ELSYS DESIGN	**CNES**	**Elsys-Design**

Reference	**P24-9771_TD_SpaceFibre_Light**	*Revision*	**0.2**
Publication date	**20/05/205**		
Confidentiality	**None**		

TESTBENCH DESCRIPTION OF THE SPACEFIBRE LIGHT IP

	Name	**Function**	**Date**	**Visa**
Written by	Thomas FAVRE-FELIX	FPGA Engineer	20/05/2025	Thomas Favre-Felix Signature numérique de Thomas Favre-Felix Date : 2025.05.21 10:46:31 +02'00'
Verified by	Y. DAURIAC	FPGA Engineer	20/05/2025	Yvan Dauriac Signature numérique de Yvan Dauriac Date : 2025.05.21 09:46:41 +02'00'
Approved by	A. DANIEL	FPGA Expert	20/05/2025	Arnaud Daniel Signature numérique de Arnaud Daniel Date : 2025.05.21 06:20:53 +02'00'

DOCUMENT HISTORY

Revision	Date	Author	Description of evolution
0.0	10/07/2024	Yvan DAURIAC	Creating the document
0.0	29/08/2024	Thomas FAVRE FELIX	Adding Virtual Testbench Description
0.0	03/02/25	Yvan DAURIAC	Reorganisation of the different chapter, adding Physical platform description.
0.1	26/02/25	Thomas FAVRE-FELIX	Adding the evolution of the testbench to include the Data Link layer in Configuration 2 Testbench description
0.2	20/05/2025	Thomas FAVRE-FELIX	Deletion of Configuration 1 dedicated to Phy+Lane layer only. All layers are covered by Configuration 2

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. PRESENTATION OF THE DOCUMENT

This document presents the architecture of the SpaceFibreLight IP testbench and describes the models for operating the different IP configurations.

The purpose of the test bench is to operate the SpaceFibreLight IP in accordance with the specification document.

The test bench is composed of:

- 1 Configurator
- 1 Lane Data & Control Generator
- 1 Lane Data & Control Analyzer
- 1 Data Link Broadcast Data Generator
- 1 Data Link Broadcast Analyzer
- 8 Data Link Data Generator
- 8 Data Link Data Analyzer

1.2. DOCUMENT DEFINITIONS

1.2.1. REFERENCE DOCUMENTS

Name	Title	Reference
[REF_01]	Device Requirement Specification	P24_9771_DRS_SpaceFibreLight_v0.3

TABLE 1 : REFERENCE DOCUMENTS

1.3. TERMINOLOGY

1.3.1. ACRONYMS AND ABBREVIATIONS

Acronym – Abbreviation	**Meaning**
AXI	Advanced eXtensible Interface
eSATA	External Serial Advanced Technology Attachment
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array

GDB	GNU Debugger
IP	Intellectual Property
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
MIB	Managed Information Base
PHY	Physical
RISC	Reduced Instruction Set computer
RX	Receiver
TB	Testbench
TX	Transmitter
UART	Universal Asynchronous Receiver / Transmitter
USB	Universal Serial Bus
WSL	Windows Subsystem for Linux

Table 1: Table of Acronyms

2. PRESENTATION OF THE GENERIC TEST BENCH ARCHITECTURE

2.1. GENERIC ARCHITECTURE

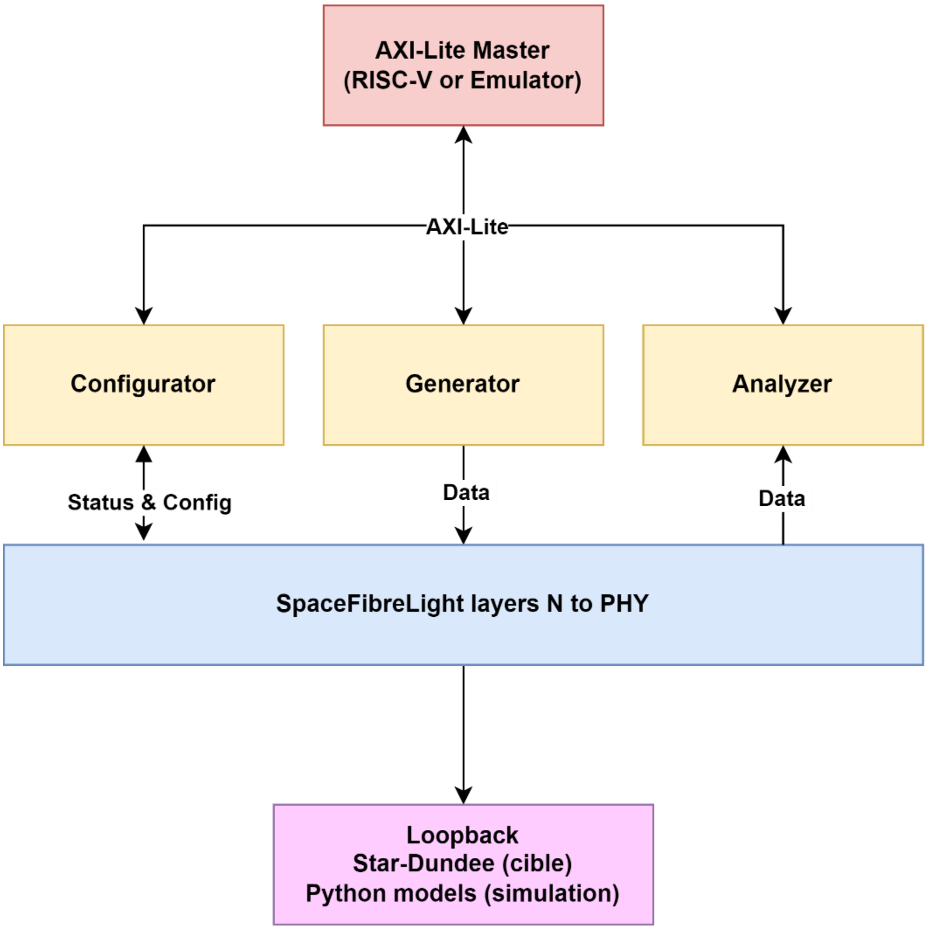

Figure 1: Generic Validation Architecture

The generic architecture of the test bench contains the following elements:

- An AXI-Lite master (RISC V/ emulator) to control the different models
- Templates to operate the IP according to its configuration:
    - A configurator
    - One or more generators
    - One or more analyzers
    - Additional modules if needed to manage certain control and status signals
- TX/RX interface of the physical layer, via equipment (on target) or model (in simulation)

2.2. PHYSICAL TESTBENCH ARCHITECTURE

2.2.1. PHYSICAL TESTBENCH ARCHITECTURE

Figure 2 provides a diagram of the actual physical test bench with all its components

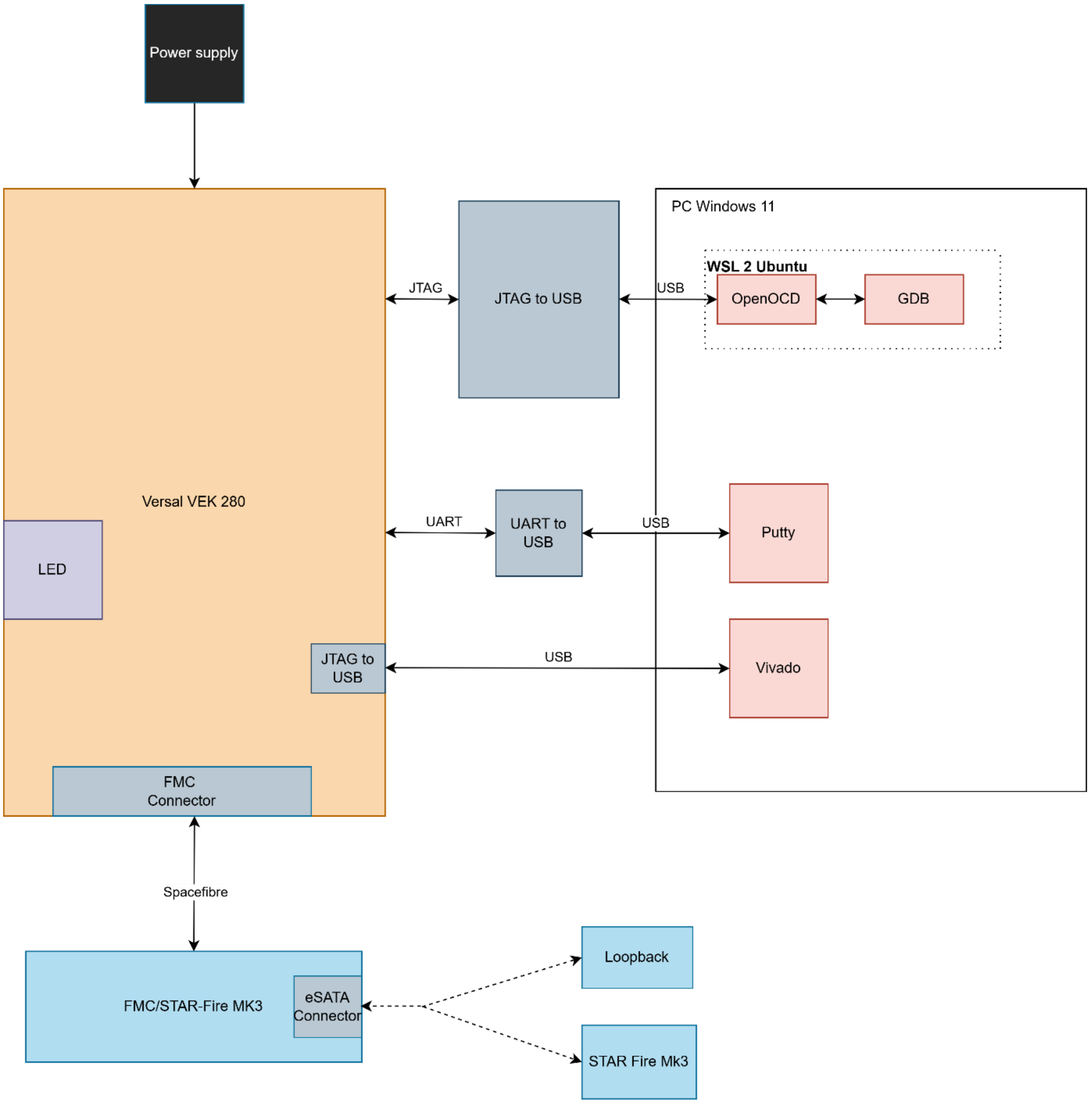

Figure 2: Physical testbench block diagram

The physical testbench consists of:

- AMD-Xilinx Versal VEK 280 (Evaluation Board)
- LED for Debug
- FMC STAR-Fire MK3 (STAR Dundee) to certify compliance with the spacefibre protocol

- JTAG to USB/ Vivado to flash the FPGA
- UART to USB/Putty allowing you to have a UART console so that you can run tests and see the results
- JTAG to USB/OpenOCD/GDB for Debug uses

2.3. AXI4-LITE

The AXI4-Lite protocol provides a bidirectional point-to-point interface between a master and a slave. It is a simplified version of the AXI protocol with a reduced set of signals.

2.3.1. READ

Figure 3 presents a read transaction using an AXI4-Lite interface. The master sends a read address to the slave via Read address channel (AR). Via Read data channel (R), the slave responds with the read data and a validation signal. Valid signals indicate that the values in the payload (data) fields are valid. Ready signals indicate whether the slave or master is ready for new transactions.

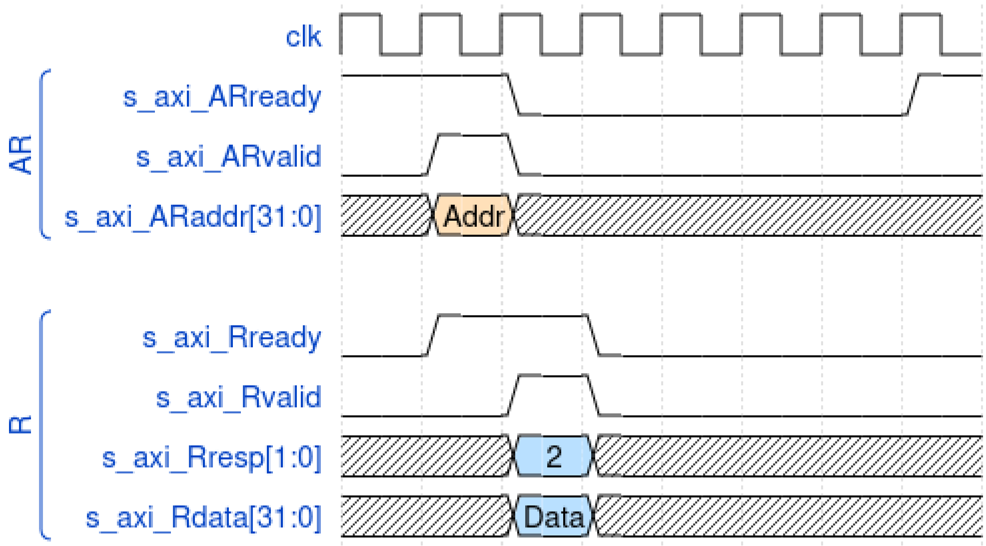

Figure 3: AXI4-Lite read transaction

2.3.2. WRITE

Figure 4 presents a write transaction using an AXI4-Lite interface. The master sends a write address to the slave via Write address channel (AW) and the data to be written via Write data channel (W). Via Write response channel (B), the slave responds with a validation signal when the write is complete. Valid signals indicate that the values in the payload (data) fields are valid. Ready signals indicate whether the slave or master is ready for new transactions.

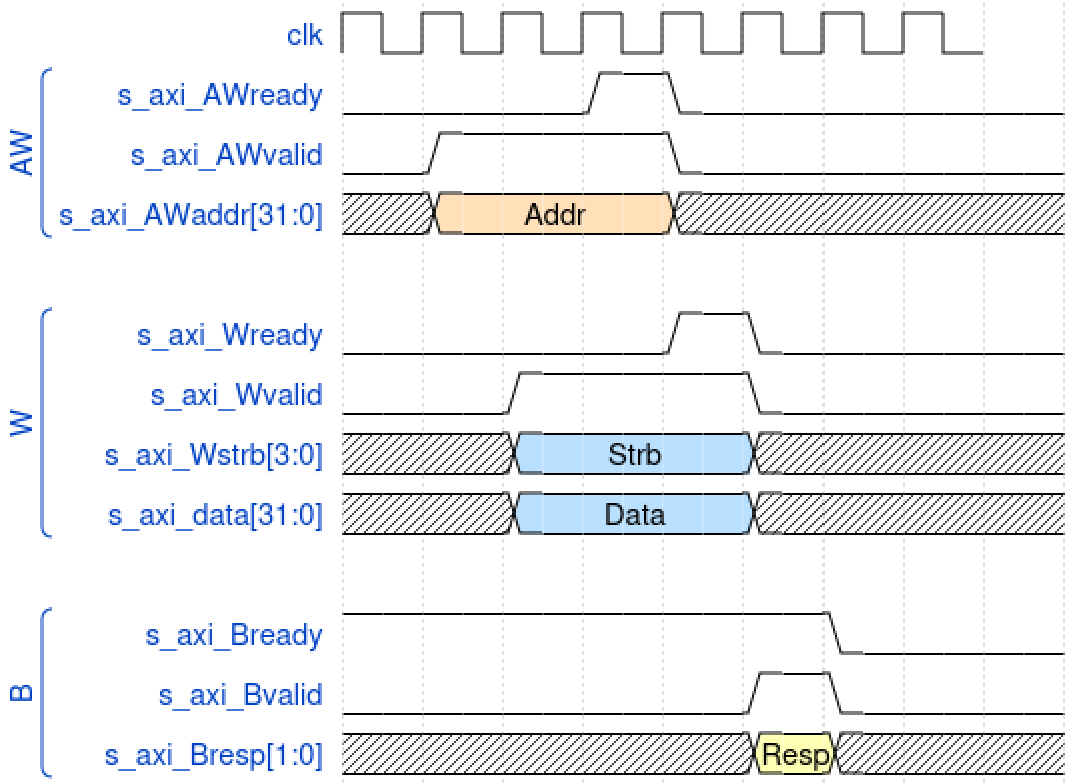

Figure 4:AXI4-Lite write transaction

2.3.1. SLAVE PORTS

Table 2 describes all the ports to an AXI4-lite *slave* in this project. “ADDR_WIDTH” and “DATA_WIDTH” are parameters specific to each model

Name	Direction	Dimensions	Description
s_axi_awvalid	Input	1	AW slave channel (AXI4-Lite)
s_axi_awready	Output	1	
s_axi_awaddr	Input	ADDR_WIDTH	
s_axi_awprot	Input	3	
s_axi_awvalid	Input	1	
s_axi_wvalid	Input	1	W slave channel (AXI4-Lite)
s_axi_wready	Output	1	
s_axi_wdata	Input	DATA_WIDTH	
s_axi_wstrb	Input	DATA_WIDTH / 8	
s_axi_arvalid	Input	1	AR slave channel (AXI4-Lite)

s_axi_arready	Output	1	
s_axi_araddr	Input	ADDR_WIDTH	
s_axi_arprot	Input	3	
s_axi_bvalid	Output	1	B slave channel (AXI4-Lite)
s_axi_bready	Input	1	
s_axi_bresp	Output	2	
s_axi_rvalid	Output	1	R slave channel (AXI4-Lite).
s_axi_rready	Input	1	
s_axi_rdata	Output	DATA_WIDTH	
s_axi_rresp	Output	2	

Table 2: Axi4-lite slave ports

2.4. AXI4-STREAM

2.4.1. TRANSACTION PROTOCOLE

Figure 5 presents a data transaction using an AXI4-Stream interface. The master sends the data on TDATA and asserts TVALID to indicate that the data on TDATA can be read. The TREADY signal indicate whether the slave is ready to read on TDATA. When TVALID and TREADY are asserted, TDATA is considered to be read, and the next data of the packet is sent on TDATA. When the last data of a packet is sent, TLAST is asserted. When a control character is used, the TUSER bit corresponding to the control character byte is asserted.

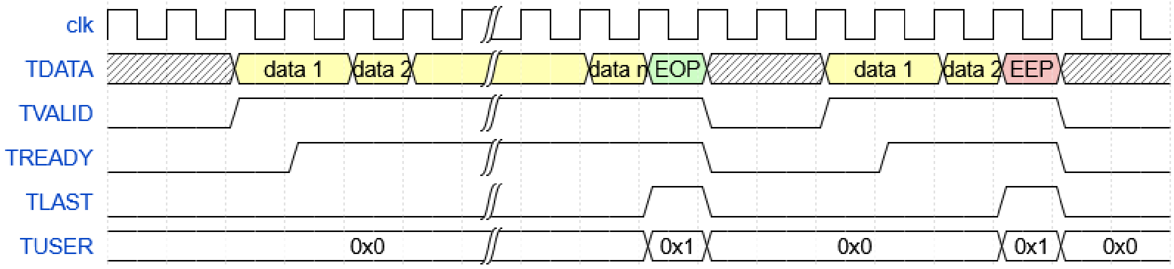

Figure 5: AXI4-Stream transaction

2.4.2. PORTS

Table 3 describes all the signal between a AXI4-Stream slave and its master in this project. “DATA_WIDTH” is a parameter specific to each model.

Name	Direction	Dimensions	Description
TDATA	Master to Slave	DATA_WIDTH	Data sent from the master to the slave
TVALID	Master to Slave	1	Indicates that TDATA value is valid
TREADY	Slave to Master	1	Indicates that TDATA value can be read by the slave
TLAST	Master to Slave	1	Indicates that the data sent is the last of a packet
TUSER	Master to Slave	DATA_WIDTH/8	Indicates that the corresponding byte is a control character

Table 3: Axi4-Stream ports

3. CONFIGURATION 2: COUCHE PHY/LANE/DATA LINK

3.1. TEST BENCH ARCHITECTURE

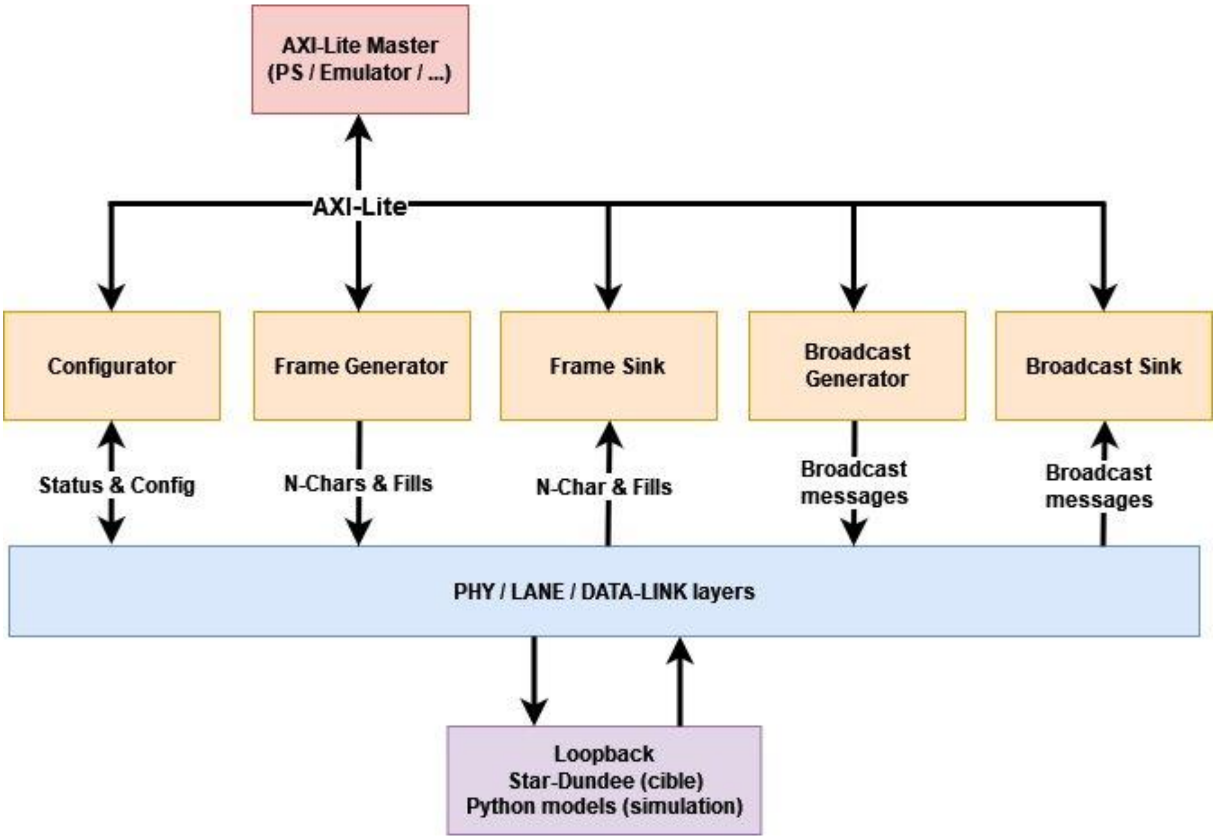

Figure 6: Test Bench Architecture for Configuration 2

Figure 6 shows the architecture to validate the PHY+LANE+DATA LINK layers. This part describes:

- Physical testbench environment
- Virtual testbench environment
- RTL models:
  - A configurator named data_link_configurator which manages the PHY+LANE+DATA LINK MIB blocks
  - A generator named lane_generator which generates data and control words to the PHY+LANE layers
  - An analyzer named lane_analyzer which analyzes data and control words from the PHY+LANE layers
  - A generator named data_link_generator which generates data and control words to the PHY+LANE+DATA LINK layers
  - An analyzer named data_link_analyzer which analyzes data and control words from the PHY+LANE+DATA LINK layers
- Pyhon models

3.1.1. PHYSICAL ARCHITECTURE

The Figure 7 describes the architecture of the FPGA containing the DUT. The FPGA is broken down into two main parts:

- Configuration testbench SpaceFibre: corresponds to the description given above
- Subsystem RISCV: describes the architecture of the RISCV subsystem

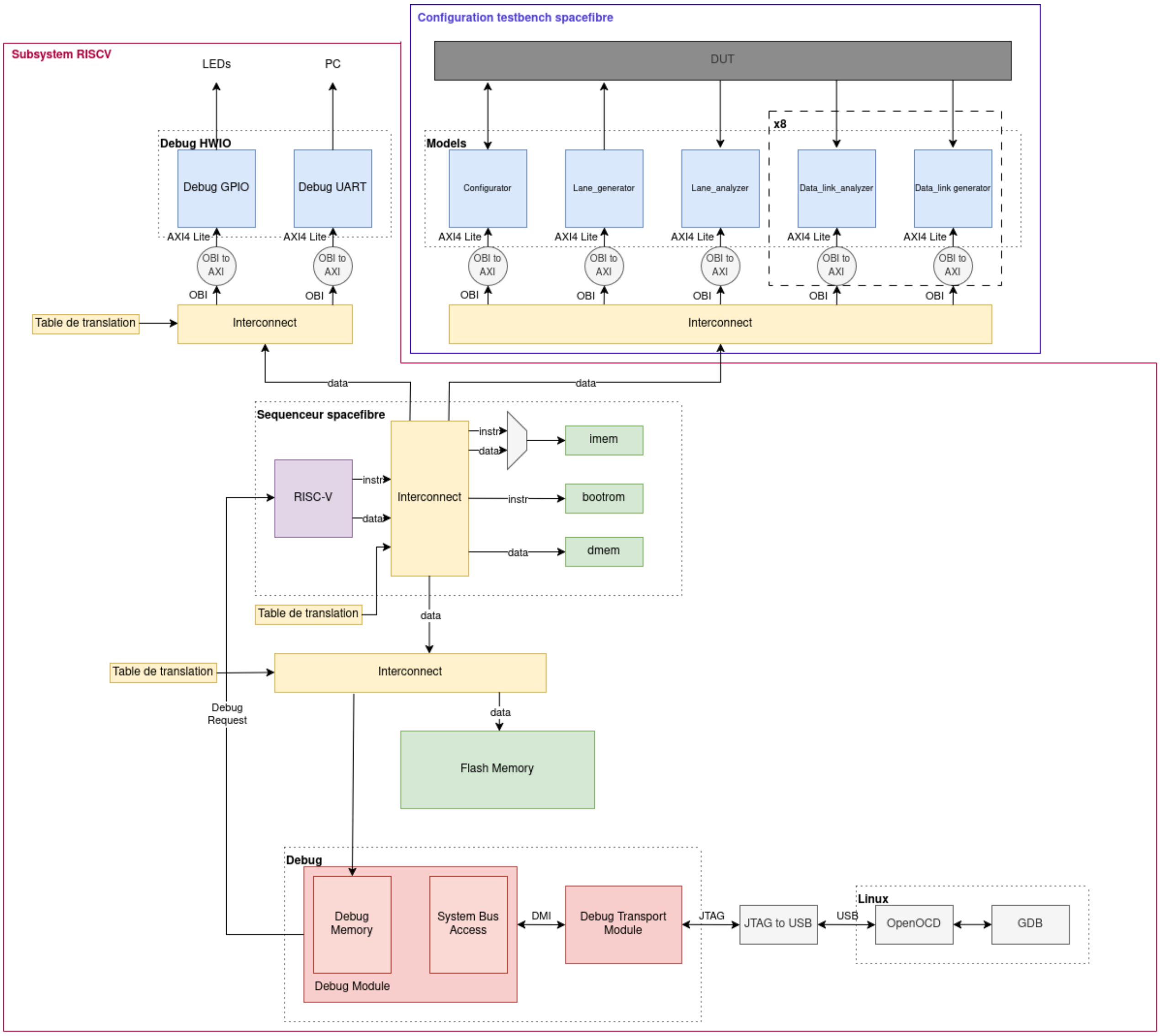

Figure 7: FPGA Block Diagram

3.1.2. VIRTUAL ARCHITECTURE

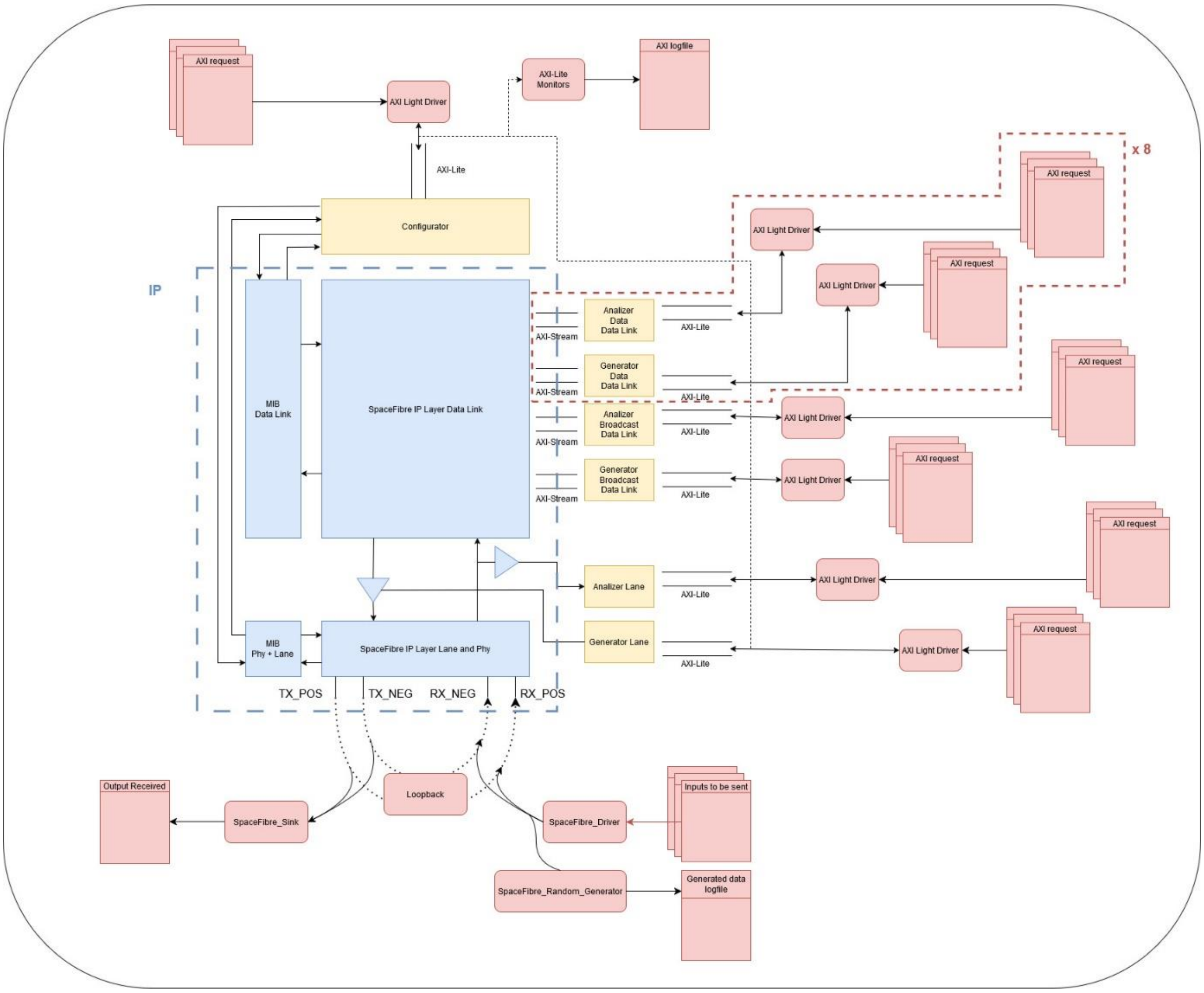

Figure 8: Virtual Testbench block diagram

The Virtual Testbench is used to wrap the IP and synthesizable models for simulation. It is implemented in Python using cocotb library. An AXI4-Lite driver and sink will pilot the models, and different virtual models will receive and transmit data to the IP RX/TX ports. The whole Virtual Testbench is implemented as a class in Python, named TB.

3.2. RTL MODELS (PHYSICAL AND VIRTUAL TB)

3.2.1. LANE_GENERATOR MODEL

3.2.1.1. MODEL OVERVIEW

The lane_generator model is used to communicate with the lane layer. It communicates with the RISC-V or emulator via an AXI4-lite bus and with the IP via discrete signals. Its role is to generate frames compatible with the data format of the lane layer. In this model, it is possible to configure the number of frames, the frame size, the inter-packet delay and the generation of the data (incremental, PRBS) This model is able to give a sequence number of a frame, generate data.

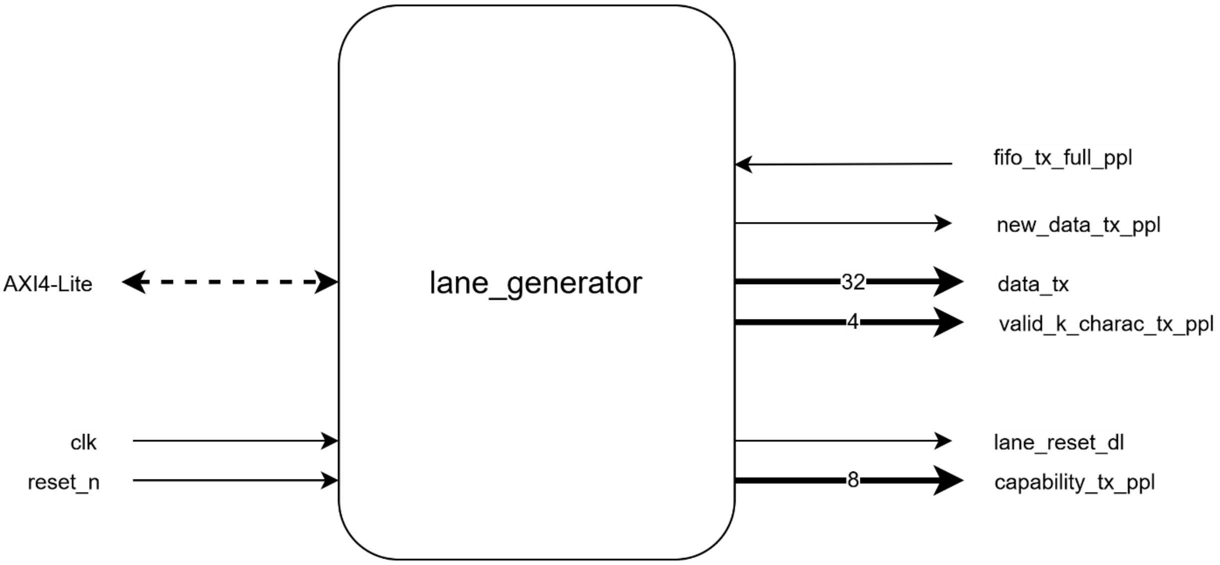

Figure 9: lane_generator block diagram

3.2.1.2. FUNCTIONAL DESCRIPTION

3.2.1.2.1. FUNCTIONAL MODE

3.2.1.2.1.1. GENERATION DATA MODE

The Figure 10 shows what the DUT will receive.

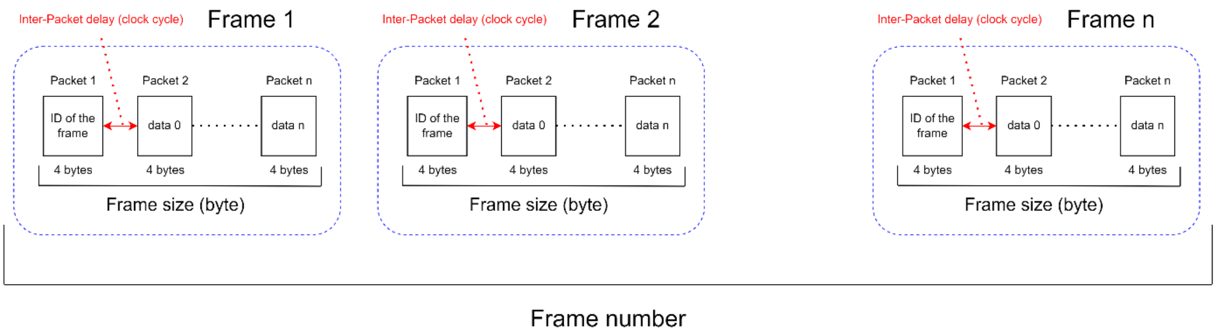

Figure 10: data generation flow diagram

The frame ID corresponds to the frame number.

Data are generated in two ways:

- Incremental: the initial value is configurable and is incremented by 1 for each packet.
- PRBS: The seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

3.2.1.2.1.2. GENERATION CONTROL WORD

Generate the following control word from the data-link layer in following order :

- SDF: D0.0, D0.0, D16.2, K28.7
- EDF: D0.0, D0.0, D0.0, K28.0
- SBF: D0.0, D0.0, D29.2, K28.7
- EBF: D0.0, D0.0, D0.0, K28.2
- SIF: D0.0, D0.0, D4.2, K28.7

3.2.1.2.1.3. GENERATION WRONG K-CHARACTER

Generate one wrong K-character: D0.0 D0.0 D0.0 K0.0

3.2.1.2.2. PARAMETERS

The Table 4 lists the parameters of the lane_generator model. So that this module can be integrated into various systems, the width of the various AXI interface signals can be configured using parameters.

Name	Type	Description
ADDR_WIDTH	Positive integer	AXI address bus width
DATA_WIDTH	Positive integer	AXI data bus width

Table 4: lane_generator model parameters

3.2.1.2.3. PORTS

The Table 5 lists all the ports in the lane_generator model. The model has a slave interface compliant with the AXI4-Lite standard.

The following bus corresponds to the data sent to the lane layer

Name	Direction	Dimensions	Description
Global signals			
Clk	Input	1	Main clock.
reset_n	Input	1	Reset low active

AXI4-Lite *slave* Port			
AXI4-lite	Inout	/	Table 2: Axi4-lite slave ports
Lane interface			
Data_tx	Output	32	Data bus to the lane layer
Lane_reset_dl	Output	1	Lane reset signal to the lane layer
new_data_tx_ppl	Output	1	Flag to write data in FIFO TX
fifo_tx_full_ppl	Input	1	Flag full of the FIFO TX
valid_k_charac_tx_ppl	Output	4	K charachter valid in the 32-bit DATA_TX_PPL vector
capability_tx_ppl	Output	8	Capability send on TX link in INIT3 control word

Table 5: lane_generator model ports

3.2.1.2.4. REGISTER MAP

The Table 6 lists the registers in the lane_generator model:

- A configuration register with read and write access.
- A control register with read and write access.
- A status register with read only access.
- A initial value register with read and write access.

Address	**Name**	**Description**
0x00	Configuration Register	Register for all the parameters of lane_generator model
0x04	Control Register	Register for all the control signals of the lane_generator model
0x08	Status Register	Register for all the status signals of the lane_generator model
0x0C	Initial Value Register	Register for the initial value of the data generated
0x10-0xFF	Reserved	

Table 6: lane_generator register model

Bits	Name	Core Access	Reset Value	Description
0-4	Frame number	Read/write	0	Number of frames to send
5-13	Frame size	Read/write	0	Size of the frame (byte)
14-23	Inter-packet delay	Read/write	0	Inter-packet delay
24	Generation data	Read/write	0	Type of generation of data: -0: Incremental -1: PRBS
25-26	Data mode	Read/write	0	00: Generation data 01: Generation control word 10: Wrong K-character 11: Reserved
27-31	Reserved			

Table 7: lane_generator configuration register description

Bits	Name	Core Access	Reset Value	Description
0	Model start	Read/write	0	Start generating data
1	Lane Reset	Read/write	0	Reset the lane via the model
2-9	Lane Capabilities	Read/write	0	Lane capabilities status
10-31	Reserved			

Table 8: lane_generator control model register description

Bits	Name	Core Access	Reset Value	Description
0	Busy	Read Only	0	Test in progress
1	Test End	Read Only	0	Test finished
2-9	Error counter	Read Only	0	Counter for number of errors, locks when at maximum
10-31	Reserved			

Table 9: lane_generator status model register description

Bits	Name	Core Access	Reset Value	Description
0-31	Initial Value	Read/write	0	Initial value for the data

Table 10: lane_generator initial value register description

3.2.2. LANE_ANALYZER MODEL

3.2.2.1. MODEL OVERVIEW

The lane_analyzer model is used to communicate with the lane layer. It communicates with the RISC-V or emulator via an AXI4-lite bus and with the IP via discrete signals. Its role is to receive frames from the lane layer. In this model, it is possible to configure the number of frames that he has to receive, the frame size, the inter-packet delay and the type of the data (incremental, PRBS) This model is able to receive lane layer frames and identify whether they are correct according to the configuration of its registers.

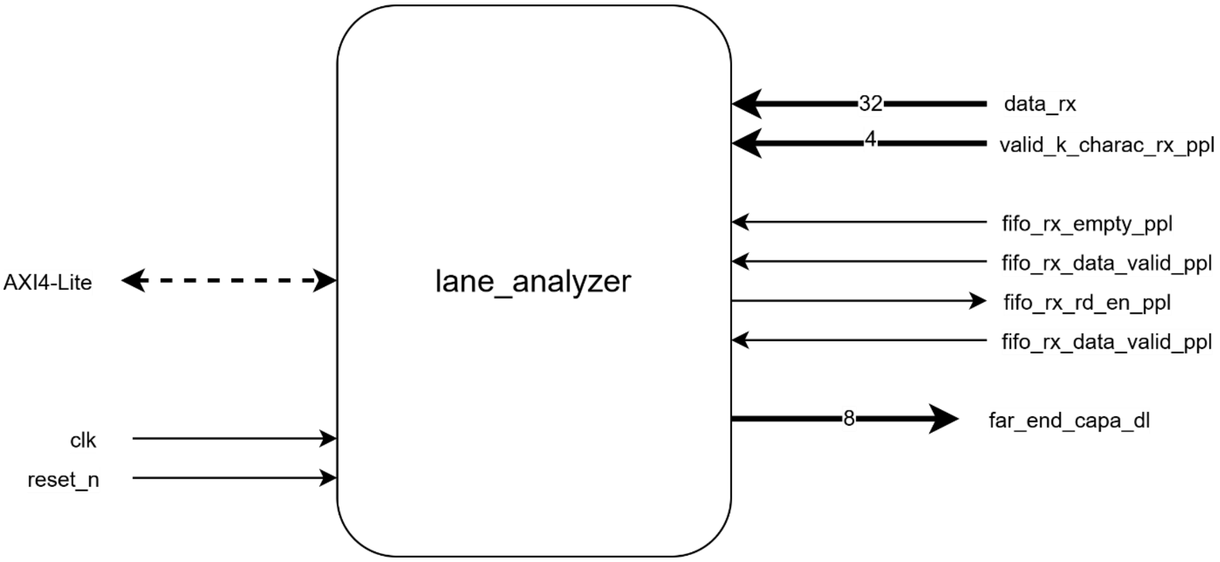

Figure 11: lane_analyzer block diagram

3.2.2.2. FUNCTIONAL DESCRIPTION

3.2.2.2.1. FUNCTIONAL MODE

3.2.2.2.1.1. ANALYZE DATA MODE

The Figure 12 shows what the model must receive.

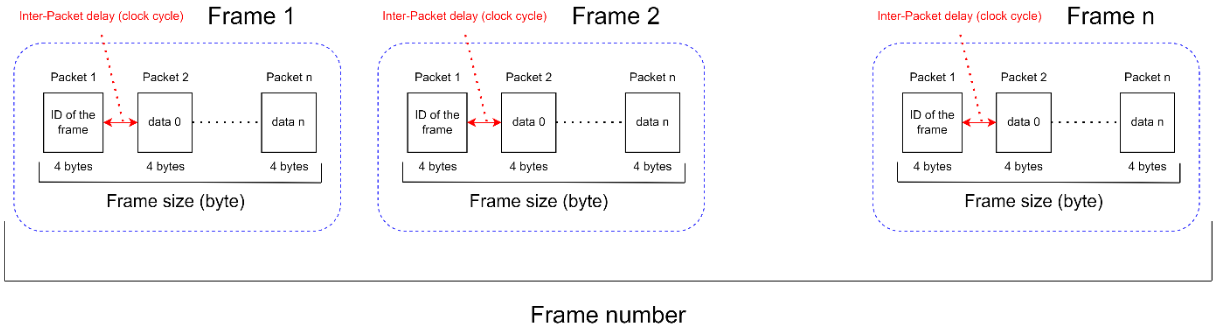

Figure 12: data for analyze flow diagram

The frame ID corresponds to the frame number.

Data are generated in two ways:

- Incremental: the initial value is configurable and is incremented by 1 for each packet.
- PRBS: The seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

3.2.2.2.1.2. ANALYZE CONTROL WORD

Wait to receive all the control word in the following order:

- SDF: D0.0, D0.0, D16.2, K28.7
- EDF: D0.0, D0.0, D0.0, K28.0
- SBF: D0.0, D0.0, D29.2, K28.7
- EBF: D0.0, D0.0, D0.0, K28.2
- SIF: D0.0, D0.0, D4.2, K28.7

3.2.2.2.1.3. ANALYZE RXERR RECEPTION

Wait for the reception RXERR control words: D0.0 D0.0 D0.0 K0.0

The number of RXERR control words expected is defined with the frame number.

3.2.2.2.2. PARAMETERS

The Table 11 lists the parameters of the lane_analyzer model. So that this module can be integrated into various systems, the widths of the various AXI interface signals can be configured using parameters.

Name	Type	Description
ADDR_WIDTH	Positive integer	AXI address bus width
DATA_WIDTH	Positive integer	AXI data bus width

Table 11: lane_analyzer model parameters

3.2.2.2.3. PORTS

The Table 12 lists all the ports in the lane_analyzer model. The model has a slave interface compliant with the AXI4-Lite standard.

The following bus corresponds to the data received from the lane layer

Name	Direction	Dimensions	Description
Global signals			

Clk	Input	1	Main clock.
reset_n	Input	1	Reset low active
AXI4-Lite *slave* Port			
AXI4-lite	Inout	/	Table 2: Axi4-lite slave ports
Lane interface			
Data_rx	input	32	Data bus from the lane layer
Valid_k_charc_rx_ppl	input	4	K charachter valid in the 32-bit DATA_TR_PPL vector
Fifo_rx_empty_ppl	input	1	Flag EMPTY of the FIFO RX
Fifo_rx_data_valid_ppl	input	1	Flag DATA_VALID of the FIFO RX
Fifo_rx_rd_en_ppl	output	1	Flag to read data in FIFO RX
Far_end_capa_dl	input	8	Capability field receive in INIT3 control word

Table 12: lane_analyzer model ports

3.2.2.2.4. REGISTER MAP

The Table 13 lists the registers in the lane_analyzer model:

- A configuration register with read and write access.
- A control register with read and write access.
- A status register with read only access.
- A initial value register with read and write access.

Address	Name	Description
0x00	Configuration Register	Register for all parameters of the lane_analyzer model
0x04	Control Register	Register for all the control signals of the lane_analyzer model
0x08	Status Register	Register for all the status signals of the lane_ analyzer model
0x0C	Initial Value Register	Register for the initial value of the data generated
0x10-0xFF	Reserved	

Table 13: lane_analyzer registers model

Bits	Name	Core Access	Reset Value	Description
0-4	Frame number	Read/write	0	Number of frames to send
5-13	Frame size	Read/write	0	Size of the frame (byte)
14-23	Inter-packet delay	Read/write	0	Inter-packet delay
24	Generation data	Read/write	0	Type of generation of data: -0: incremental -1: PRBS
25-26	Data mode	Read/write	0	00: Generation data 01: Control word 10: RXERR 11: Reserved
27-31	Reserved			

Table 14: lane_analyzer configuration register description

Bits	Name	Core Access	Reset Value	Description
0	Model start	Read/write	0	Start generating data
1-31	Reserved			

Table 15: lane_analyzer control register description

Bits	Name	Core Access	Reset Value	Description
0	Busy	Read Only	0	Test in progress
1	Test End	Read Only	0	Test finished
2-9	Error counter	Read Only	0	Counter for number of errors, locks when at maximum
10-17	Lane Capabilities	Read Only	0	Lane capabilities status
18-31	Reserved			

Table 16: lane_analyzer status register description

Bits	Name	Core Access	Reset Value	Description
0-31	Initial Value	Read/write	0	Initial value for the data

Table 17: lane_analyzer initial value register description

3.2.3. DATA_LINK_CONFIGURATOR MODEL

3.2.3.1. MODEL OVERVIEW

The data_link_configurator model is used to communicate with IP PHY/LANE/DATA LINK MIB blocks. It communicates with the RISC-V or emulator via an AXI4-lite bus and with the IP via discrete signals. Its role is to transmit software parameters to the IP and, conversely, to send all IP status information back to the software.

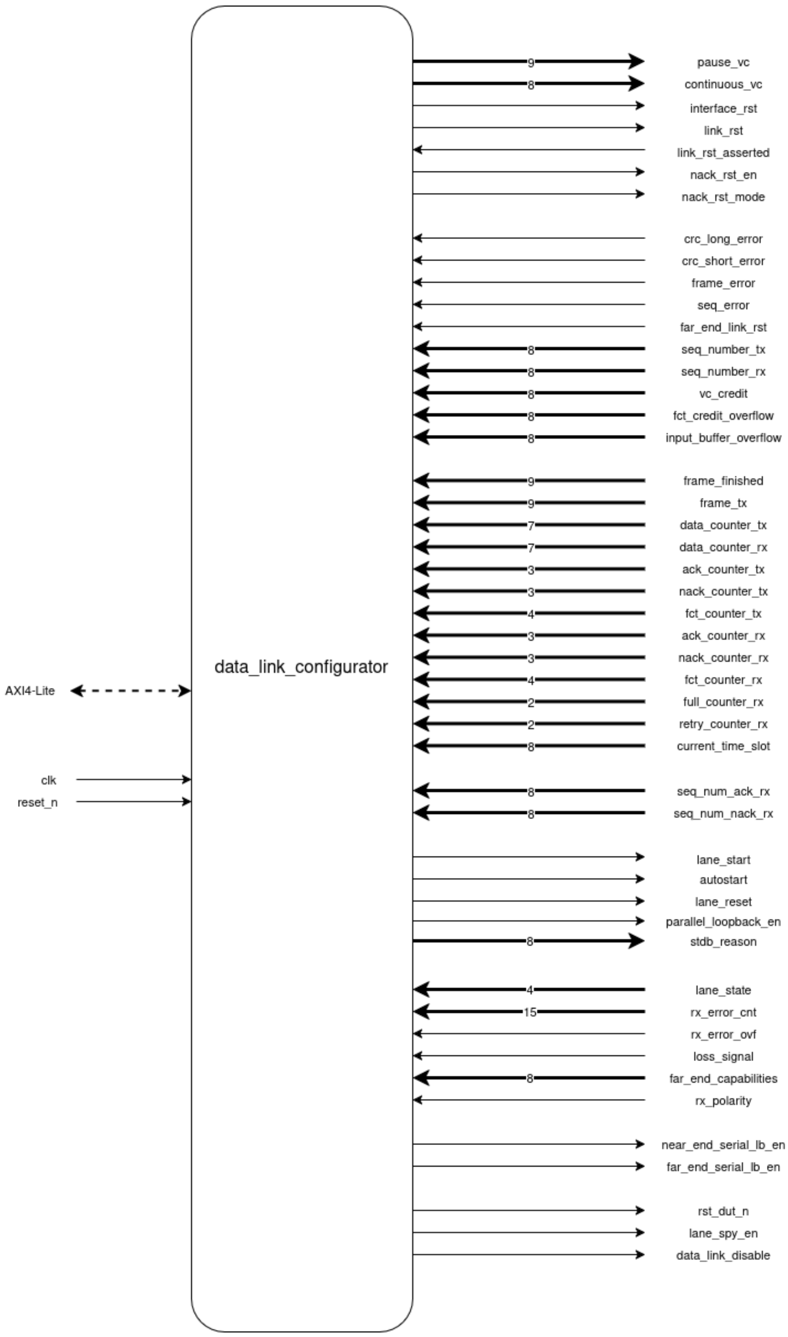

Figure 13: data_link_configurator block diagram

3.2.3.2. FUNCTIONAL DESCRIPTION

3.2.3.2.1. PARAMETERS

Table 18 lists the parameters of the data_link_configurator model. So that this module can be integrated into various systems, the width of the various AXI interface signals can be configured using parameters.

Name	Type	Description
ADDR_WIDTH	Positive integer	AXI address bus width
DATA_WIDTH	Positive integer	AXI data bus width

Table 18: data_link_configurator model parameter

3.2.3.2.2. PORTS

Table 19 lists all the ports in the data_link_configurator model. The model has a slave interface compliant with the AXI4-Lite standard.

The following signals correspond to all MIB configuration signals

Finally, all status signals from the MIB are listed.

Name	Direction	Dimensions	Description
Global signals			
Clk	Input	1	Main Clock.
reset_n	Input	1	Reset low active
AXI4-Lite *slave* Port			
AXI4-lite	Inout	/	Table 2: Axi4-lite slave ports
Configuration Data Link Port			
interface_rst	Output	1	Reset all interface register and Data Link layer
link_rst	Output	1	Reset the link
link_rst_asserted	Input	1	Asserted when the link has been reseted
nack_rst_en	Output	1	Enable automatic reset on NACK reception
nack_rst_mode	Output	1	configure reset on NACK reception strategy between immediate reset or at the end of current packet being received
pause_vc	Output	9	Pause each corresponding virtual channel at the end of the current transmission
continuous_vc	Output	8	Enable continuous mode for each corresponding virtual channel

Status Data Link Port			
seq_number_tx	Input	8	SEQ_NUMBER in transmission
seq_number_rx	Input	8	SEQ_NUMBER in reception
crc_long_error	Input	1	Up if a CRC-16bits error is detected
crc_short_error	Input	1	Up if a CRC-8bits error is detected
frame_error	Input	1	Up if a frame error is detected
seq_error	Input	1	Up if a SEQ_NUMBER error is detected
far_end_link_rst	Input	1	Far-end Link Reset status
vc_credit	Input	8	Up if the corresponding virtual channel far-end input buffer has credit
input_buffer_overflow	Input	8	Up if the corresponding input buffer has overflowed
fct_credit_overflow	Input	8	Up if the corresponding fct credit counter has overflowed
QoS Data Link Port			
frame_finished	Input	9	Up if the corresponding virtual channel frame emission is finished
frame_tx	Input	9	Up if the corresponding virtual channel is emitting a frame
data_counter_tx	Input	7	Number of data sent in the last frame emitted
data_counter_rx	Input	7	Number of data received in the last frame received
ack_counter_tx	Input	3	Indicate the number of ACK control word transmitted during last frame emitted
nack_counter_tx	Input	3	Indicate the number of NACK control word transmitted during last frame emitted
fct_counter_tx	Input	4	Indicate the number of FCT control word transmitted during last frame emitted
ack_counter_rx	Input	3	Indicate the number of ACK control word received during last frame received
nack_counter_rx	Input	3	Indicate the number of NACK control word received during last frame received
fct_counter_rx	Input	4	Indicate the number of FCT control word received during last frame received
full_counter_rx	Input	2	Indicate the number of FULL control word received during last frame received

retry_counter_rx	Input	2	Indicate the number of RETRY control word received during last frame received
current_time_slot	Input	8	Indicate the current time-slot for time -slot QoS
Error Management Port			
seq_num_ack_rx	Input	8	Indicate the last SEQ_NUM value received in an ACK control word
seq_num_nack_rx	Input	8	Indicate the last SEQ_NUM value received in an NACK control word
Configuration Lane Port			
lane_start	Output	1	SpaceFibre lane start initialization signal
autostart	Output	1	Enables communication lane to initialize automatically when a link is established
lane_reset	Output	1	Reset Lane layer signal
parallel_loopback_en	Output	1	Enable parallel loopback of Lane layer
stdb_reason	Output	8	Standby reason field
Status Lane Port			
lane_state	Input	4	Lane state
rx_error_cnt	Input	15	RX Error Counter
rx_error_ovf	Input	1	RX Error Overflow
loss_signal	Input	1	Far-end lost Signal
far_end_capabilities	Input	8	Far-end Capablities
rx_polarity	Input	1	RX Polarity
Configuration Port PHY Layer			
near_end_serial_lb_en	Output	1	Near-End Serial Loopback
far_end_serial_lb_en	Output	1	Far -End Serial Loopback
Global signal			
rst_dut_n	Output	1	Reset DUT (active low)
lane_spy_en	Output	1	Enable the Lane layer spy
data_link_disable	Output	1	Enable the Lane layer injector

Table 19: data_link_configuator model ports

3.2.3.2.3. REGISTER MAP

Table 20 lists the various registers in the lane_configurator model:

- A configuration register with read and write access to the Data Link layer
- A read-only status register for the Data Link layer
- A read-only QoS register for the Data Link layer
- A read-only Error Recovery register for the Data Link layer
- A configuration register with read and write access to the Data Link layer
- A read-only status register for the Data Link layer
- A configuration register with read and write access to the PHY layer
- A global register

Address	**Name**	**Description**
0x00	Global Register	Register to drive different signals
0x04	Parameters PHY-Register	Register for all parameters of the MIB PHY
0x08	Parameters Lane-Register	Register for all parameters of the MIB Lane
0x0C	Status Lane-Register	Register for all status of the MIB Lane
0x10	Parameters Data Link Register	Register for all parameters of the MIB Data Link
0x14	Status 1 Data Link Register	First register for all status of the MIB Data Link
0x18	Status 2 Data Link Register	Second register for all status of the MIB Data Link
0x1C	QoS 1 Data Link Register	First register for all QoS information of the Data Link
0x20	QoS 2 Data Link Register	Second register for all QoS information of the Data Link
0x24	Error Management Data Link Register	Register for all necessary information for error management of the Data Link
0x28-0xFF	Reserved	

Table 20: Register Map

Bits	**Name**	**Core Access**	**Reset Value**	**Description**

0	rst_dut_n	Read/write	0	Reset the DUT (low active)
1	lane_spy_en	Read/write	0	Enable the lane spy
2	data_link_disable	Read/write	0	Enable the lane injector
3-31	Reserved			

Table 21: Global Register Description

Bits	**Name**	**Core Access**	**Reset Value**	**Description**
0	Near-End Serial Loopback	Read/write	0	Enables or disables the near-end serial loopback for the corresponding lane
1	Far-End Serial Loopback	Read/write	0	Enables or disables the far-end serial loopback for the corresponding lane
2-31	Reserved			

Table 22: Parameter-PHY Register Description

Bits	**Name**	**Core Access**	**Reset Value**	**Description**
0	Lanestart	Read/write	0	SpaceFibre lane start initialisatiion signal
1	Autostart	Read/write	0	Enables communication lane to initialize automatically when a link is established
2	Lanereset	Read/write	0	Reset Lane layer signal
3	Parallel loopback enables	Read/write	0	Parallel loopback enables signal
4-11	Standbyreason	Read/write	0	Standby reason field
12	Lanestart pulse	Read/write	0	Lane start signal generated via a pulse
13-31	Reserved			

Table 23: Parameter Lane Register Description

Bits	Name	Core Access	Reset Value	Description
0-3	Lane state	Read Only	0	Indicates current channel status
4-11	RX Error Counter	Read Only	0	Counter of reception errors detected
12	RX Error Overflow	Read Only	0	Reception error counter overflow indicator
13	Far-end lost Signal	Read Only	0	Indicates loss of signal at remote end
14-21	Far-end Capabilities	Read Only	0	Far-end capacity
22	RX Polarity	Read Only	0	Indicates reception signal polarity (normal or inverted)
23-31	Reserved			

Table 24: Status Lane Register Description

Bits	Name	Core Access	Reset Value	Description
0	Interface Reset	Read/write	0	Reset the link and all configuration register of the Data Link layer
1	Link Reset	Read/write	0	Reset the link
2	Link Reset Asserted	Read/write	0	Asserted when the link has been reseted, must be de-asserted manually
3	NACK_rst_en	Read/write	1	Enable automatic link reset on NACK reception
4	NACK_rst_mode	Read/write	1	Up for instant link reset on NACK reception, down for link reset at the end of the current received frame on NACK reception
5-13	Pause VC	Read/write	0	Pause the corresponding virtual channel after the end of current transmission

14-21	Continuous VC	Read/write	0	Enable the corresponding virtual channel continuous mode
22	Reset Error Flag	Read/write	1	If asserted, de-assert itself and CRC long error, CRC short error, Frame error, Sequence error
22-31	Reserved			

Table 25: Parameter Data Link Register Description

Bits	Name	Core Access	Reset Value	Description
0-7	SEQ_NUMBER TX	Read Only	0	SEQ_NUMBER in transmission
8-15	SEQ_NUMBER RX	Read Only	0	SEQ_NUMBER in reception
16-23	Credit VC	Read Only	0	Indicates if each corresponding far-end input buffer has credit
24-31	FCT credit overflow	Read Only	0	Indicates overflow of each corresponding FCT credit counter

Table 26: Status 1 Data Link Register Description

Bits	Name	Core Access	Reset Value	Description
0	CRC long error	Read Only	0	Indicates that a CRC-16bits error was detected
1	CRC short error	Read Only	0	Indicates that a CRC-8bits error was detected
2	Frame error	Read Only	0	Indicates that a frame error was detected
3	Sequence error	Read Only	0	Indicates that a sequence error was detected
4	Far-end link reset	Read Only	0	Indicates the far-end link reset status

5-12	Input buffer overflow	Read Only	Read Only	Indicates that the corresponding input buffer has overflowed
13-31	Reserved			

Table 27: Status 2 Data Link Register Description

Bits	Name	Core Access	Reset Value	Description
0-8	Frame finished	Read Only	0	Indicates that corresponding channel finished emitting a frame
9-17	Frame TX	Read Only	0	Indicates that corresponding channel is emitting a frame
18-24	Data counter TX	Read Only	0	Indicate the number of data transmitted in last frame emitted
25-31	Data counter RX	Read Only	0	Indicate the number of data received in last frame received

Table 28: QoS 1 Data Link Register Description

Bits	Name	Core Access	Reset Value	Description
0-2	ACK counter TX	Read Only	0	Indicate the number of ACK control word transmitted during last frame emitted
3-5	NACK counter TX	Read Only	0	Indicate the number of NACK control word transmitted during last frame emitted
6-9	FCT counter TX	Read Only	0	Indicate the number of FCT control word transmitted during last frame emitted
10-12	ACK counter RX	Read Only	0	Indicate the number of ACK control word received during last frame received
13-15	NACK counter RX	Read Only	0	Indicate the number of NACK control word

				received during last frame received
16-19	FCT counter RX	Read Only	0	Indicate the number of FCT control word received during last frame received
20-21	FULL counter RX	Read Only	0	Indicate the number of FULL control word received during last frame received
22-23	RETRY counter RX	Read Only	0	Indicate the number of RETRY control word received during last frame received
24-31	current time slot	Read Only	0	Indicate the current time-slot for time -slot QoS

Table 29: QoS 2 Data Link Register Description

Bits	**Name**	**Core Access**	**Reset Value**	**Description**
0-7	SEQ_NUMBER ACK RX	Read Only	0	SEQ_NUMBER in last ACK received
8-15	SEQ_NUMBER NACK RX	Read Only	0	SEQ_NUMBER in last NACK received
16-31	Reserved			

Table 30: Error Management Data Link Register Description

3.2.4. DATA_LINK_GENERATOR MODEL

3.2.4.1. MODEL OVERVIEW

The data_link_generator model is used to communicate with the Data Link layer. It communicates with the RISC-V or emulator via an AXI4-lite bus and with the IP via discrete signals. Its role is to generate frames compatible with the data format of the lane layer. In this model, it is possible to configure the number of packets, the packet size, the inter-packet delay and the generation of the data (incremental, PRBS) This model is able to give a sequence number of a packet, generate data.

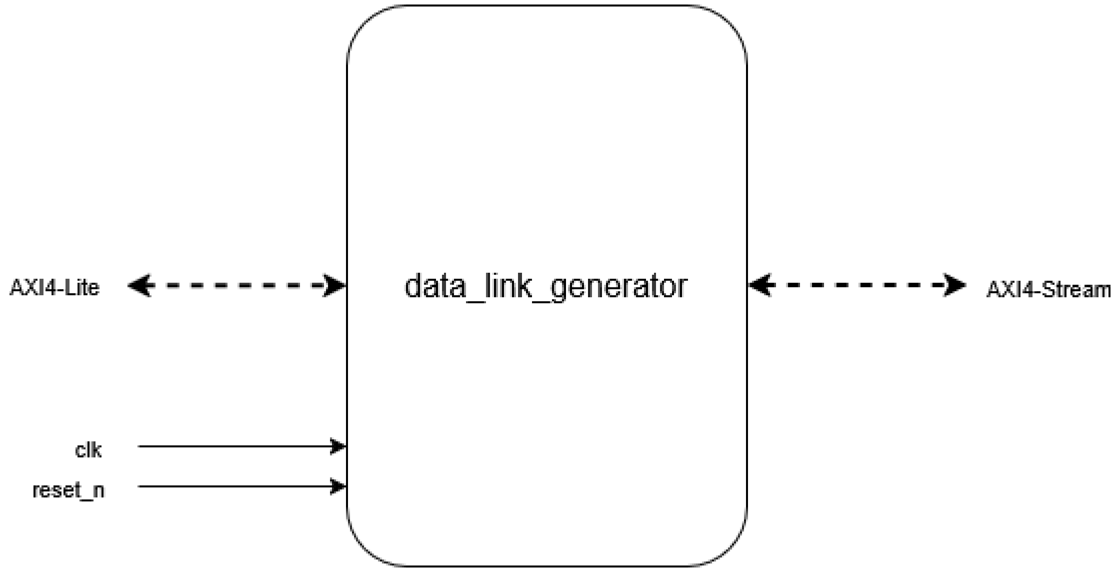

Figure 14: data_generator block diagram

3.2.4.2. FUNCTIONAL DESCRIPTION

3.2.4.2.1. FUNCTIONAL MODE

3.2.4.2.1.1. GENERATION DATA MODE

Figure 15 shows what the DUT will receive.

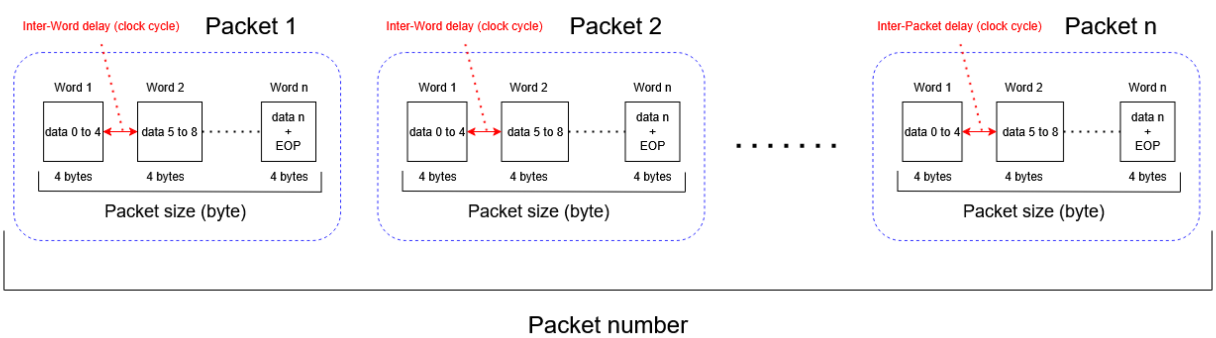

Figure 15: data generation flow diagram

Data are generated in two ways:

- Incremental: the initial value is configurable and is incremented by 1 for each word.
- PRBS: The seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

Data can be generated normally, or by being in a continuous mode, which discard the TREADY value of the AXI4-Stream interface with the data link.

3.2.4.2.2. PARAMETERS

Table 31 lists the parameters of the data_link_generator model. So that this module can be integrated into various systems, the width of the various AXI interface signals can be configured using parameters.

Name	Type	Description
ADDR_WIDTH	Positive integer	AXI address bus width
DATA_WIDTH	Positive integer	AXI data bus width

Table 31: data_link_generator model parameters

3.2.4.2.3. PORTS

Table 32 lists all the ports in the data_link_generator model. The model has a slave interface compliant with the AXI4-Lite standard.

The following bus corresponds to the data sent to the Data Link layer

Name	Direction	Dimensions	Description
Global signals			
Clk	Input	1	Main clock.
reset_n	Input	1	Reset low active
AXI4-Lite *slave* Port			
AXI4-lite	Inout	/	Table 2: Axi4-lite slave ports
Data Link interface			
AXI4-Stream Master	Inout	/	Table 3: Axi4-Stream ports

Table 32: data_link_generator model ports

3.2.4.2.4. REGISTER MAP

Table 33 lists the registers in the data_link_generator model:

- A configuration register with read and write access.

- A control register with read and write access.

- A status register with read only access.

- A initial value register with read and write access.

Address	Name	Description
0x00	Configuration Register	Register for all the parameters of lane_generator model
0x04	Control Register	Register for all the control signals of the lane_generator model
0x08	Status Register	Register for all the status signals of the lane_generator model
0x0C	Initial Value Register	Register for the initial value of the data generated
0x10-0xFF	Reserved	

Table 33: data_link_generator register model

Bits	Name	Core Access	Reset Value	Description
0-4	Packet number	Read/write	0	Number of frames to send
5-13	Packet size	Read/write	0	Size of the frame (byte)
14-23	Inter-packet delay	Read/write	0	Inter-packet delay
24	Generation data	Read/write	0	Type of generation of data: -0: Incremental -1: PRBS
25-26	Data mode	Read/write	0	00: Generation data 01: Continuous Generation 10: Reserved 11: Reserved
27-31	Reserved			

Table 34: data_link_generator configuration register description

Bits	Name	Core Access	Reset Value	Description
0	Model start	Read/write	0	Start generating data
1-31	Reserved			

Table 35: data_link_generator control model register description

Bits	Name	Core Access	Reset Value	Description
0	Busy	Read Only	0	Test in progress
1	Test End	Read Only	0	Test finished
2-9	Error counter	Read Only	0	Counter for number of errors, locks when at maximum
10-31	Reserved			

Table 36: data_link_generator status model register description

Bits	Name	Core Access	Reset Value	Description
0-31	Initial Value	Read/write	0	Initial value for the data

Table 37: data_link_generator initial value register description

3.2.5. DATA_LINK_ANALYZER MODEL

3.2.5.1. MODEL OVERVIEW

The data_link_analyzer model is used to communicate with the Data Link layer. It communicates with the RISC-V or emulator via an AXI4-lite bus and with the IP via discrete signals. Its role is to receive frames from the Data Link layer. In this model, it is possible to configure the number of packets that he has to receive, the packet size, the inter-packet delay and the type of the data (incremental, PRBS) This model is able to receive Data Link layer packets and identify whether they are correct according to the configuration of its registers.

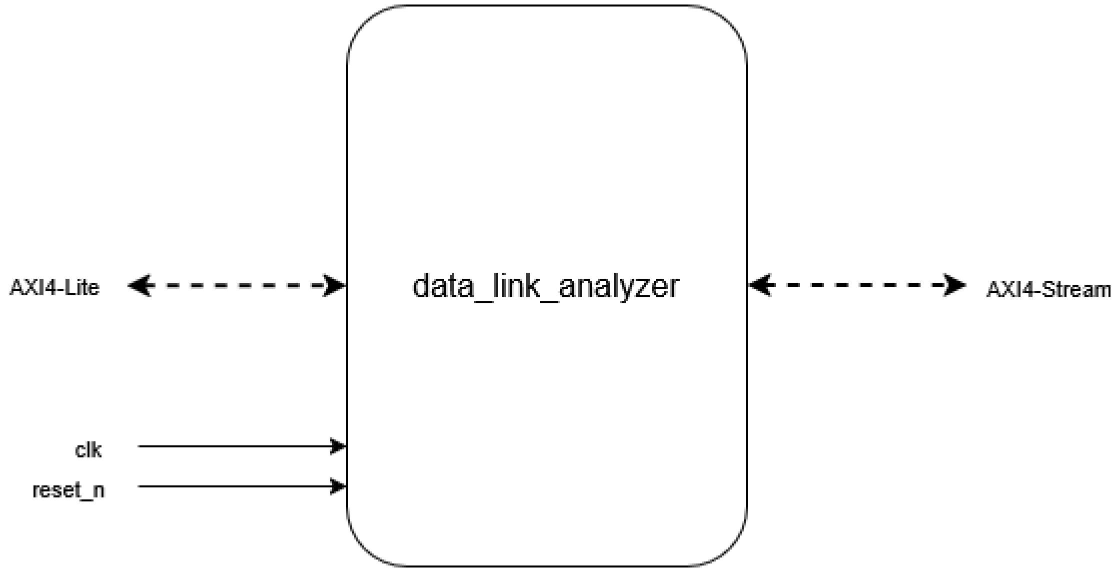

Figure 16: data_link_analyzer block diagram

3.2.5.2. FUNCTIONAL DESCRIPTION

3.2.5.2.1. FUNCTIONAL MODE

3.2.5.2.1.1. ANALYZE DATA MODE

Figure 17 shows what model must receive.

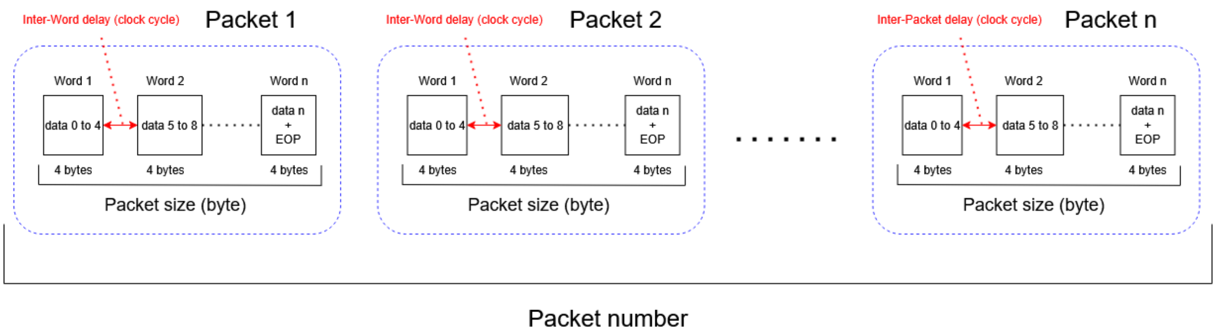

Figure 17: data for analyze flow diagram

Data are generated in two ways:

- Incremental: the initial value is configurable and is incremented by 1 for each packet.
- PRBS: The seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

If a data received contains an EEP character, the data will be ignored.

3.2.5.2.2. PARAMETERS

Table 38 lists the parameters of the lane_analyzer model. So that this module can be integrated into various systems, the widths of the various AXI interface signals can be configured using parameters.

Name	Type	Description
ADDR_WIDTH	Positive integer	AXI address bus width
DATA_WIDTH	Positive integer	AXI data bus width

Table 38: data_link_analyzer model parameters

3.2.5.2.3. PORTS

Table 39 lists all the ports in the data_link_analyzer model. The model has a slave interface compliant with the AXI4-Lite standard.

The following bus corresponds to the data received from the Data Link layer

Name	Direction	Dimensions	Description
Global signals			
Clk	Input	1	Main clock.
reset_n	Input	1	Reset low active
AXI4-Lite *slave* Port			
AXI4-lite	Inout	/	Table 2: Axi4-lite slave ports

Data Link interface			
AXI4-Stream Slave	Inout	/	Table 3: Axi4-Stream ports

Table 39: data_link_analyzer model ports

3.2.5.2.4. REGISTER MAP

Table 40 lists the registers in the data_link_analyzer model:

- A configuration register with read and write access.
- A control register with read and write access.
- A status register with read only access.

Address	Name	Description
0x00	Configuration Register	Register for all parameters of the data_link_analyzer model
0x04	Control Register	Register for all the control signals of the data_link_analyzer model
0x08	Status Register	Register for all the status signals of the data_link_analyzer model
0x0C	Initial Value Register	Register for the initial value of the data generated
0x10-0xFF	Reserved	

Table 40: data_link_analyzer registers model

Bits	Name	Core Access	Reset Value	Description
0-4	Packet number	Read/write	0	Number of packets to send
5-13	Packet size	Read/write	0	Size of the packet (byte)
14-23	Inter-packet delay	Read/write	0	Inter-packet delay
24	Generation data	Read/write	0	Type of generation of data: -0: incremental -1: PRBS
25-26	Data mode	Read/write	0	00: Generation data 01: Reserved

				10: Reserved 11: Reserved
27-31	Reserved			

Table 41: data_link_analyzer configuration register description

Bits	Name	Core Access	Reset Value	Description
0	Model start	Read/write	0	Start generating data
1-31	Reserved			

Table 42: data_link_analyzer control register description

Bits	Name	Core Access	Reset Value	Description
0	Busy	Read Only	0	Test in progress
1	Test End	Read Only	0	Test finished
2-9	Error counter	Read Only	0	Counter for number of errors, locks when at maximum
10-31	Reserved			

Table 43: data_link_analyzer status register description

Bits	Name	Core Access	Reset Value	Description
0-31	Initial Value	Read/write	0	Initial value for the data

Table 44: data_link_analyzer initial value register description

3.3. PYTHON MODELS (VIRTUAL TB)

3.3.1. LOOPBACK MODEL

3.3.1.1. MODEL OVERVIEW

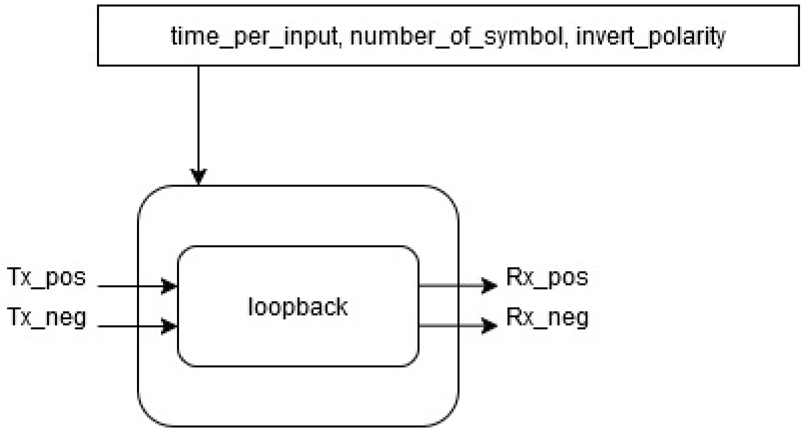

Figure 18: Loopback block diagram

The Loopback model is a class used to redirect the data transmitted by the Tx port of the IP to the Rx port of the IP. It is developed in python using cocotb library. The number of symbols of data to redirect, and the polarity can be configured. The Loopback model is instantiated in the TB class, where the period of the clock can be configured.

3.3.1.2. FUNCTIONAL DESCRIPTION

3.3.1.2.1. PARAMETERS

The Table 45 lists the parameters to configure the Loopback model.

Name	Type	Description
time_per_input	Positive integer	Period of each bits on the serial connection, cannot be 0
number_of_symbol	Positive integer	Number of symbols of 10 bits to be redirected
invert_polarity	Boolean	Flag of polarity, 1 for inverted, 0 by default

Table 45: Loopback model parameters

3.3.2. PYTHON SPACEFIBRE_SINK MODEL

3.3.2.1. MODEL OVERVIEW

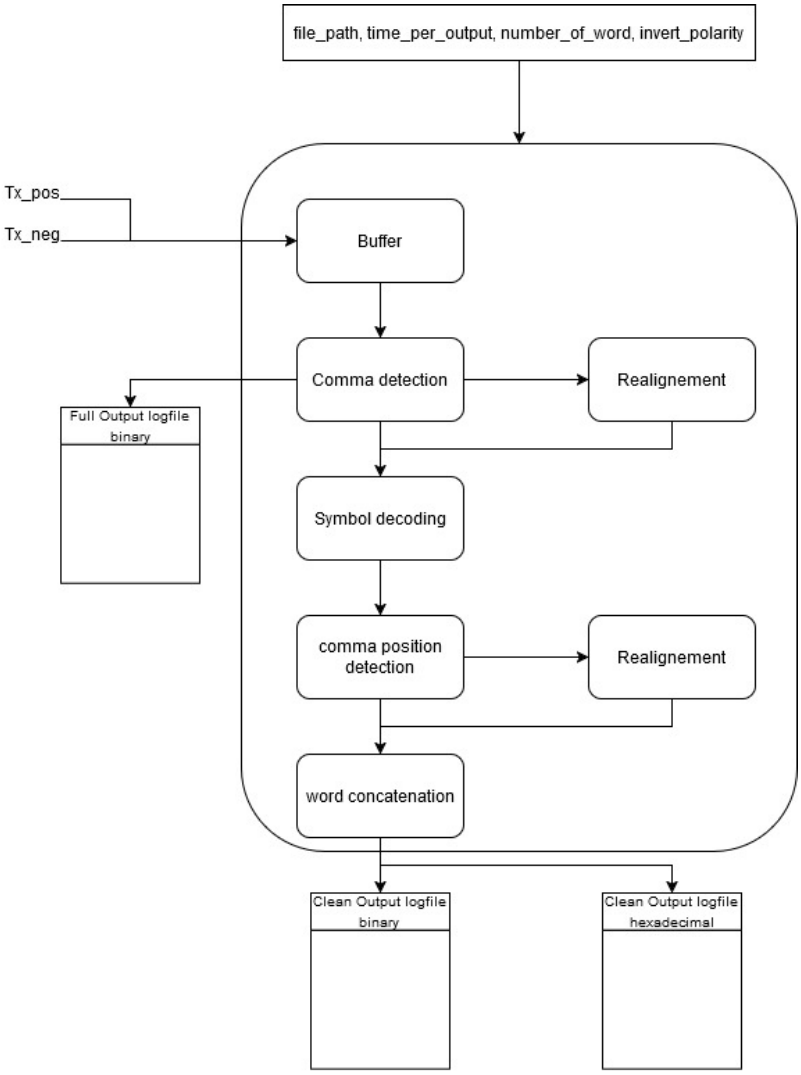

Figure 19: Spacefibre_Sink block diagram

The Spacefibre_Sink model is a class used to capture the data transmitted via the Tx port of the IP. The model can detect the commas to realign the symbol reading, decode the symbol using the 8b/10b table, and realign the word reading using the commas. It generates three files:

-the full output logfile, which will record every bit transmitted in binary format

-the clean output logfile in binary format, which will record every valid symbol transmitted in binary format

-the clean output logfile in hexadecimal format, which will record every valid symbol transmitted in hexadecimal format

These three files are created if they don't exist or are open in append mode. The path to the three files, the period of each bit on the serial link and the number of valid words to be read can be configured. The Spacefibre_Sink model is instantiated in the TB class.

3.3.2.2. FUNCTIONAL DESCRIPTION

3.3.2.2.1. PARAMETERS

The Table 46 lists the parameters to configure the SpaceFibre_Sink model

Name	Type	Description
file_path	String	Path to the directory to stock the three files, including a prefix common to all files
time_per_input	Positive integer	Period of each bits on the serial connection, cannot be 0
number_of_word	Positive integer	Number of 32 bits word to be recorded
invert_polarity	Boolean	Flag of polarity, 1 for inverted, 0 by default

Table 46: Spacefibre_Sink parameters

3.3.2.2.2. OUTPUTS

The Table 47 lists the different files generated by the SpaceFibre_Sink model. The clean output logfiles use the format described below:

"word;D/K_flag_per_byte;time_per_input; realignment_flag"

Example:

```
05E7E3F0;0000;10;0
85F3D0BF;0000;10;0
AEA50B5E;0000;10;0
10528A75;0000;10;0
3D78BA97;0000;10;0
D0A0BA28;0000;10;0
77309E05;0000;10;0
6D6710D2;0000;10;0
A3845DE8;0000;10;0
1CA0BEEF;1000;10;0
1CA0BEEF;0000;10;0
FE08DEFD;1001;10;0
26;0;10;0
ADA685FC;0001;10;1
ADBEEFBE;0000;10;0
```

Figure 20: Clean output logfile format example

Name	Path	Description
full output logfile binary	file_path + "_10b.dat"	File recording every bit read by the model. Those bits are recorded in the order they are received by the

		model, one line per word or partial word.
clean output logfile binary	file_path + "_clean_bin.dat"	File recording every valid symbol read by the model. They are concatenated to form valid or partial words. Each line of the file records one word in binary format.
clean output logfile hexa	file_path + "_clean_hexa.dat"	File recording every valid symbol read by the model. They are concatenated to form valid or partial words. Each line of the file records one word in hexadecimal format.

Table 47: SpaceFibre_Sink output files

The Table 48 specify each part of the output logfiles format.

Name	Format	Description
word	Binary (8b or 10b) or hexadecimal	Store the data that has been received.
D/K_flag_per_byte	Up to 4 booleans	Each boolean indicate if the respective byte was encoded as a D symbol or K symbol. It is 1 for K symbol and 0 for D symbol
time_per_input	Positive integer	Period of time associated to each bits, in nanosecond
realignment_flag	Boolean	It is 1 if a realignment has been performed before receiving the data.

Table 48: Clean output logfile format

3.3.1. PYTHON SPACEFIBRE_RANDOM_GENERATOR MODEL

3.3.1.1. MODEL OVERVIEW

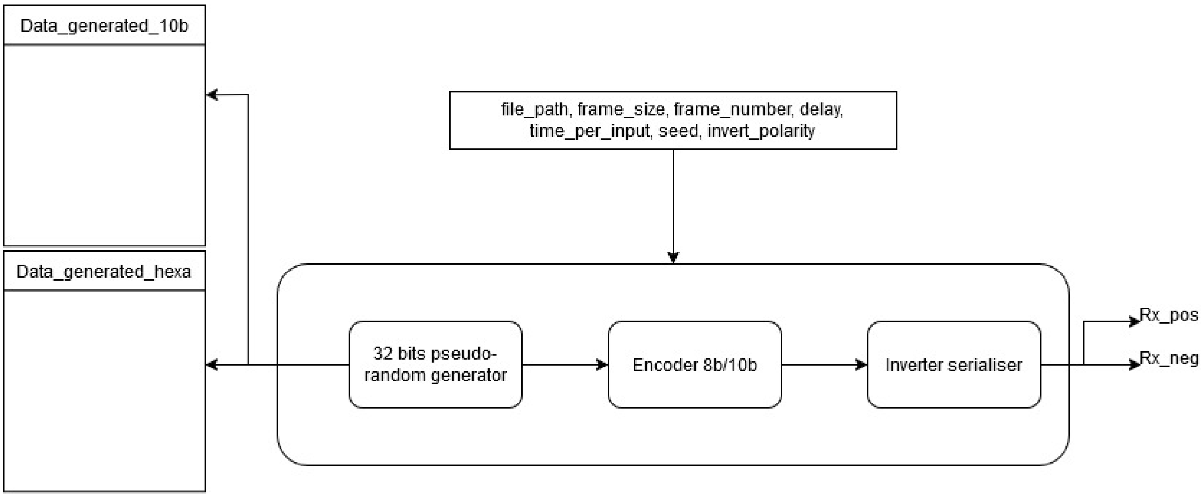

Figure 21: SpaceFibre_Random_Generator block diagram

The SpaceFibre_Random_Generator is a class used to generate pseudo-random symbols and to transmit them to the Rx port of the IP. The random generator generates frames of 32bits pseudo-random data. The Figure 22 describes the structure of the frames generated:

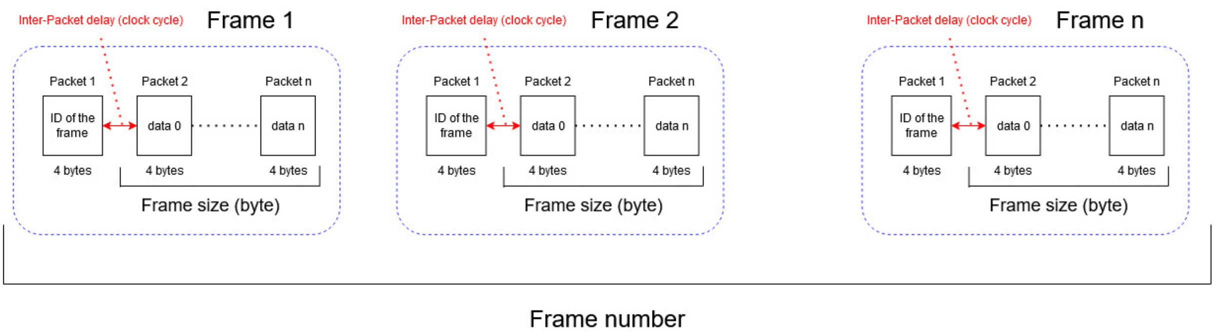

Figure 22 : Python random generator data generation

The frame ID corresponds to the frame number.

Data are generated using a polynomial, the seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

Each data is recorded into the generated data logfile. It is then encoded using a 8b/10b encoder and is then transmitted to the Rx port of the IP. Every 5000 words of 32 bits, a SKIP control word (D31.3, D31.3, D14.6, K28.7) is inserted, and before and after a generation of frames, ten IDLE control word (D15.6, D15.6, D14.6, K28.7) are inserted.

The file path of the generated data logfile, the frame size, the number of frames to be generated, the seed of pseudo-random generator, an initial delay and the period for each bit on the serial link of the IP can be configured.

3.3.1.2. FUNCTIONAL DESCRIPTION

3.3.1.2.1. PARAMETERS

The **Erreur ! Source du renvoi introuvable.** lists the different parameters of the S paceFibre_Random_Generator.

Name	Type	Description
file_path	String	Path to the generated data logfile which will record the data generated.
time_per_input	Positive integer	Period of each bits on the serial connection, cannot be 0.
delay	Positive integer	Initial delay to be waited before transmitting the first data.
seed	Positive integer	Seed of the random generator. It is set at 42 by default.
frame_size	Positive integer	Number of 8 bits data to be generated in a frame.
frame_number	Positive integer	Number of frame to be generated
invert_polarity	Boolean	Flag of polarity, 1 for inverted, 0 by default

Table 49: SpaceFibre_Random_Generator parameters

3.3.1.2.2. OUTPUT

The Figure 23 presents the files generated by the SpaceFibre_Random_Generator model. The generated data logfile in hexadecimal use the format described below:

"data_size;data;delay;D/K_flag_per_byte"

Example:

```
32;CFCFCEFC;0;0001
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
```

Figure 23: Generated data logfile format example

The generated data logfile in 10bits encoding use the format described below:

“data_size;data;delay;D/K_flag_per_byte;simulation_time”

```
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331483302045.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331489968695.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331496635365.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331503302035.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331509968715.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331516635385.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331523302065.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331529968725.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331536635385.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331543302045.0
```

Figure 24 : Generated data logfile format 10b example

Name	Path	Description
generated data logfile	file_path	File recording every data generated by the model. Those data are recorded in the order they are transmitted by the model, one line per word of 32 bits.
generated data logfile	file_path + “_10b	File recording every data generated by the model. Those data are recorded in the order they are transmitted by the model, one line per word of 40 bits.

Table 50: SpaceFibre_Random_Generator output file

The Table 51 specify the different parts of the generated data logfile format. This format is similar to the one used in the inputs to be sent file of the SpaceFibre_Driver.

Name	Format	Description
data_size	Positive integer	Size of the data generated, fixed to 32.
data	Hexadecimal or binary(10b)	Store the data that has been transmitted.
delay	Positive integer	Initial delay awaited before performing the transmition of data, fixed to 0.
time_per_input	Positive integer	Period of time associated to each bits, in nanosecond
D/K_flag_per_byte	4 booleans	Each boolean indicate if the respective byte was encoded as a D symbol or K symbol. It is 1 for K symbol and 0 for D symbol.
Simulation_time	Positive integer	Only in the 10b logfile, indicate the simulation time of generation and transmission of data, in femtoseconds.

Table 51: Generated data logfile format

3.3.2. PYTHON SPACEFIBRE_RANDOM_GENERATOR_DATA_LINK MODEL

3.3.2.1. MODEL OVERVIEW

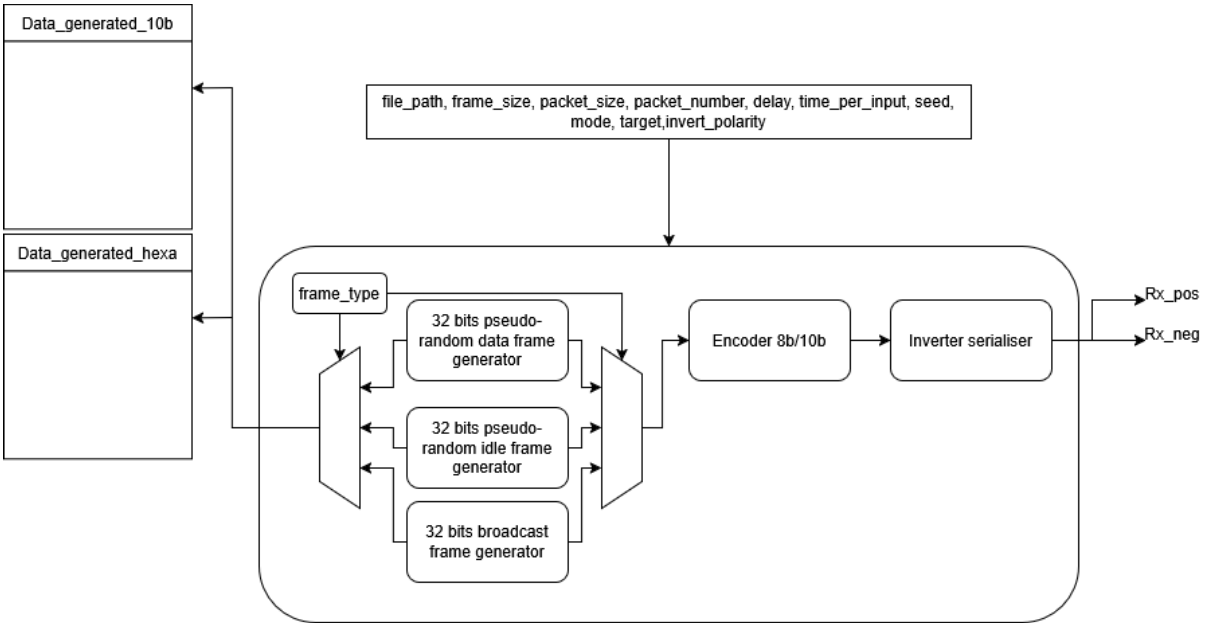

Figure 25: SpaceFibre_Random_Generator block diagram

The SpaceFibre_Random_Generator is a class used to generate pseudo-random symbols and to transmit them to the Rx port of the IP. The random generator can generate different type of frame: data frama, broadcast frame or idle frame.

Each word transmitted is recorded into the generated data logfile. It is then encoded using a 8b/10b encoder and is then transmitted to the Rx port of the IP. Every 5000 words of 32 bits, a SKIP control word (D31.3, D31.3, D14.6, K28.7) is inserted, and before and after a generation of frames, ten IDLE control word (D15.6, D15.6, D14.6, K28.7) are inserted.

The file path of the generated data logfile, the frame size, the packet size, the number of packet to be generated, the seed of pseudo-random generator, the type of frame to be generated, the target of the transmission, the sequence number, an initial delay and the period for each bit on the serial link of the IP can be configured.

When generating data frame generates packets of 32bits pseudo-random data. The Figure 26 describes the structure of the packets generated:

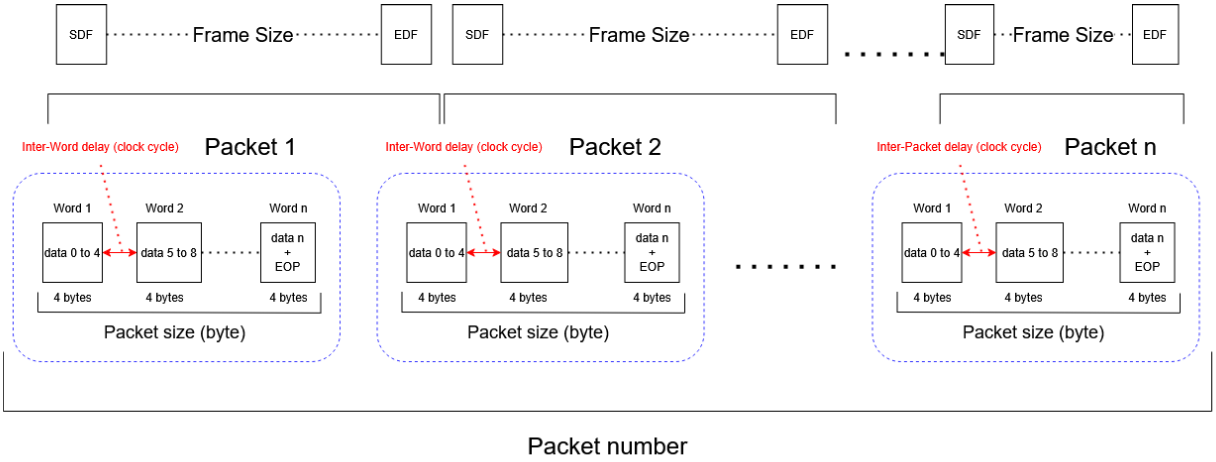

Figure 26 : Python random generator data frame generation

The packets end with a EOP Char. The packets are distributed into frames of fixed size. The last frame size can be reduced to adapt to the number of data to be generated. If the last data word of the last frame is not aligned on 32 bits, FILL Char can be inserted to realign the data.

Data are generated using a polynomial, the seed is configurable and the polynomial is $P(x) = x^{31} + x^{30} + x^{28} + x^{27} + 1$

When generating data frames, each frame starts with SDF and end with EDF. SEQ_NUMBER is incremented after each frame transmission, and the CRC-16bits is computed with the following ITU polynomial, with the seed 0xFFFF, applied to the whole frame including SDF and EDF:

$$P(x) = x^{16} + x^{12} + x^5 + 1$$

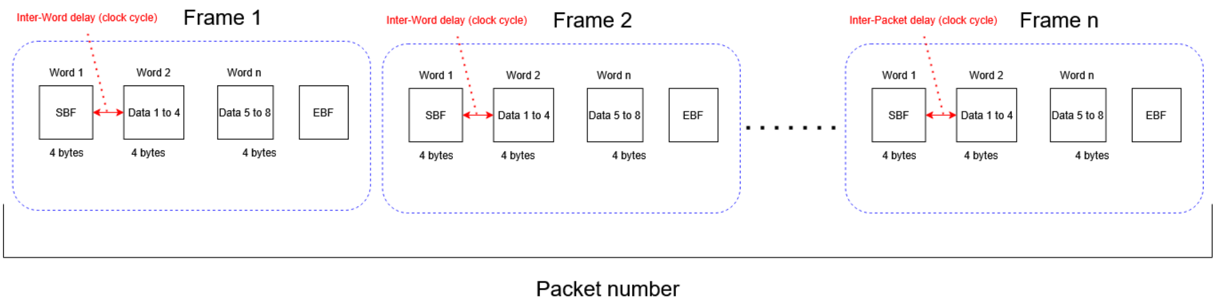

Figure 27 : Python random generator broadcast frame generation

When generating a broadcast frame, the size of the frame is set to 2 words. The packet number represents then the number of frame generated. If frame size is specified, the value is used for the data of the broadcast frame, else the value is set to 0x0000_FFFF_DEAD_BEEF. Each frame starts with a SBF() and end with a EBF(). SEQ_NUMBER is incremented after each frame transmission, and the CRC-8bits is computed with the algorithm described in [REF_01] in [SPACEFIBRE_LIGHT_IP -04. 0550], applied to the whole frame including SBF and EBF.

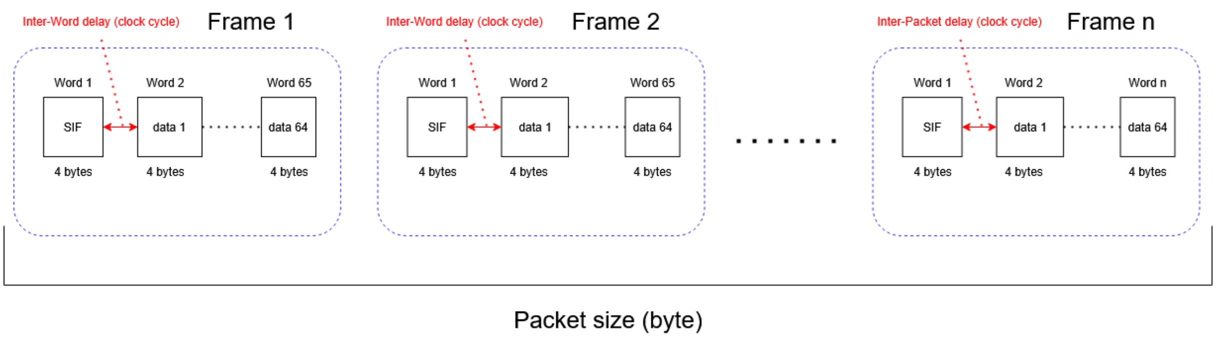

Figure 28 : Python random generator data frame generation

When generating an idle frame, the PRBS algorithm used for data generation use a polynomial, with the seed being configurable and the polynomial being $P(x) = x^{16} + x^5 + x^4 + x^3 + 1$

The packet size represents the amount of data to be generated, and frame size is set to 64 words. Each frame starts with a SIF(). SEQ_NUMBER is fixed for all frame transmission, and the CRC-8bits is computed with the algorithm described in [REF_01] in [SPACEFIBRE_LIGHT_IP -04. 0550], applied to the first three Char of the SIF:

3.3.2.2. FUNCTIONAL DESCRIPTION

3.3.2.2.1. PARAMETERS

The Table 52 lists the different parameters of the SpaceFibre_Random_Generator.

Name	Type	Description
file_path	String	Path to the generated data logfile which will record the data generated.
time_per_input	Positive integer	Period of each bits on the serial connection, cannot be 0.
delay	Positive integer	Initial delay to be waited before transmitting the first data.
seed	Positive integer	Seed of the random generator. It is set at 42 by default.
frame_type	Positive integer	Type of frame generated, 0 for data frame, 1 for broadcast frame, 2 for idle frame
frame_size	Positive integer	Number of 32 bits data to be generated in a frame.
packet_number	Positive integer	Number of packet to be generated
packet_size	Positive integer	Number of 8 bits data to be generated in a packet.
target	Positive integer	Virtual channel targeted by the transmission
sequence	Positive integer	Sequence number of the first transmission
invert_polarity	Boolean	Flag of polarity, 1 for inverted, 0 by default

Table 52: SpaceFibre_Random_Generator parameters

3.3.2.2.2. OUTPUT

The Table 53presents the files generated by the SpaceFibre_Random_Generator model. The generated data logfile in hexadecimal use the format described below:

"data_size;data;delay;D/K_flag_per_byte"

Example:

```
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
32;CFCFCEFC;0;0001;
```

Figure 29: Generated data logfile format example

The generated data logfile in 10bits encoding use the format described below:

"data_size;data;delay;D/K_flag_per_byte;simulation_time"

```
32;1010000110 0101110110 0111000110 0011111000 ;0;0001;331483302045.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331489968695.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331496635365.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331503302035.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331509968715.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331516635385.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331523302065.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331529968725.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331536635385.0
32;1010000110_0101110110_0111000110_0011111000_;0;0001;331543302045.0
```

Figure 30 : Generated data logfile format 10b example

Name	Path	Description
generated data logfile	file_path	File recording every data generated by the model. Those data are recorded in the order they are transmitted by the model, one line per word of 32 bits.
generated data logfile	file_path + "_10b	File recording every data generated by the model. Those data are recorded in the order they are transmitted by the model, one line per word of 40 bits.

Table 53: SpaceFibre_Random_Generator output file

The Table 54 specify the different parts of the generated data logfile format. This format is similar to the one used in the inputs to be sent file of the SpaceFibre_Driver.

Name	Format	Description
data_size	Positive integer	Size of the data generated, fixed to 32.
data	Hexadecimal or binary(10b)	Store the data that has been transmitted.
delay	Positive integer	Initial delay awaited before performing the transmition of data, fixed to 0.
time_per_input	Positive integer	Period of time associated to each bits, in nanosecond
D/K_flag_per_byte	4 booleans	Each boolean indicate if the respective byte was encoded as a D symbol or K symbol. It is 1 for K symbol and 0 for D symbol.
Simulation_time	Positive integer	Only in the 10b logfile, indicate the simulation time of generation and transmission of data, in femtoseconds.

Table 54: Generated data logfile format

3.3.3. PYTHON SPACEFIBRE_DRIVER MODEL

3.3.3.1. MODEL OVERVIEW

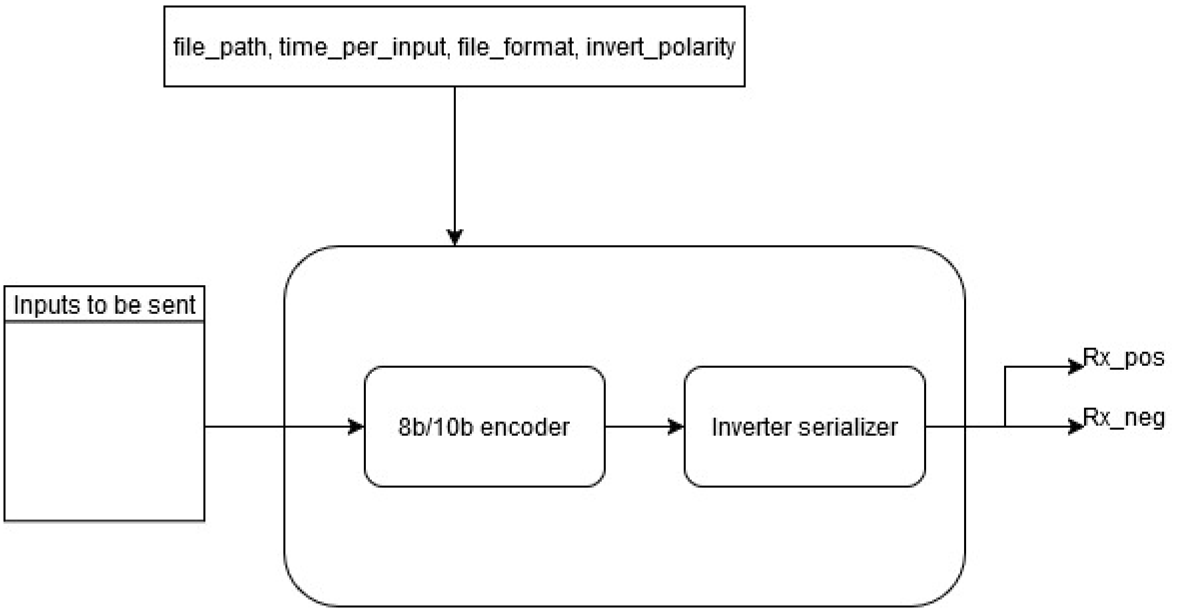

Figure 31: SpaceFibre_Driver block diagram

The SpaceFibre_Driver is a class used to transmit data from a file to the Rx port of the IP. The data is encoded with an 8b/10b encoder before being serialized and transmitted to the Rx port.

Every 5000 words of 32 bits, a SKIP control word (D31.3, D31.3, D14.6, K28.7) is inserted, and before and after a generation of frames, ten IDLE control word (D15.6, D15.6, D14.6, K28.7) are inserted.

The input file path, the input file format and the polarity on the serial link can be configured. The SpaceFibre_Driver is instantiated in the TB class, where the period of each bit can be configured.

3.3.3.2. FUNCTIONAL DESCRIPTION

3.3.3.2.1. PARAMETERS

The Table 55 lists the parameters to configure the SpaceFibre_Driver model.

Name	Type	Description
file_path	String	Path to the inputs to be sent file.
time_per_input	Positive integer	Period of each bit on the serial connection, cannot be 0.
file_format	Positive integer	Indicate the format of the data in the inputs to be sent, either 2 for binary or 16 for hexadecimal.
invert_polarity	Boolean	Flag of polarity, 1 for inverted, 0 by default

Table 55: SpaceFibre_Driver model parameters

3.3.3.2.2. INPUT

The Table 56 presents the inputs to be sent file used by the SpaceFibre_Driver model. The inputs to be sent file use the format described below:

“data_size;data;D/K_flag_per_byte;time_per_input”

Example:

```
32;1CA0BEEF;0;1000
32;1CA0BEEF;0;0000
32;FE08DEFD;0;1001
32;A685FC26;0;0010
8;AD;0;0
8;BE;0;0
32;DEADBEEF;0;0000
32;BED0FEED;0;0000
32;3C423696;0;1000
32;B7BC69FC;0;0101
8;FC;0;0
8;AD;0;0
8;AC;0;0
32;FCACDCCF;1;1000
32;DEADBEEF;0;0000
```

Figure 32: Inputs to be sent file format example

Name	Path	Description
Inputs to be sent file	file_path	File storing data to be sent by the model. Those data are stored in the order they will be transmitted by the model.

Table 56: SpaceFibre_Driver input file

The Table 57 specify the different parts of the inputs to be sent file format.

Name	Format	Description
data_size	Positive integer	Indicate the data size in bits. Either 8, 32 or 64.
data	Binary or hexadecimal	Store the data that has been transmitted.
D/K_flag_per_byte	Up to 8 booleans	Each boolean indicate if the respective byte was encoded as a D symbol or K symbol. It is 1 for K symbol and 0 for D symbol
time_per_input	Positive integer	Period of time associated to each bits, in nanosecond

Table 57: Inputs to be sent file format

End of document