', the IP has the TX PAER I/f
- Has TX GTP
  - When '0', the IP doesn't have the TX GTP I/f
  - When '0', the IP has the TX GTP I/f
- Has TX SpiNNlink
  - When '0', the IP doesn't have the TX SpiNNlink I/f
  - When '0', the IP has the TX SpiNNlink I/f
- RX HSSAER #channels -1
  - This shows the number of channels of RX HSSAER I/f. For instance, if it is 2'b10, it means that the RX HSSAER interface has 3 channels

4.19 Fifo Threshold register (FIFO_THRSH_REG)

This register contains the number of elements of the INFIFO after which the *"RXFIFO > THRS"* of the IRQ Register (IRQ_REG) bit goes high.

FIFO_THRSH_REG (HPUCore Base + 0x54) Reset Value: **0x00000000**

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
					Threshold value										
					r/w										

4.20 LoopBack AUX Configuration (LPBK_AUX_CNFG_REG)

This register contains the configuration for the AUX interface loopback.

LPBK_AUX_CNFG_REG (HPUCore Base + 0x58) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUX RX chan 3 LPB cnfg				AUX RX chan 2 LPB cnfg				AUX RX chan 1 LPB cnfg				AUX RX chan 0 LPB cnfg			
r/w				r/w				r/w				r/w			

The register is used in debug to test the connection. For further details, look at the RTL code.

4.21 Identification register (ID_REG)

This register contains the ID of the NeuElab.

ID_REG (HPUCore Base + 0x5C) Reset Value: **48505535**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H								P							
r/o								r/o							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U								Major				Minor			
r/o								r/o				r/o			

Minor = 6;

Major = 3;

4.22 AUXiliary RX Control register (AUX_RX_CTRL_REG)

This is the Auxiliary Rx control register.

AUX_RX_CTRL_REG (HPUCore Base + 0x60) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				AUX HSSAER Channel En				Reserved				AuxRX SpNNInkEn	AUX GTP En	AUX PAER En	AUX HSSAER En
				Channel 3	Channel 2	Channel 1	Channel 0								
				rw									rw	rw	rw

Where:

- AUX HSSAER Enable
  - When '0', the AUX HSSAER interface is not enabled
  - When '1', the AUX HSSAER interface is enabled
- AUX PAER Enable
  - When '0', the AUX PAER interface is not enabled
  - When '1', the AUX PAER interface is enabled
- AUX GTP Enable
  - When '0', the AUX GTP interface is not enabled
  - When '1', the AUX GTP interface is enabled
- AUX SpiNNlink Enable
  - When '0', the AUX SpiNNlink interface is not enabled
  - When '1', the AUX SpiNNlink interface is enabled
- AUX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

4.23 HSSAER AUX RX Error register (HSSAER_AUX_RX_ERR_REG)

This is the HSSAER Rx error register.

HSSAER_AUX_RX_ERR_REG (HPUCore Base + 0x64) Reset Value: **0x00000000**

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
Chan3 AUX RX err_of	Chan3 AUX RX err_to	Chan3 AUX RX err_rx	Chan3 AUX RX err_ko	Chan2 AUX RX err_of	Chan2 AUX RX err_to	Chan2 AUX RX err_rx	Chan2 AUX RX err_ko	Chan1 AUX RX err_of	Chan1 AUX RX err_to	Chan1 AUX RX err_rx	Chan1 AUX RX err_ko	Chan0 AUX RX err_of	Chan0 AUX RX err_to	Chan0 AUX RX err_rx	Chan0 AUX RX err_ko
ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro

The user can read the error contributors for the 4 channels of the AUX interface. See [3].

4.24 HSSAER AUX RX MSK register (HSSAER_AUX_RX_MSK_REG)

This is the HSSAER AUX Rx mask register.

HSSAER_AUX_RX_MSK_REG (HPUCore Base + 0x68) Reset Value: **0x00000000**

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
Chan3 AUX RX err_of	Chan3 AUX RX err_to	Chan3 AUX RX err_rx	Chan3 AUX RX err_ko	Chan2 AUX RX err_of	Chan2 AUX RX err_to	Chan2 AUX RX err_rx	Chan2 AUX RX err_ko	Chan1 AUX RX err_of	Chan1 AUX RX err_to	Chan1 AUX RX err_rx	Chan1 AUX RX err_ko	Chan0 AUX RX err_of	Chan0 AUX RX err_to	Chan0 AUX RX err_rx	Chan0 AUX RX err_ko
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

The user can mask (writing 0) or not (writing 1) the corresponding contributors of error register. See [3].

4.25 Aux Error counter threshold register (HSSAER_AUX_RX_ERR_THR_REG)

This is register is used for setting the threshold of the AUX Rx counter error.

As soon as the number of the corresponding error overcome the threshold here set, the interrupt related will be raised if opportunely masked.

HSSAER_AUX_RX_ERR_THR_REG (HPUCore Base + 0x6C) Reset Value: **0x10101010**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of AUX of error								Number of AUX to error							
r/w								r/w							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of AUX rx error								Number of AUX ko errors							
r/w								r/w							

- Number of AUX ko errors
  - The number of ko errors which, if overcome, can raise an interrupt
- Number of AUX rx errors
  - The number of rx errors which, if overcome, can raise an interrupt
- Number of AUX to errors
  - The number of to errors which, if overcome, can raise an interrupt
- Number of AUX of errors
  - The number of of errors which, if overcome, can raise an interrupt

4.26 Aux Error counter CH0 register (HSSAER_AUX_RX_ERR_CH0_REG)

This is register is used to read the number of errors occurred in HSSAER AUX channel 0 lines.

Reading this register we also clear it.

HSSAER_AUX_RX_ERR_CH0_REG (HPUCore Base + 0x70) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of CH0 of error								Number of CH0 to error							
r/c								r/c							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of CH0 rx error								Number of CH0 ko errors							
r/c								r/c							

- Number of CH0 ko errors
  - The number of ko errors
- Number of CH0 rx errors
  - The number of rx errors
- Number of CH0 to errors
  - The number of to errors
- Number of CH0 of errors
  - The number of of errors

4.27 Aux Error counter CH1 register (HSSAER_AUX_RX_ERR_CH1_REG)

This is register is used to read the number of errors occurred in HSSAER AUX channel 1 lines.

Reading this register we also clear it.

HSSAER_AUX_RX_ERR_CH1_REG (HPUCore Base + 0x74) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of CH1 of error								Number of CH1 to error							
r/c								r/c							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of CH1 rx error								Number of CH1 ko errors							
r/c								r/c							

- Number of CH1 ko errors
  - The number of ko errors
- Number of CH1 rx errors
  - The number of rx errors
- Number of CH1 to errors
  - The number of to errors
- Number of CH1 of errors
  - The number of of errors

4.28 Aux Error counter CH2 register (HSSAER_AUX_RX_ERR_CH2_REG)

This is register is used to read the number of errors occurred in HSSAER AUX channel 2 lines.

Reading this register we also clear it.

HSSAER_AUX_RX_ERR_CH2_REG (HPUCore Base + 0x78) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of CH2 of error								Number of CH2 to error							
r/c								r/c							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of CH2 rx error								Number of CH2 ko errors							
r/c								r/c							

- Number of CH2 ko errors
  - The number of ko errors
- Number of CH2 rx errors
  - The number of rx errors
- Number of CH2 to errors
  - The number of to errors
- Number of CH2 of errors
  - The number of of errors

4.29 Aux Error counter CH3 register (HSSAER_AUX_RX_ERR_CH3_REG)

This is register is used to read the number of errors occurred in HSSAER AUX channel 3 lines.

Reading this register we also clear it.

HSSAER_AUX_RX_ERR_CH3_REG (HPUCore Base + 0x7C) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of CH3 of error								Number of CH3 to error							
r/c								r/c							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of CH3 rx error								Number of CH3 ko errors							
r/c								r/c							

- Number of CH3 ko errors
  - The number of ko errors
- Number of CH3 rx errors
  - The number of rx errors
- Number of CH3 to errors
  - The number of to errors
- Number of CH3 of errors
  - The number of of errors

4.30 SpiNNlink Start command key (SPNN_START_KEY_REG)

This is register is used to define the Command Key that HPU Core expects to receive from SpiNNaker before starting to transmit Packets to it.

TX Spinnaker Module is in "dump mode" until the Key is received.

SPNN_START_KEY_REG (HPUCore Base + 0x80) Reset Value: **0x80000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value of START Command Key (MSB)															
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value of START Command Key (LSB)															
r/w															

- Value of START Command Key

NOTE: if both SPNN_START_KEY_REG and SPNN_STOP_KEY_REG are set to 0x00000000, the functionality is bypassed and TX interface is allowed to transmit despite of a START command has not be received.

4.31 SpiNNlink Stop command key (SPNN_STOP_KEY_REG)

This is register is used to define the Command Key that HPU Core expects to receive from SpiNNaker before stopping to transmit Packets to it.

TX Spinnaker Module is in "dump mode" after the Key is received.

SPNN_START_KEY_REG (HPUCore Base + 0x84) Reset Value: **0x40000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value of STOP Command Key (MSB)															
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value of STOP Command Key (LSB)															
r/w															

- Value of STOP Command Key

NOTE: if both SPNN_START_KEY_REG and SPNN_STOP_KEY_REG are set to 0x00000000, the functionality is bypassed and TX interface is allowed to transmit despite of a START command has not be received.

4.32 SpiNNlink TX Data Mask (SPNN_TX_MASK_REG)

This is register is used to define the mask applied to data that are to be transmitted to SpiNNaker.

The default value is 0x00FFFFFF

SPNN_TX_MASK_REG (HPUCore Base + 0x88) Reset Value: **0x00FFFFFF**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value of START Command Key (MSB)															
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value of START Command Key (LSB)															
r/w															

- Value of START Command Key

4.33 SpiNNlink RX Data Mask (SPNN_RX_MASK_REG)

This is register is used to define the mask applied to data received from SpiNNaker.

Please note: the mask affect only data, and not commands (i.e. START and STOP commands)

The default value is 0x00FFFFFF

SPNN_RX_MASK_REG (HPUCore Base + 0x8C) Reset Value: **0x00FFFFFF**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value of STOP Command Key (MSB)															
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value of STOP Command Key (LSB)															
r/w															

- Value of STOP Command Key

4.34 SpiNNlink Control Register (SPNN_CTRL_REG)

This is register is used to define some specific controls dedicated to SpiNNaker link.

The default value is 0x00000000

SPNN_CTRL_REG (HPUCore Base + 0x90) Reset Value: **0x00FFFFFF**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX LEFT SpiNNlink								RX RIGHT SpiNNlink							
							START STOP key enable								START STOP key enable
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX AUX SpiNNlink								TX SpiNNlink							
							START STOP key enable						Force STOP	Force START	Time out Disable
r/w															

- TX SpiNNlink Timeout disable
  When a packet transmission is in progress, a timeout counter monitors the activity of acknowledge signal from SpiNnaker, and –if enabled- set the "dump mode" when after 128 clock cycles (1.28 us) there isn't any response.
  - When "0" the timeout counter is enabled
  - When "1" the timeout counter is disabled
- TX SpiNNlink Force START (Write and Clear bit)
  - When a "1" is wrote, the TX stage is forced to Start operations even if START command has not been received from SpiNNaker. Note that the bit clears itself immediately, and the command acts even if bits 8, 16 or 24 are set to "1".
- TX SpiNNlink Force STOP
  - When a "1" is wrote, the TX stage is forced to Stop operations even if STOP command has not been received from SpiNNaker. Note that the bit clears itself immediately, and the command acts even if bits 8, 16 or 24 are set to "1".
- RX AUX SpiNNlink START/STOP Key Command Enable
  - When "1", RX stage is allowed to recognize START and STOP Commands from SpiNNaker.
- RX RIGHT SpiNNlink START/STOP Key Command Enable
  - When "1", RX stage is allowed to recognize START and STOP Commands from SpiNNaker.
- RX LEFT SpiNNlink START/STOP Key Command Enable
  - When "1", RX stage is allowed to recognize START and STOP Commands from SpiNNaker.

4.35 SpiNNlink Status Register (SPNN_STATUS_REG)

This is register is used to report some specific status related to SpiNNaker link.

SPNN_RX_MASK_REG (HPUCore Base + 0x94) Reset Value: **0x00FFFFFF**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RX LEFT SpiNNlink								RX RIGHT SpiNNlink							
						Parity Error	Rx Error							Parity Error	Rx Error
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX AUX SpiNNlink								TX SpiNNlink							
						Parity Error	Rx Error							Offload	Link Time out
r															

- TX SpiNNlink Link Timeout
  When a packet transmission is in progress, a timeout counter monitors the activity of acknowledge signal from SpiNnaker, and –if enabled- set the "dump mode" when after 128 clock cycles (1.28 us) there isn't any response.
  - When "1" the timeout counter reached the end
- TX SpiNNlink Offload
  - When "1" SpiNNlink is in Offload Dump Mode because START command has not been received/forced or STOP command has been received/forced
- RX AUX SpiNNlink Rx Error
  - When '0', the Aux SpiNNlink RX is working
  - When '1', the Aux SpiNNlink RX is receiving wrong symbols
- RX AUX SpiNNlink Parity Error
  - When '0', the Aux SpiNNlink RX is working
  - When '1', the Aux SpiNNlink RX is receiving packets with wrong parity
- RX Right SpiNNlink Rx Error
  - When '0', the Right SpiNNlink RX is working
  - When '1', the Right SpiNNlink RX is receiving wrong symbols
- RX Right SpiNNlink Parity Error
  - When '0', the Right SpiNNlink RX is working
  - When '1', the Right SpiNNlink RX is receiving packets with wrong parity
- RX Left SpiNNlink Rx Error
  - When '0', the Left SpiNNlink RX is working
  - When '1', the Left SpiNNlink RX is receiving wrong symbols
- RX Left SpiNNlink Parity Error
  - When '0', the Left SpiNNlink RX is working
  - When '1', the Left SpiNNlink RX is receiving packets with wrong parity

4.36 TLAST TIMEOUT Register (TLASTTO_REG)

This register is used to issue a premature end of an Axistream burst. When the time counted by this register expires and at least a couple of data have been trasferred, a dummy data (0xF0CACC1A) is sent as last data of the burst. In this way we can perform a premature end of an axistream burst. This is useful to decrease the latency in DMA responsiveness.

TLASTTO_REG (HPUCore Base + 0xA0) Reset Value: **0x00010000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of clock period to perform a premature Axistream burst end during a "slow" transfer															
r/w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of clock period to perform a premature Axistream burst end during a "slow" transfer															
r/w															

4.37 TLAST Counter Register (TLASTCNT_REG)

This register is used to read the number of valid Tlast Axistream events that the HPU has performed.

TLASTCNT_REG (HPUCore Base + 0xA4) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of Tlast events in RX Fifo															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of Tlast events in TX Fifo															
r/o															

4.38 TData Counter Register (TDATACNT_REG)

This register is used to read the number of valid Axistream DATA that the HPU has performed.

TDATACNT_REG (HPUCore Base + 0xA8) Reset Value: **0x00000000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Number of TDATA valid in RX Fifo															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Number of TDATA valid in TX Fifo															
r/o															

4.39 Gigabit Transceiver receiver - left (GTRX_LEFT_REG)

This register is dedicated to the Gigabit Transceiver receiver on the left side.

Reset Value: **0x00000000**

GTRX_LEFT_REG (HPUCore Base + 0xB0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Events per milliseconds															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Messages per milliseconds														GT Disaligned	PLL Error
r/o														r/o	r/o

- PLL Error: Clock is missing or the PLL is not locked
- GT Disaligned: gtp receiver is not aligned with transmitted words
- Messages per milliseconds
  - Reports number of 8-bits messages sent per millisecond via GTP
- Events per milliseconds
  - Reports number of events sent per millisecond via GTP

4.40 Gigabit Transceiver receiver - right (GTRX_RIGHT_REG)

This register is dedicated to the Gigabit Transceiver receiver on the left side.

Reset Value: **0x00000000**

GTRX_RIGHT_REG (HPUCore Base + 0xB4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Events per milliseconds															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Messages per milliseconds														GT Disaligned	PLL Error
r/o														r/o	r/o

- PLL Error: Clock is missing or the PLL is not locked
- GT Disaligned: gtp receiver is not aligned with transmitted words
- Messages per milliseconds
  - Reports number of 8-bits messages sent per millisecond via GTP
- Events per milliseconds
  - Reports number of events sent per millisecond via GTP

4.41 Gigabit Transceiver receiver – aux (GTRX_AUX_REG)

This register is dedicated to the Gigabit Transceiver receiver on the left side.

Reset Value: **0x00000000**

GTRX_AUX_REG (HPUCore Base + 0xB8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Events per milliseconds															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Messages per milliseconds														GT Disaligned	PLL Error
r/o														r/o	r/o

- PLL Error: Clock is missing or the PLL is not locked
- GT Disaligned: gtp receiver is not aligned with transmitted words
- Messages per milliseconds
  - Reports number of 8-bits messages sent per millisecond via GTP
- Events per milliseconds
  - Reports number of events sent per millisecond via GTP

4.42 Gigabit Transceiver transmitter (GTTX_REG)

This register is dedicated to the Gigabit Transceiver receiver on the left side.

Reset Value: **0x00000000**

GTTX _REG (HPUCore Base + 0xBC)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Events per milliseconds															
r/o															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Messages per milliseconds														GT Disaligned	PLL Error
r/o														r/o	r/o

- PLL Error: Clock is missing or the PLL is not locked
- GT Disaligned: gtp receiver is not aligned with transmitted words
- Messages per milliseconds
  - Reports number of 8-bits messages sent per millisecond via GTP
- Events per milliseconds
  - Reports number of events sent per millisecond via GTP

5 References

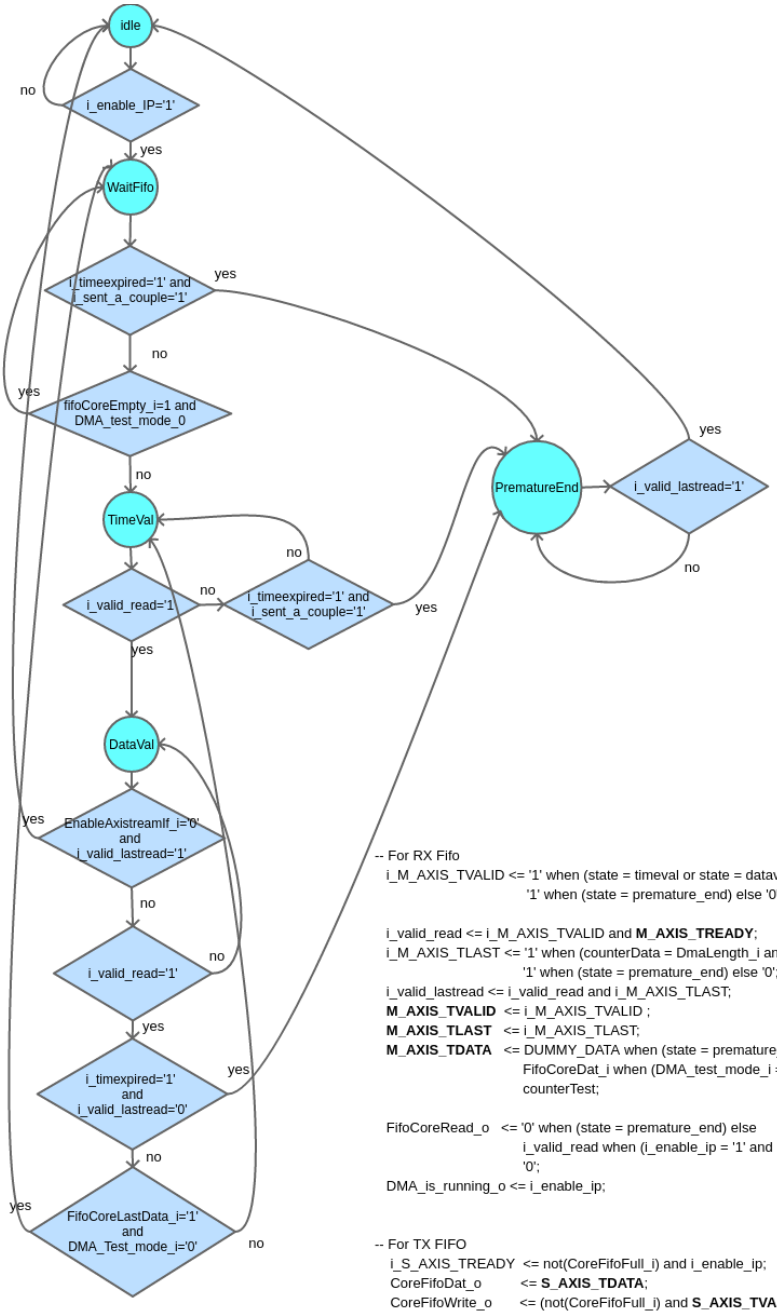
[1] ARM AMBA AXI protocol v2.0

[2] "Combining the ADS1202 with an FPGA Digital Filter for Current Measurement in Motor Control Applications", Texas Instruments, Application Report SBAA094 – June 2003.

[3] "Asynchronous DC-Free Serial Protocol for Event-Based AER Systems", P. Motto Ros, M. Crepaldi, C. Bartolozzi and D. Demarchi, 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)

6 Appendixes

6.1 Axistream ASM




```
-- For RX Fifo
i_M_AXIS_TVALID <= '1' when (state = timeval or state = dataval) else
                   '1' when (state = premature_end) else '0';

i_valid_read <= i_M_AXIS_TVALID and M_AXIS_TREADY;
i_M_AXIS_TLAST <= '1' when (counterData = DmaLength_i and i_M_AXIS_TVALID = '1') else
                  '1' when (state = premature_end) else '0';
i_valid_lastread <= i_valid_read and i_M_AXIS_TLAST;
M_AXIS_TVALID  <= i_M_AXIS_TVALID ;
M_AXIS_TLAST   <= i_M_AXIS_TLAST;
M_AXIS_TDATA   <= DUMMY_DATA when (state = premature_end) else
                  FifoCoreDat_i when (DMA_test_mode_i = '0') else
                  counterTest;

FifoCoreRead_o   <= '0' when (state = premature_end) else
                    i_valid_read when (i_enable_ip = '1' and DMA_test_mode_i = '0') else
                    '0';
DMA_is_running_o <= i_enable_ip;
```

```
-- For TX FIFO
 i_S_AXIS_TREADY  <= not(CoreFifoFull_i) and i_enable_ip;
 CoreFifoDat_o    <= S_AXIS_TDATA;
 CoreFifoWrite_o  <= (not(CoreFifoFull_i) and S_AXIS_TVALID) and i_enable_ip;
 S_AXIS_TREADY    <= i_S_AXIS_TREADY;
```