

LOGICAL LAYER

Verification Plan

**Submitted by:**

**Karim Samy**

**Ali Ahmed**

**Walid Salah**

**Mohamed Salah**

**Submitted to:**

**Eng. Mohamed Samy**

**Eng. Mohamed Rayan**

**Eng. Randa Aboudief**

**Eng. Ahmed Mohamed**

**Eng. Sarah Mostafa**

**Date: 31/01/2024**

Table of Contents

[Introduction: 1](#_Toc166282859)

[Verification Environment: 2](#_Toc166282860)

[Test Environment: 2](#_Toc166282861)

[System Verilog Mailboxes: 3](#_Toc166282862)

[Stimulus Generator: 4](#_Toc166282863)

[Driver: 4](#_Toc166282864)

[Monitor: 5](#_Toc166282865)

[Scoreboard: 5](#_Toc166282866)

[Reference Model: 6](#_Toc166282867)

[Virtual Sequence: 6](#_Toc166282868)

[Transaction Flow: 9](#_Toc166282869)

[AI Reference Model: 13](#_Toc166282870)

[Classes 13](#_Toc166282871)

[1. Configuration Space Class: 13](#_Toc166282872)

[2. Upper Layer Class: 14](#_Toc166282873)

[3. Electrical Layer Class: 14](#_Toc166282874)

[Design Limitations: 19](#_Toc166282875)

[SYSTEM Level VERIFICATION: 20](#_Toc166282876)

[Interfaces: 20](#_Toc166282877)

[Interface 1 ports (Upper Layer Interface): 21](#_Toc166282878)

[Interface 2 Ports (Electrical Layer Interface): 22](#_Toc166282879)

[Interface 3 Ports(Configuration Space Interface): 23](#_Toc166282880)

[ENVIRONMENTS TRANSACTIONS: 23](#_Toc166282881)

[Transport Layer Transaction 23](#_Toc166282882)

[Configuration Spaces Transaction 24](#_Toc166282883)

[Electrical Layer transactions: 24](#_Toc166282884)

[USB4 Logical Layer Flow 27](#_Toc166282885)

[Test Scenarios 31](#_Toc166282886)

[Task Breakdown: 33](#_Toc166282887)

[Duration of each task (by hand): 35](#_Toc166282888)

# Introduction:

This document contains the full verification plan proposal for the Logical Layer Architecture. For this Design, we will make a Class-based Verification Environment to be able to test all design features in an efficient and organizable way. The Document flow will be as follows:

First, The Verification IP(VIP) Architecture will be presented along with a thorough description of the role of each component.

Next, full description of the transaction flow through the environment will be represented to visualize the operation of the Environment.

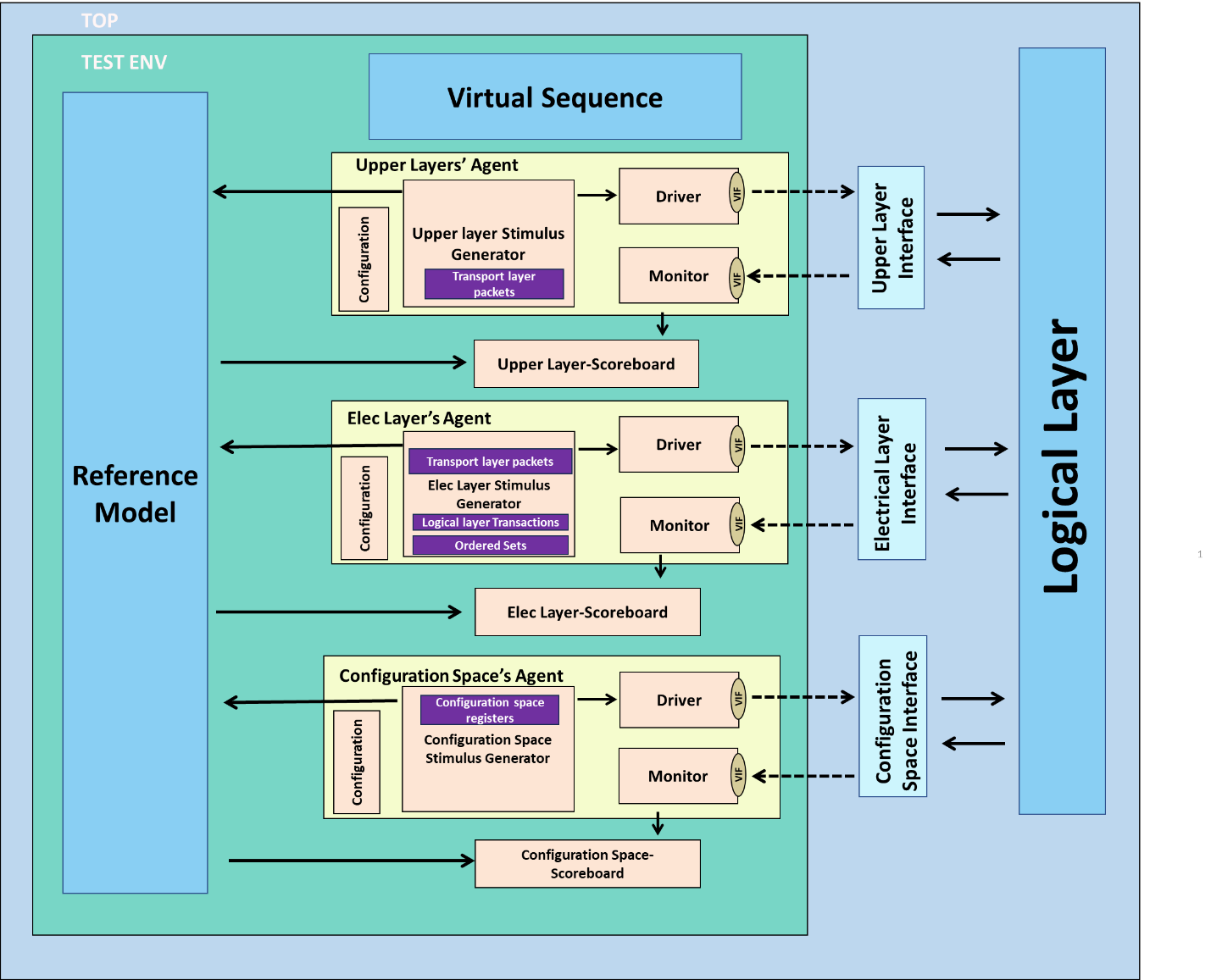
Moreover, Interfaces definition that connects the Environment to the Logical layer will be fully analyzed.

Furthermore, The Design Operational Flow along with the test Scenarios will be represented.

Finally, A traceability matrix will be constructed which is used to make sure that the created test scenarios actually test all the Design features.

Last Page will contain the Complete Task breakdown with the assigned tasks to each team member along with the expected time of completion.

# Verification Environment:

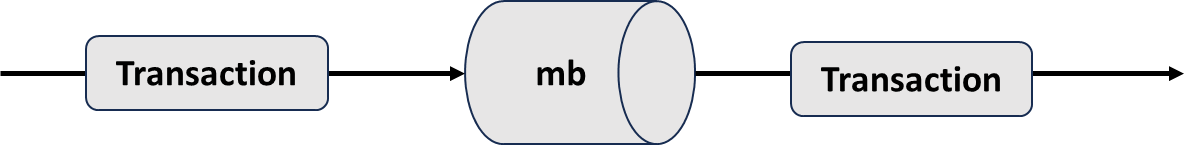


## Test Environment:

This class contains all the VIP components:

* Reference Model
* Virtual Sequencer
* 3 interfaces:
* Upper Layer Interface
* Electrical Layer Interface
* Configuration Spaces Interface
* 3 Agents for each Interface which includes:
* Stimulus Generator
* Driver
* Monitor
* 3 Scoreboard for the 3 Agents
* Mailboxes for communication
* S2d\_mb: connects the Stimulus Generator to the Driver
* Scr\_mod: Connects the Stimulus Generator to the Reference Model
* Log\_mon: Connects the Monitor to the Scoreboard
* Mod\_scr: Connects the Reference Model to the Scoreboard
* Sqcr\_agnt: Connects the Virtual Sequencer to the Agent

## System Verilog Mailboxes:



System Verilog Mailboxes enables the communication between the classes using 2 simple functions:

* .put(transaction): used to send transactions to the mailbox.
* .get(transaction): used to receive transactions from the mailbox.

The Mailboxes will be used to connect the operation of all mentioned components together to be able to perform the verification tasks in the correct order seamlessly to test the Logical layer in a fast and efficient way.

## Stimulus Generator:

It is the responsibility of the stimulus generator to create different test scenarios to test the required features of the design.

It combines all inputs into a packet/ transaction and sends the data to the driver via a mailbox when the driver is ready to send the next transaction through the Virtual interface.

Simulation ends when no more transactions are remaining to be sent by the stimulus generator.

## Driver:

**Driver**

**VIF**

A driver has the responsibility to generate the Pin level signals of the logical layer based on the transaction contents taken received from the Stimulus generator via a mailbox.

## Monitor:

**Monitor**

**VIF**

The main task for monitor is to capture any activity on the DUT signals (either inputs or outputs). The monitor assigns the pin level signal (from the interface) to the transaction/packet then send the transaction to the scoreboard via a mailbox.

## Scoreboard:

**Electrical Layer-Scoreboard**

**Electrical Layer-Scoreboard**

**Upper Layer-Scoreboard**

This is where the comparison between the DUT outputs and the golden reference is made to verify that the design works as expected.

We will need 3 Scoreboards where each scoreboard will receive a type of transaction based on the interface assigned to it.

Also, Functional Coverage takes place inside the Scoreboard in order to keep track of the progress of testing after each test scenario being applied. Based on the exit coverage criteria for each DUT, the Functional Coverage reports determine whether the design verification is done or not.

## Reference Model:

**Reference Model**

This is a golden reference that mimics the operation of the logical layer using a simpler, not necessarily synthesizable, model. This model takes its inputs from the 3 agents’ Stimulus generator.

The reference Model then delivers the Expected output to a Scoreboard via a mailbox connecting them together.

## Virtual Sequence:

**Virtual Sequence**

The Virtual Sequence synchronizes the operation between all 3 Stimulus generators in order to drive signals in the correct order from and to the Logical Layer.

Reference Model:

The primary objective of the reference model is to receive input from the generator and produce the expected output. This output is then compared with the actual output in the scoreboard to determine the presence of any errors.

A screenshot of a computer

Description automatically generated

*Architecture:*

* We want to check the input for each phase by getting the expected output of this phase to push this value into scoreboard.

A diagram of a circuit

Description automatically generated

A diagram of a flowchart

Description automatically generatedSBTX Controller:

Step1: read from configuration stimulus generator data and write it on side band memory and check SB\_TX output from controller be high.

Step2: wait transaction from electrical stimulus generator for AT transaction and check the correction then out from controller transaction response.

Step3: detect from electrical stimulus generator 16 order sets then out 2 order sets with respect to the type from controller selector and out it on the score board.

Step4: wait data from the transport stimulus generator electrical stimulus generator then forward it on score board to check it with output from DUT.

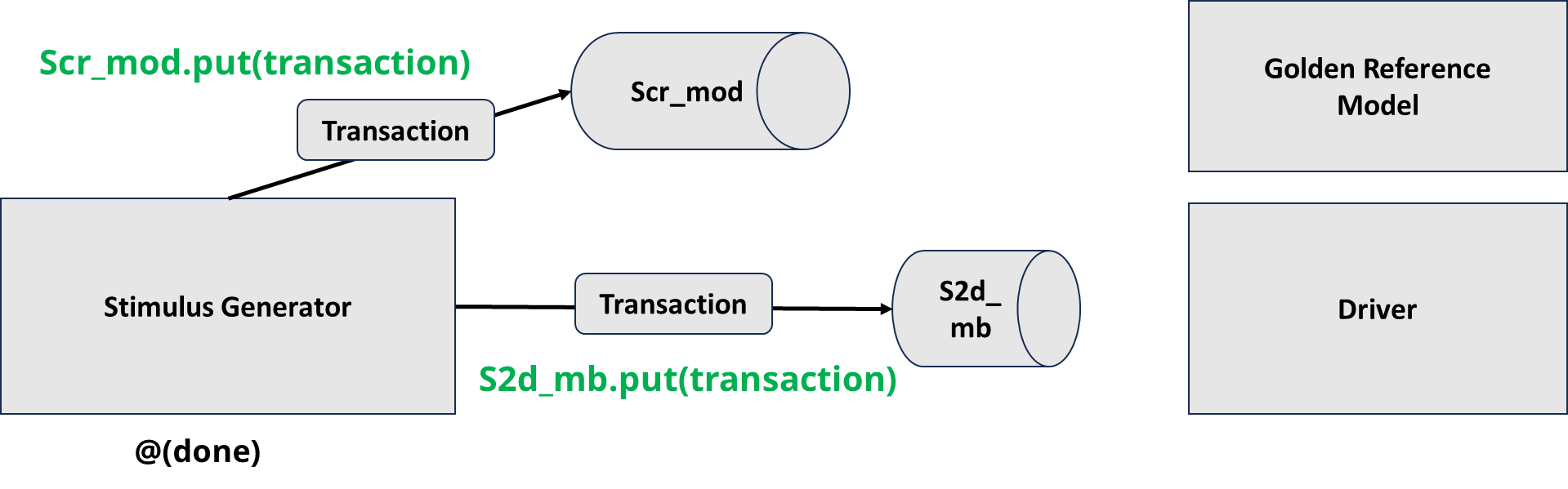
Notes:

- L\_disable bin acts as enable for data out on electrical score board throw mailbox.

-we can read or write from configuration stimulus generator data during transfer data form lane adapter.

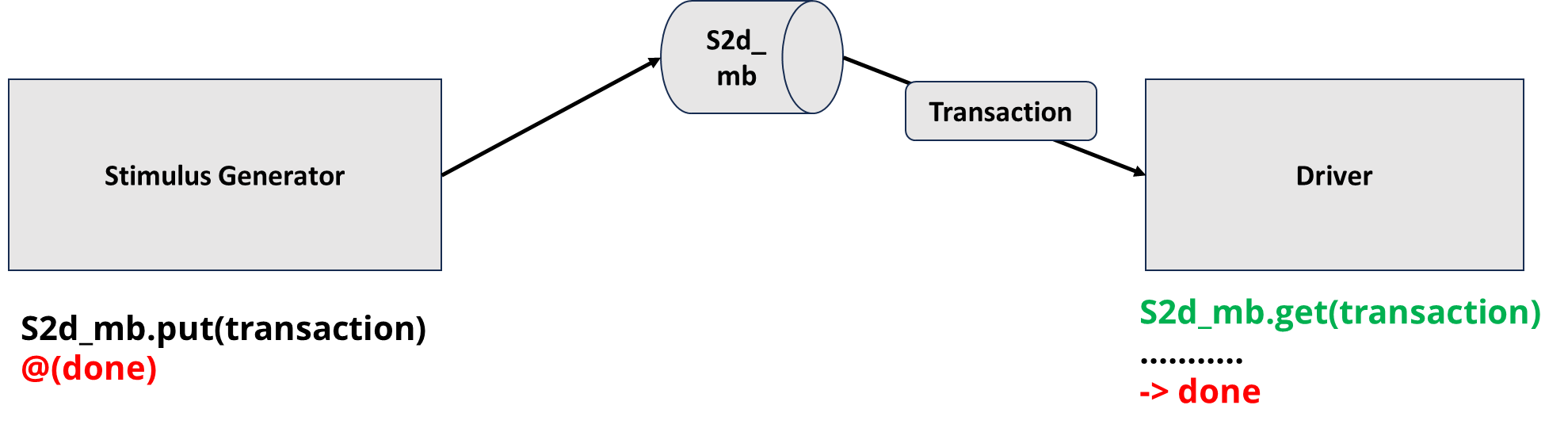
## Transaction Flow:

#### Step 1:



* The stimulus generator sends a scenario transaction to the **driver and Golden Reference’s** **mailbox**.
* The **.put method** is used to push data into the mailboxes.

#### Step 2:



* The driver obtains the stimulus from the mailbox using **.get()** function.
* The generator is **blocked** from sending any other data until the **event (done)** is triggered by the driver.

#### 

#### Step 3:

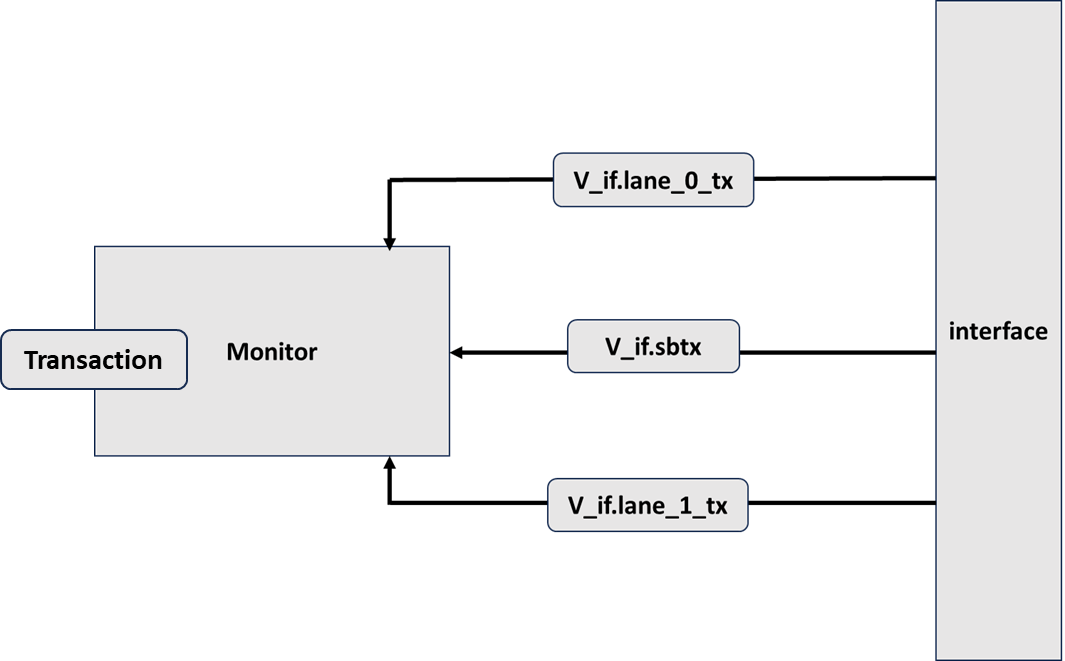
* According to the obtained transaction, the driver assigns data to the virtual interface every clock cycle.

#### Step 4:

#### 

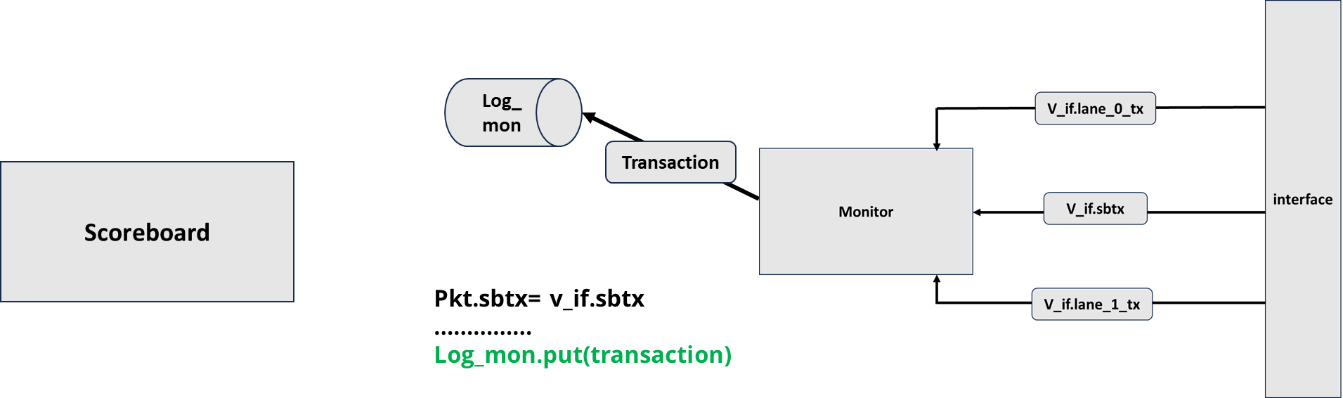
* Driver triggers the **event (done).**
* The stimulus generator is now **unblocked** and is free to send the **next** transaction.

#### Step 5:



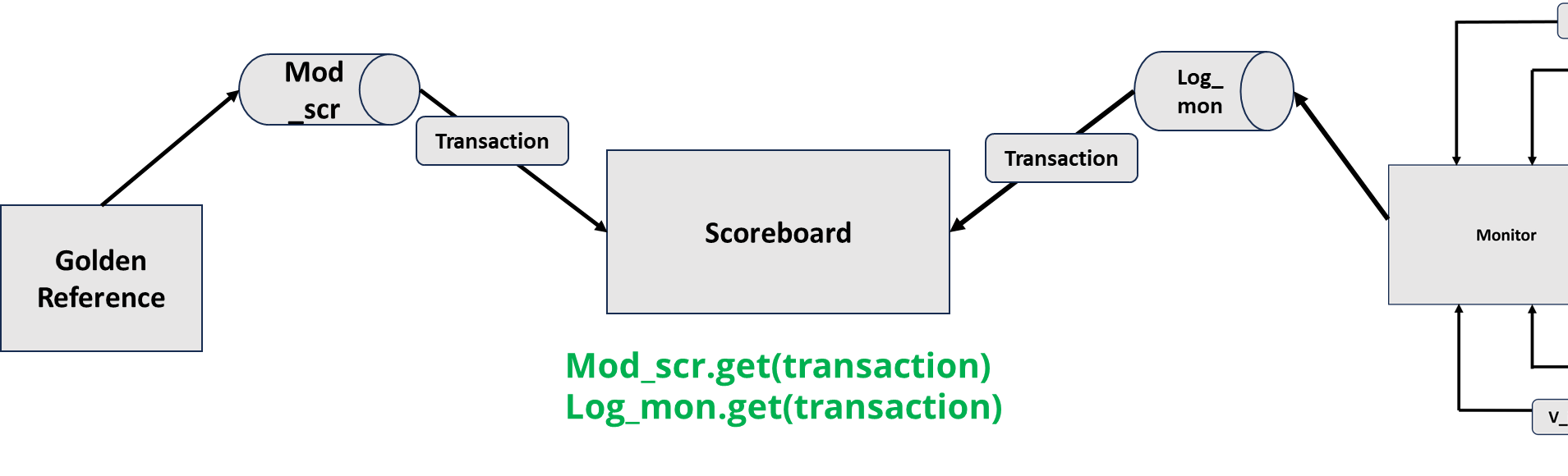
* Monitor obtains the interface contents (inputs/ outputs).
* Creates the transaction.

#### Step 6:



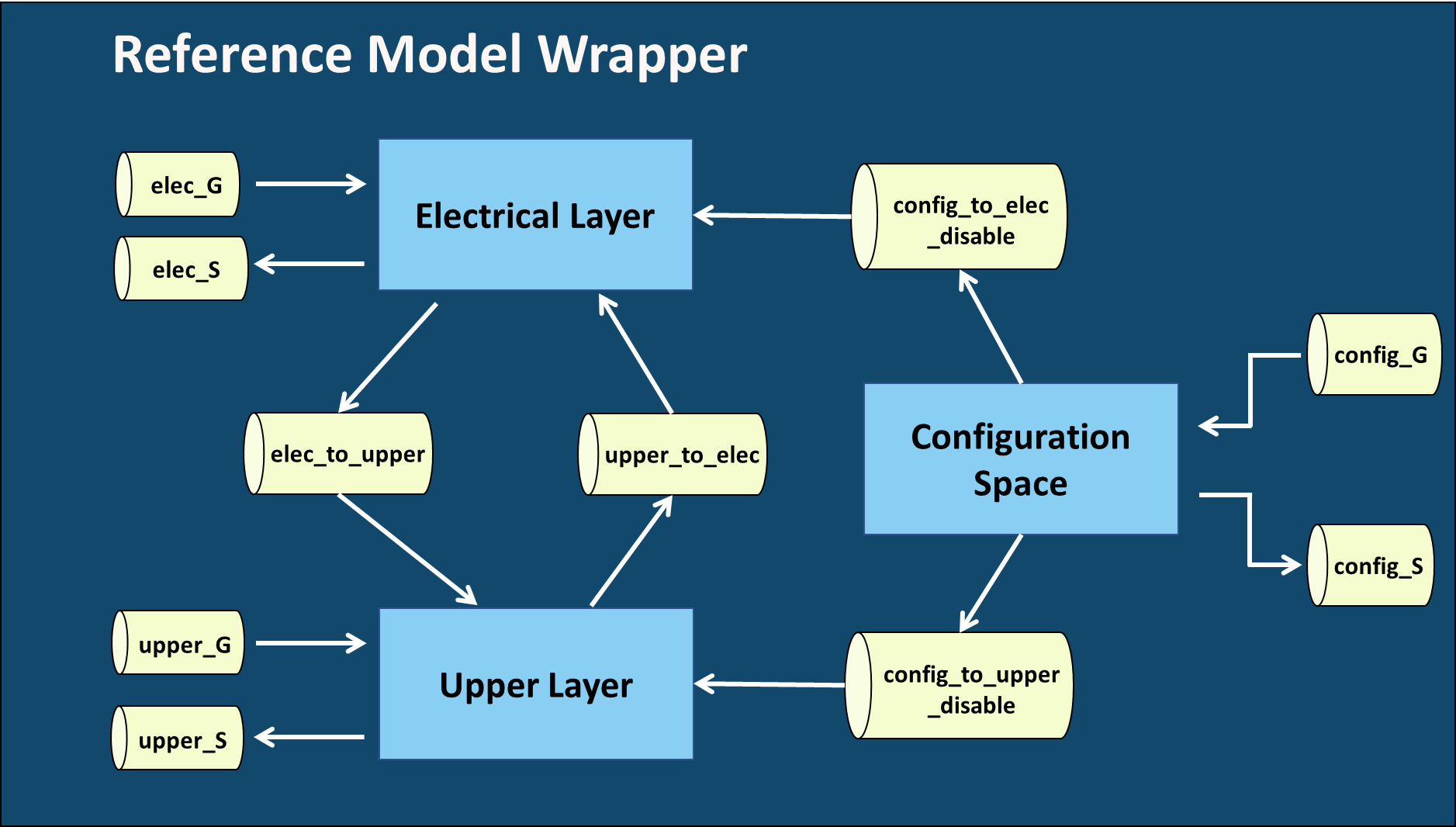
* Monitor sends data to the scoreboard’s mailbox connected via the **.put function**

#### Step 7:



* The scoreboard receives both transactions from the golden reference and the monitor.
* Performs direct comparison between the inputs.
* Proceeds to check the DUT outputs to verify its functionality.

# AI Reference Model:



The approach we used to build the reference model using Ai was to divide the model into 3 main classes, each class mimics a part of a DUT, according to the interface it uses, where we have 3 main interfaces, one with the electrical layer, one with the upper layer (transport layer) and one with the configuration spaces. These 3 classes interact with each other and with the verification environment using mailboxes, then all the classes and mailboxes are integrated together in a wrapper that we named “Reference Model Wrapper”.

## Classes

### Configuration Space Class:

This class is responsible for mimicking what we are expecting the DUT to act like through its configuration space interface, where it is responsible to do the following:

* At the beginning of the connection, it sends a read command to the configuration spaces, by raising the c\_read signal, and specifies the address the needs to be read from, which is 8’d18, then waits for the response from the configuration spaces, and if the response was not as expected (in case of any fault), read commands will be sent again until the correct data is received.
* After this step, the class waits for the lane\_disable signal to be raised, and in that case, the connection should be terminated, so the configuration space class sends a transaction to the electrical layer class and upper layer class through the mailboxes “config\_to\_elec\_disable” and “config\_to\_upper\_disable” to inform them that the connection has been disabled.

### Upper Layer Class:

This class is responsible for mimicking what we are expecting the DUT to act like through its Transport layer interface, where it is responsible to do the following:

* It receives the data from the transport layer, forwards it to the Electrical layer class where it should be sent out to its scoreboard.
* It also receives data internally from the electrical layer class and send it out to the upper layer scoreboard.

This was done by generating two threads working concurrently in each direction in order to receive and send data in both ways, Electrical to transport layer or Transport to electrical.

Since the two internal mailboxes, (upper\_to\_elec) & (elec\_to\_upper) are of different transaction type (electrical & upper layer transactions), it is mandatory to map the 8 bits of data coming from one interface to the other before forwarding the transaction to the appropriate scoreboard.

### Electrical Layer Class:

This class is responsible for mimicking what we are expecting the DUT to act like through its electrical layer interface, where it is responsible to do the following:

* Performs phase 2 of lane initialization (Router Detection).
* Performs phase 3 of lane initialization (Determining Link Attributes).
* Performs phase 4 of lane initialization (Link Training for Synchronization).
* Performs phase 5 of lane initialization (Exchanging Transport Layer Packets).

**Phase 2 (Router Detection)**:

This phase is responsible for router detection, where each router raises its sbtx, and waits for sbrx to be raised, to detect that another router is connected to this router, so during this phase, a transaction is sent to the scoreboard of the electrical layer, with the sbtx signal raised to 1.

**Phase 3 (Determining Link Attributes):**

Since this model mimics a “**Host**” Router, this phase starts by sending out the AT command to the electrical layer Scoreboard with the following transaction contents:

* Address = 78 which is the address of Register 12 in the sideband registers.
* Length = 3 to read the first 3 bytes only
* Empty data symbols since it is a Read command.

The Model then waits for an AT Command from the Generator mailbox with the same address of 78 to issue the AT response with the following Transaction Contents:

* Address = 78, Length = 3.
* Data Symbols = 24’h053303 which corresponds to saying that the model supports up to Gen4 Speed as expected by the Logical Layer itself.

For both Transactions, a CRC generator is used to create and transmit the CRC inside the same transaction.

**Phase 4 (Link Training for Synchronization):**

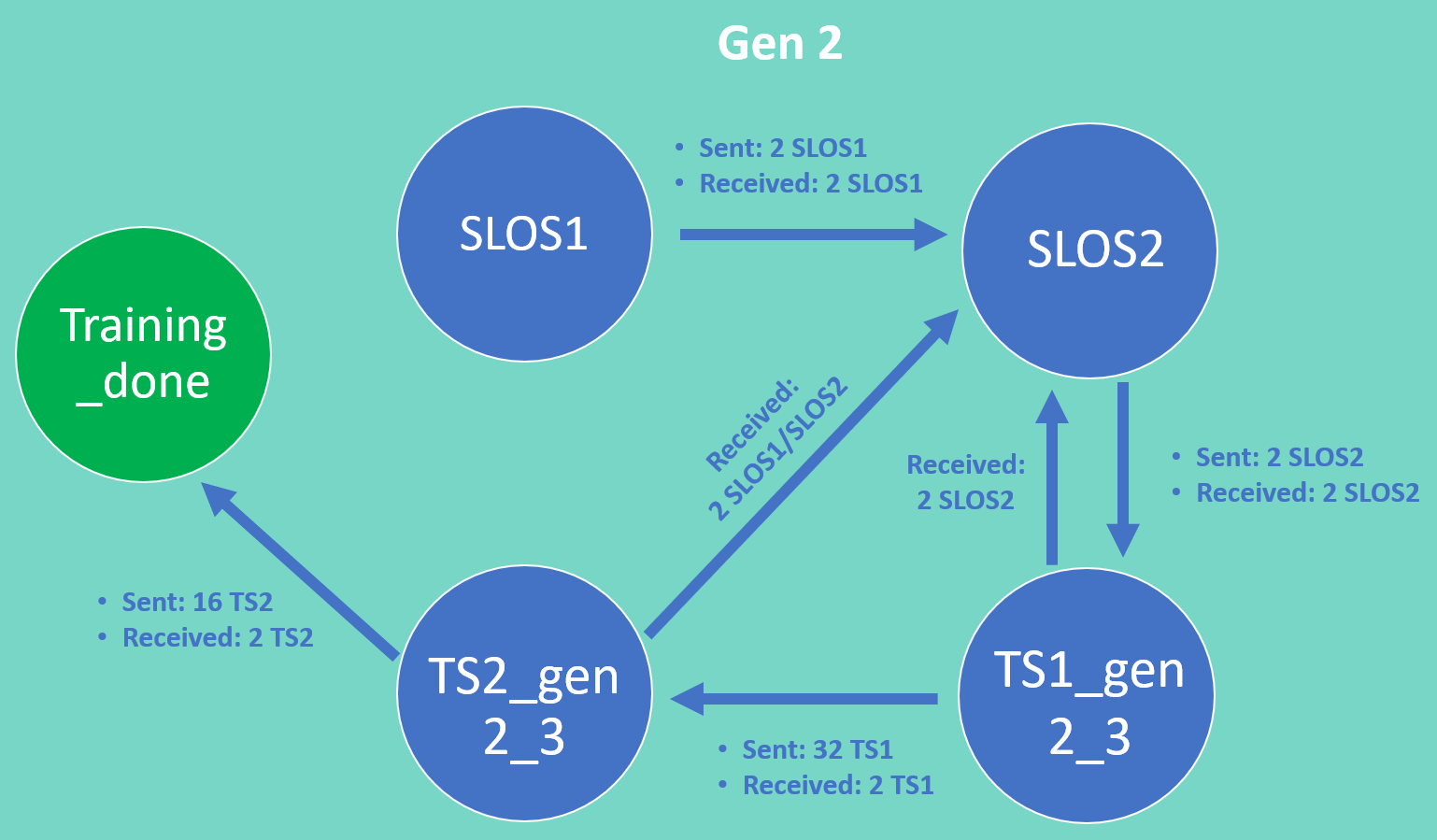
This phase is responsible for training, where each router sends ordered sets to the other router according to the generation speed that they agreed upon during phase 3, and the each ordered set is sent a certain amount of time, so a finite state machine (FSM) is needed inside this class to control the transition and the ordered sets to be sent, and it is according to the following for each generation speed:

* Gen 2: The model should keep sending SLOS1 until it has sent at least 2 SLOS1 and has received 2 SLOS1 in a row, then keeps sending SLOS2 until it has sent at least 2 SLOS2 and has received 2 SLOS2 in a row, then keeps sending TS1 until it has sent at least 32 TS1 and has received 2 TS1 in a row, then keeps sending TS2 until it has sent at least 16 TS2 and has received 2 TS2 in a row.
* Gen 3: The model should keep sending SLOS1 until it has sent at least 2 SLOS1 and has received 2 SLOS1 in a row, then keeps sending SLOS2 until it has sent at least 2 SLOS2 and has received 2 SLOS2 in a row, then keeps sending TS1 until it has sent at least 16 TS1 and has received 2 TS1 in a row, then keeps sending TS2 until it has sent at least 8 TS2 and has received 2 TS2 in a row.
* Gen 4: The model should keep sending TS1 until it has sent at least 16 TS1 and has received at least 1 TS1 or TS2, then keeps sending TS2 until it has sent at least 16 TS2 and has received at least 1 TS2 or TS3, then keeps sending TS3 until it has sent at least 16 TS3 and has received at least 1 TS3 or TS4, then keeps sending TS4 until it has sent at least 16 TS4.
* Additional transitions for Gen 2 and Gen 3:
  + The model should move from the state of TS1 to the state of SLOS2 if it received 2 SLOS1 Symbols in a row.
  + The model should move from state of TS2 to the state of SLOS2 if it received 2 SLOS Symbols (SLOS1 and/or SLOS2) in a row.

**Note:** The shape of SLOS1 and SLOS2 ordered sets is not the same for Gen 2 and Gen 3, and the shape of TS1 and TS2 is not the same for Gen 2, Gen 3 and Gen 4.

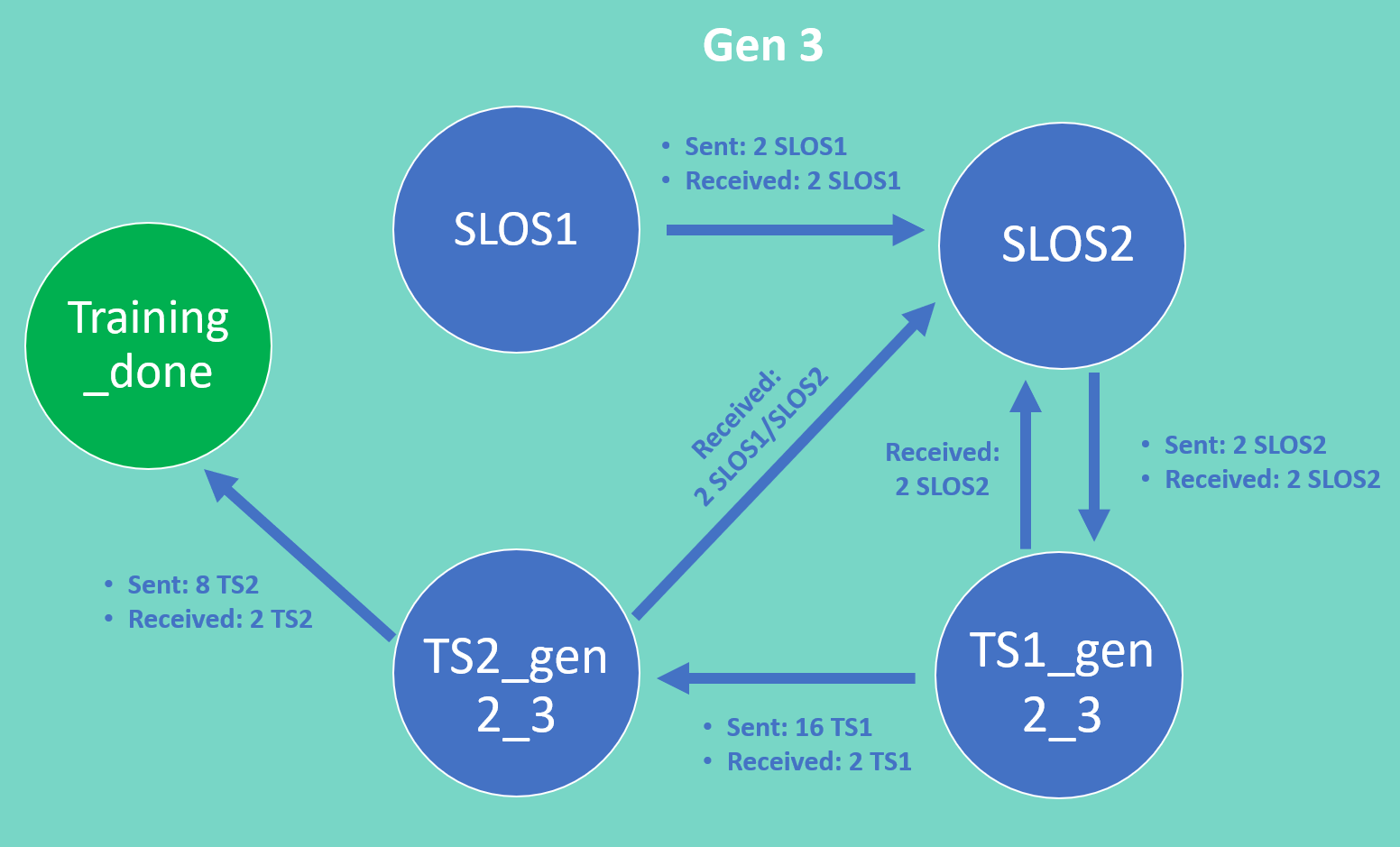
This phase has the following finite state machines (FSMs):

* FSM for Gen 2



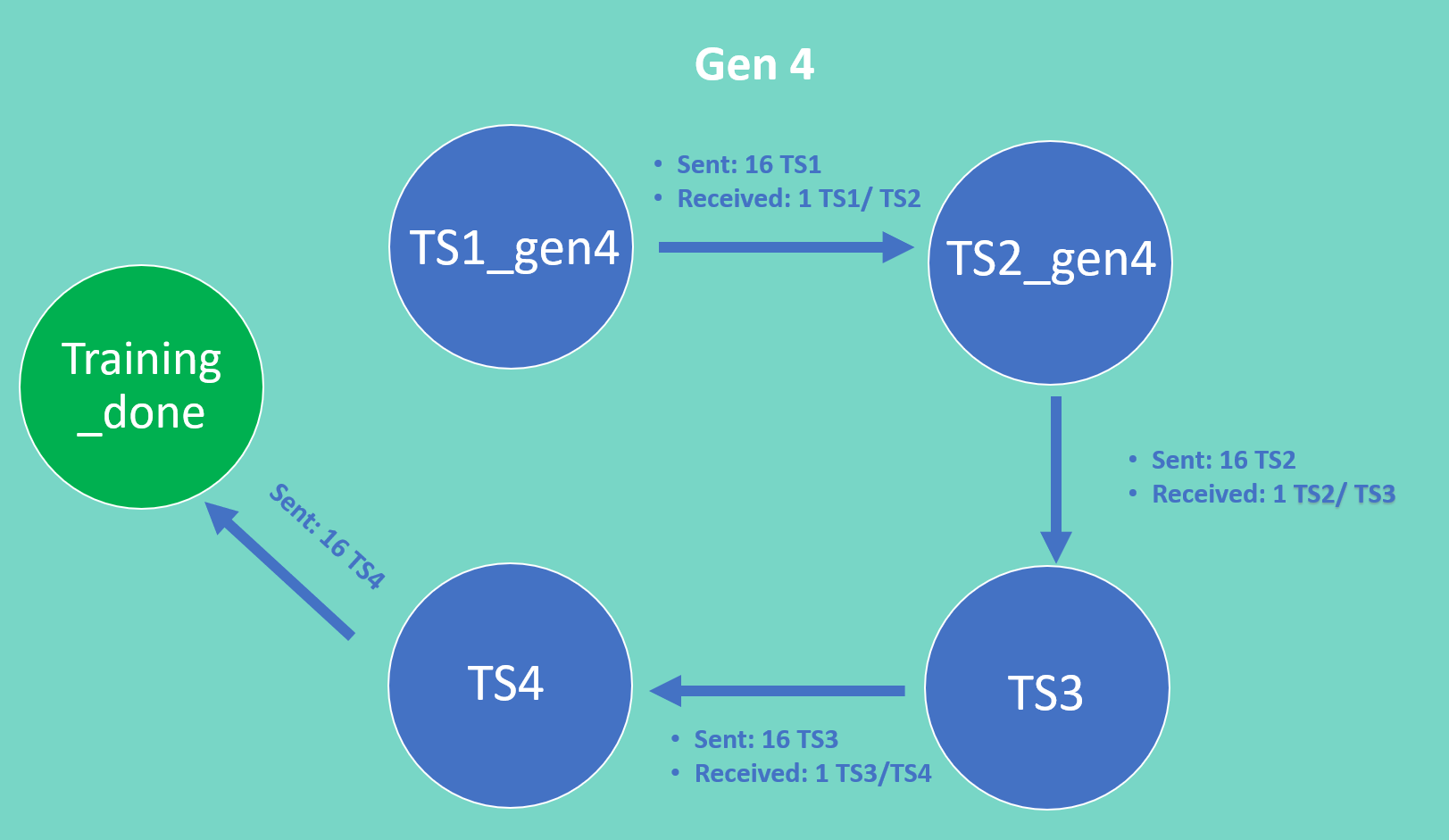
The number of sent ordered set is the minimum number to be sent, and the number of received ordered sets should be received in a row (consecutive).

* FSM for Gen 3



The number of sent ordered set is the minimum number to be sent, and the number of received ordered sets should be received in a row (consecutive).

* FSM for Gen 4



The number of sent/received ordered set is the minimum number to be sent/received.

**Phase 5 (Exchanging Transport Layer Packets):**

Just like the upper layer Class, this phase needs two threads working together but this time to receive data from the internal mailbox (upper\_to\_elec) and forwarding its contents to the electrical layer scoreboard or taking data from the electrical generator and forwarding it to the (elec\_to\_upper) internal mailbox.

# Design Limitations:

1. No Logical Layer Encoding:

* The logical layer should provide Reed Solomon Forward Error Correction (RS-FEC) but it won’t be part of this implementation.

1. No Compliance Testing:

* There are no built-in compliance testing features within the logical layer.

1. No Low Power States:

* The logical layer does not support low power states, meaning it operates at full power without power-saving features.

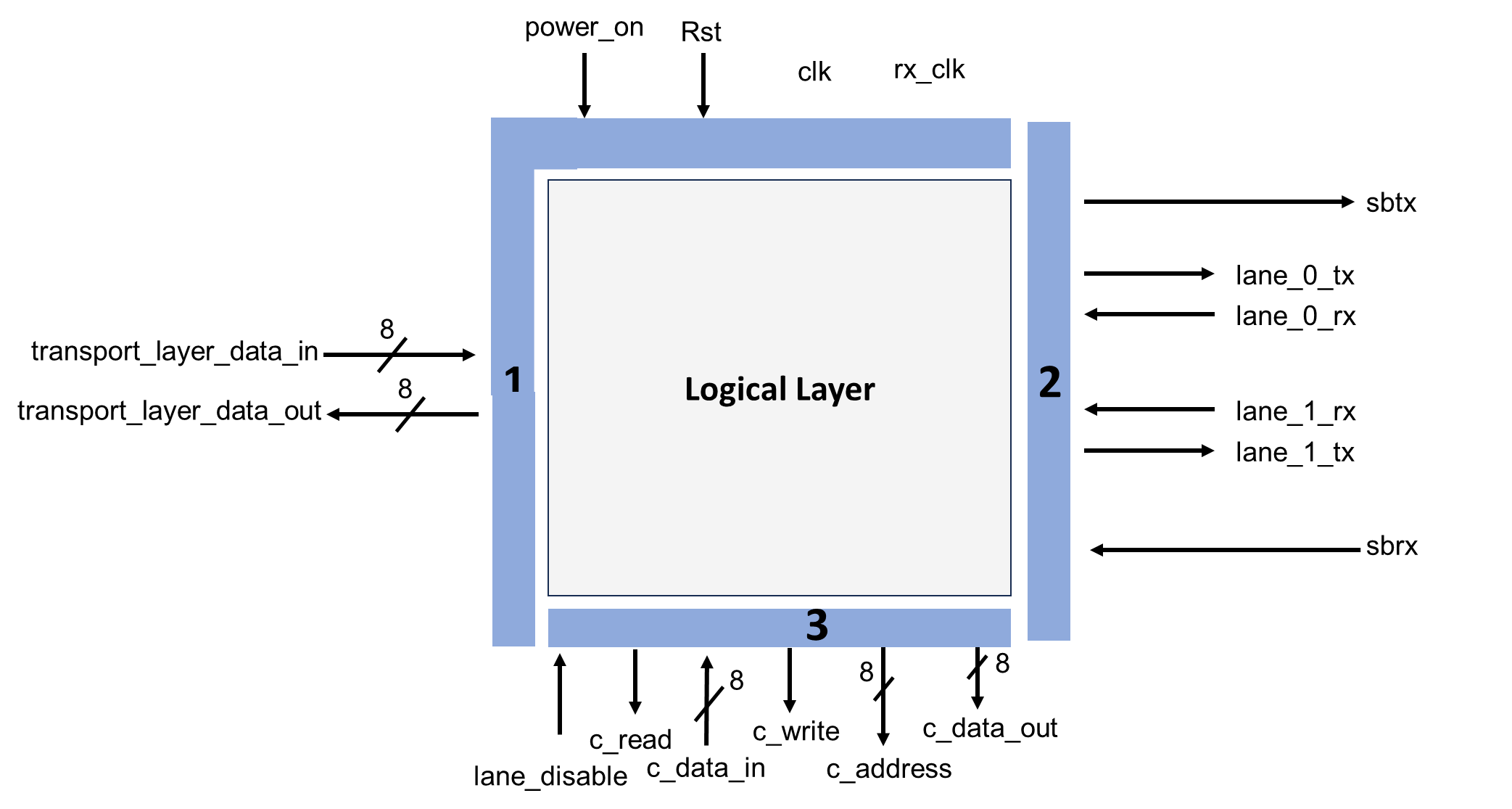
1. No Re-timers

* The connecting cable is assumed to be always passive (have no re-timers)

1. No PRTS Generation
   * PRTS is not included in the ordered sets in gen 4 (only the header is sent).

# SYSTEM Level VERIFICATION:

## Interfaces:





For maximum Verification Flexibility, the Interface will be divided into 3 interfaces representing 3 different Entities:

1. Upper layer Protocols
2. Electrical layer (Opposite Router)
3. Configuration Spaces

## Interface 1 ports (Upper Layer Interface):

|  |  |  |
| --- | --- | --- |
| Name | Size | Description |
| Operating Clocks | | |
| clk | 1 | Local clock |
| gen2\_fsm\_clk | 1 | 9.697 GHz Clock frequency |
| gen3\_fsm\_clk | 1 | 19.394 GHz Clock frequency |
| gen4\_fsm\_clk | 1 | 40 GHz Clock frequency |
| Interface Signals | | |
| rst | 1 | Reset |
| tansport\_layer\_data\_in | [7:0] | Packet transmitter/ receiver that connects to the transport layer |
| tansport\_layer\_data\_out | [7:0] | Packet transmitter/ receiver that connects to the transport layer |
| Internal Signals | | |
| generation\_speed | Enum  (GEN) | Indicates the Generation speed to determine the operating clock frequency.   * Gen2 * Gen3 * Gen4 |
| phase | 3 | Indicates the current operating phase.   * 1 --> phase 1 * 2 --> phase 2 * 3 --> phase 3 * 4 --> phase 4 * 5 --> phase 5 |

## Interface 2 Ports (Electrical Layer Interface):

|  |  |  |
| --- | --- | --- |
| Name | Size | Description |
| Operating Clocks | | |
| clk | 1 | Local clock |
| SB\_clock | 1 | Sideband Clock: 1 MHz |
| gen2\_lane\_clk | 1 | 2nd generation lane Clock: 10 GHz |
| gen3\_lane\_clk | 1 | 3rd generation lane Clock: 20 GHz |
| gen4\_lane\_clk | 1 | 4th generation lane Clock: 40 GHz |
| Interface Signals | | |
| sbrx | 1 | Sideband receiver |
| lane\_0\_rx | 1 | Packet/ ordered set receiver |
| lane\_1\_rx | 1 | Packet/ ordered set receiver |
| sbtx | 1 | Sideband transmitter |
| lane\_0\_tx | 1 | Packet/ ordered set transmitter |
| lane\_1\_tx | 1 | Packet/ ordered set transmitter |
| Internal Signals | | |
| generation\_speed | Enum  (GEN) | Indicates the Generation speed to determine the operating clock frequency.   * Gen2 * Gen3   Gen4 |
| phase | 3 | Indicates the current operating phase.   * 1 --> phase 1 * 2 --> phase 2 * 3 --> phase 3 * 4 --> phase 4 * 5 --> phase 5 |

## Interface 3 Ports(Configuration Space Interface):

|  |  |  |
| --- | --- | --- |
| Name | Size | Description |
| Operating Clocks | | |
| clk | 1 | Local clock |
| gen4\_fsm\_clk | 1 | 4th generation lane Clock: 40 GHz |
| Interface Signals | | |
| lane\_disable | 1 | Disable the FSM Control Unit block |
| c\_data\_in | [7:0] | Data read from the configuration spaces |
| c\_read | 1 | Read flag for the configuration spaces |
| c\_write | 1 | Write flag for the configuration spaces |
| c\_address | [7:0] | Address to read/write to the configuration space |
| c\_data\_out | [7:0] | Data to be written in the configuration spaces |

## ENVIRONMENTS TRANSACTIONS:

### Transport Layer Transaction

|  |  |  |
| --- | --- | --- |
| Bin name | Size (bits) | Description |
| T\_DATA | 8 | Refer to data between transport layer and electrical layer. |
| phase | 3 |  |
| Gen\_speed | Enum  (GEN) | Indicates the generation speed to be used.  - 00’b --> gen2  - 01’b --> gen3  - 10’b --> gen4 |

### Configuration Spaces Transaction

|  |  |  |
| --- | --- | --- |
| Bin name | Size(bits) | Description |
| lane\_disable | 1 | Disable the FSM Control Unit block |
| c\_data\_in | [7:0] | Data read from the configuration spaces  -40’h: In case of USB4  -??’h: In case of Gen4 |
| c\_data\_out | [7:0] | Data to be written in the configuration spaces |
| c\_read | 1 | Read flag for the configuration spaces |
| c\_write | 1 | Write flag for the configuration spaces |
| c\_address | [7:0] | Address to read/write to the configuration space |

### Electrical Layer transactions:

|  |  |  |
| --- | --- | --- |
| Bin name | Size(bits) | Description |
| sbrx | 1 | SBRX (for sending data to the DUT)  Always “1” starting from the training phase except when sending LT\_Fall command. |
| transaction\_type | Enum  (tr\_type) | -000’b: none  -001’b: LT\_Fall  -010’b: AT\_cmd  -011’b: AT\_rsp  -100’b: LT\_Fall\_wrong  -101’b: AT\_cmd\_wrong  -110’b: AT\_rsp\_wrong |
| read\_write | 1 | Indicates whether the AT operation is read or write.  -0’b: read  -1’b: write |
| address | 8 | Address of the register being read from or written to. |
| len | 7 | Number of bytes to read/write. Shall not be greater than 64. |
| cmd\_rsp\_data | 24 | Cmd: Data to be written  Rsp: Data read  **Note:**  -Bit 5 in byte 1 in register 12 = 1 in case of GEN3 support.  -Bit 2 in byte 2 in register 12 = 1 in case of GEN4 support.  - If both bits = 0, therefore GEN2 support.  \*\*Length in standard: Not more than 64 bytes (we assumed 3 bytes only for simplicity). |
| o\_sets | Enum  (OS\_type) | Refer to type of Ordered set.  -1000’b: refer to SLOS 1.  -1001’b: refer to SLOS 2.  -1010’b: refer to TS1 for (Gen 2,3).  -1011’b: refer to TS2 for (Gen 2,3).  -1100’b: refer to TS1 for gen 4.  -1101’b: refer to TS2 for gen 4.  -1110’b: refer to TS3 for gen 4.  -1111’b: refer to TS4 for gen 4.  -MSB refers to whether there are O\_sets or no.  (\* 0xxx’b: refer to no O\_sets.  \* 1xxx’b: refer to there are O\_sets.) |
| gen\_speed | Enum  (GEN) | Indicates the generation.  -00’b --> gen2  -01’b --> gen3  -10’b --> gen4 |
| electrical\_to\_transport | 8 | Data sent after training between electrical layer and transport layer. |
| phase | 3 | -001’b -->we are in phase 1  -010’b --> we are in phase 2  -011’b --> we are in phase 3  -100’b --> we are in phase 4  -101’b --> we are in phase 5  default --> we aren’t in any phase "sbtx = 0" |
| tr\_os | 2 | Indicates whether the driver will send transaction or ordered set. |
| sbtx | 1 | SBTX (for receiving data from the DUT) |
| transport\_to\_electrical | 8 | Data sent after training between electrical layer and transport layer. |
| crc\_received | 16 | Crc field received in the AT cmd and AT rsp |
| order | 4 | Indicates the order of the TS ordered set |
| lane | Enum  (LANE) | Indicates lane that the data is associated with.  -00 --> None  -01 --> lane\_0  -10 --> lane\_1  -11 --> both |

## USB4 Logical Layer Flow

|  |  |  |
| --- | --- | --- |
| No. | Flow | Description |
| 1 | Reset | SBTX should be “0” once the reset is asserted. |
| Phase 1: initial condition acquisition | | |
| 2 | Detection of USB4 capability | The logical layer should send signal via c\_read/write, c\_address to acquire the required data about the connection |
| 3 | Detection of USB4 generation | If USB4 connection is detected, the layer then proceeds with the same manner to obtain information about the USB4 version (gen2 or gen3 or gen4?) |
| Phase 2: Router Detection | | |
| 4 | Router Detection | * Once SBRX is detected high, the SBTX should turn high as well. * If the USB4 Port receives a Transaction while it is still in Phase 2, it may ignore or drop the Transaction. |
| Phase 3: Exchange of lane parameters | | |
| 5 | Lane parameter request | The router is required to send AT transaction via the SBTX which have the address of the register 12 to be read from the sideband of the other router |
| 6 | Saving requested lane parameters | The router should detect the parameters that were requested via an AT Response on its SBRX and save their values internally. |
| 7 | Response to an AT command | The router should then respond to the AT command received on its SBRX by sending the reg 12 SB register via its SBTX |
| Phase 4: Link Training | | |
|  | Gen4 | |
| 8 | TS1 state behavior | * Layer should send at least 16 TS1s with indication field = 2h on lane\_0\_tx and lane\_1\_tx * Remain in this state until it receives (on lane\_x\_rx): * Received TS1 with indication field = 2h * Or Received TS2 |
| 9 | TS2 state behavior | * Layer should send at least 16 TS2s with indication field = 4h on lane\_0\_tx and lane\_1\_tx * Remain in this state until it receives (on lane\_x\_rx): * Received TS2 with indication field = 4h * Or Received TS3 |
| 10 | TS3 state behavior | * Layer should send at least 16 TS3s with indication field = 6h on lane\_0\_tx and lane\_1\_tx * Remain in this state until it receives (on lane\_x\_rx): * Received TS3 with indication field = 6h * Or Received TS4 |
| 11 | TS4 state behavior | Layer should send TS4 in increasing order of counter field until it reaches value = Fh |
|  | Gen2 and Gen3 | |
| 12 | SLOS1 state behavior | * Layer should send 2 SLOS1 * Remain in this state until it receives 2 SLOS1 (on lane\_x\_rx). |
| 13 | SLOS2 state behavior | * Layer should send 2 SLOS2 * Remain in this state until it receives 2 SLOS2 (on lane\_x\_rx). |
| 14 | TS1 state behavior | * Layer should send 32 TS1 for gen 2 and 16 TS1 for gen 3 * Remain in this state until it receives 2 TS1 (on lane\_x\_rx). |
| 15 | TS2 state behavior | * Layer should send 16 TS2 for gen 2 and 8 TS2 for gen 3 * Remain in this state until it receives 2 TS2 (on lane\_x\_rx). |
| 16 | Training Disable | If lane\_disable bit inside the configuration space changed to 1, the layer should be disabled and no ordered sets should be sent or processed and SBTX should be zeros. |
| 17 | Timeout | If the layer remains for tTrainingAbort time in the same state, the layer should repeat the process of initialization again from phase 3. |
| Phase 5: Exchanging Transport Layer Packets | | |
| 18 | Packet transmission from its own router transport layer to another router | Any payload received via the transport\_layer\_data port should be forwarded to the lane\_x\_tx in parallel |
| 19 | Packet transmission from another router to its own router transport layer | Any payload received via the lane\_x\_rx should be forwarded to the transport\_layer\_data port. |
| 20 | CL0 Disable | If lane\_disable bit inside the configuration space changed to 1, the layer should be disabled and no packets should be forwarded. And SBTX should be zeros. |
| Disconnections | | |
| 21 | LT\_fall transaction behavior | * Should begin phase 1 again. * Load the configuration spaces and SB registers with the default values as required by the specifications * And SBTX should be zeros |
| 22 | SBRX low behavior | * Should begin phase 1 again. * SBTX should be zeros |
| Outputs default behavior | | |
| 23 | SBTX | * Should be = 0 when the router just turned on or no connection is still established. * Should be = 1 when there is a link connection with other routers but no transactions to be sent. |
| Dropping Transactions | | |
| 24 | Dropping LT Transactions | The LT transactions will be dropped if the received CLSE symbol (byte 2) is not the bitwise complement of the LSE symbol (byte 1). |
| 25 | Dropping AT Transactions | * The AT transactions will be dropped if the received CRC transaction is invalid. * The AT transactions will be dropped if the transaction has no data and no CRC field. |
| Managing wrong inputs | | |
| 26 | Unsupported Generation | If the connection type stored in the configuration unit is not supported (not gen2, gen3 or gen4), the logical layer shall not proceed the CLd substates. |
| 27 | Ignoring unexpected ordered sets | * If the negotiated speed is gen2 or gen3, and the received ordered sets are incorrect or related to gen4. * If the negotiated speed is gen 4, and the received ordered sets are incorrect or related to gen2 or gen3. |
| 28 | Ignoring incorrect transactions | If the received transaction on the SBRX is incorrect, the transactions’ fsm will act according to its current state. |
| 29 | Ignoring write commands in read-only SB register locations | If the link partner sends AT command to write in a RO SB register location, the logical layer shall ignore this write command. |

## Test Scenarios

|  |  |  |  |
| --- | --- | --- | --- |
| Test Scenario | Description | Generation | Status |
| Normal Scenarios | | | |
| Basic Operation | Going Through All Lane Initialization Phases | GEN4 | COMPLETED |
| GEN3 | COMPLETED |
| GEN2 | COMPLETED |
| Basic Operation with Disconnection (SBRX low) | The Logical Layer should start Lane Initialization from the beginning if SBRX was lowered | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Basic Operation with Disconnection (LT FALL) | The Logical Layer should start Lane Initialization from the beginning if it received LT FALL transaction | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Basic Operation with Lane disable (C\_disable) | The Logical Layer should start Lane Initialization from where it left off | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Negative Scenarios | | | |
| Mixing up Ordered Sets | Receiving 2 SLOS1 OR SLOS2 symbols in a row at State TS2 returns Training to State SLOS2 | GEN3 |  |
|  |
| GEN2 |  |
| Receiving 2 SLOS1 symbols in a row at StateTS1 returns Training to  State SLOS2 | GEN3 |  |
|
| GEN2 |  |
| Sending the ordered sets non consecutively Expected: The DUT should only consider consecutive ordered sets | GEN3 |  |
| GEN2 |  |
| Re-Training Logical Layer | Sending 2 SLOS Signals during  Phase 5(Exchanging Transport layer Packets) | GEN3 |  |
|
| GEN2 |  |
| Parameter Exchange Timeout | If the Logical Layer doesn't receive  an AT Response to an AT outstanding AT command, them, it should reissue an AT command again | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| SBRX raised for less than tConnectRX time | Logical Layer should Ignore the pulse and not proceed to phase 3 | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Sending AT command before  phase 2 completion | Shall be Ignored by the logical Layer (Should not send an AT Response) | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Fault Injection Scenarios | | | |
| Sending wrong AT Command | Shall be Ignored by the logical Layer  (Should not send an AT Response) | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Sending Wrong AT Response | the Logical Layer shall resend  an AT Command & not proceed to phase 4 | GEN4 |  |
| GEN3 |  |
| GEN2 |  |
| Sending Wrong Ordered Sets during Training | the Logical Layer shall wait for the  correct Ordered Set until the time out period and not Proceed to phase 5 | GEN4 |  |
| GEN3 |  |
| GEN2 |  |

## Task Breakdown:

|  |  |  |  |
| --- | --- | --- | --- |
| Task | | Team Member Responsible | Expected Completion TIME |
| Build Verification Environment  (Implementation) | Electrical layer Agent | Ali Ahmed &  Karim Samy | 9/2/2024 |
| Configuration space Agent | Karim Samy |
| Upper layer Agent | Ali Ahmed |
| Build Reference Model | | Mohamed Salah & Walid Salah | 9/2/2024 |
| Build Verification Environment  (Integration) | | Ali Ahmed &  Karim Samy | 10/2/2024 |
| Build Verification Environment  (Integration with DUT) | | All | 14/2/2024 |
| Basic Logical Layer Flow (1) Scenario | Development | Ali Ahmed &  Karim Samy | 24/2/2024 |
| Running |
| Debugging |
| Basic Logical Layer Flow (2) Scenario | Development | Ali Ahmed &  Karim Samy | 26/2/2024 |
| Running |
| Debugging |
| Basic Logical Layer Flow (3) Scenario | Development | Mohamed Salah & Walid Salah | 24/2/2024 |
| Running |
| Debugging |
| Mixing Ordered Sets | Development | Mohamed Salah & Walid Salah | 26/2/2024 |
| Running |
| Debugging |
| Wrong AT response | Development | Ali Ahmed &  Karim Samy | 28/2/2024 |
| Running |
| Debugging |
| Wrong AT Read command | Development | Mohamed Salah & Walid Salah | 28/2/2024 |
| Running |
| Debugging |
| Wrong AT Write command | Development | Ali Ahmed &  Karim Samy | 30/2/2024 |
| Running |
| Debugging |
| Wrong LT\_Fall | Development | Mohamed Salah & Walid Salah | 30/2/2024 |
| Running |
| Debugging |
|  | | | |
| AI | | | |
| Build Verification Environment  (Implementation) | | Mohamed Salah & Walid Salah | 8/5/2024 |
| Build Reference Model | | Ali Ahmed &  Karim Samy | 8/5/2024 |
| Build Verification Environment  (Integration with DUT) | | Mohamed Salah & Walid Salah | 8/5/2024 |
| Teset Scenarios | | To be decided | To be decided |

## Duration of each task (by hand):

|  |  |  |  |
| --- | --- | --- | --- |
| Task | | Time | |
| Build Verification Environment  (Implementation) | | | |
| Configuration Space’s Agent | | 4 hrs | |
| Configuration Space’s Sequence | | 2 hrs | |
| Configuration Space’s Scoreboard | | 1 hr | |
| Upper Layer’s Agent | Driver | 17 hrs | |
| Monitor | 15 hrs | |
| Upper Layer’s Sequence | | 8 hrs | |
| Upper Layer’s Scoreboard | | 2 hrs | |
| Electrical Layer’s Agent | Driver | 43 hrs | |
| Monitor | 55 hrs | |
| Electrical Layer’s Sequence | | 20 hrs | |
| Electrical Layer’s Scoreboard | | 4 hrs | |
| All Transactions and Interfaces | | 2 hrs | |
| Virtual Sequence | | 3 hrs | |
| Build Reference Model | Modelling diagram | 15 hrs | Avg time per day  (4.13 hrs/day) |
| Codding 1 (CRC, Serializer,  SB registers,) | 20 hrs |
| Build phases | 38 hrs |
| Debugging and connect with environment | 20 hrs |
| Build Verification Environment  (Integration) | | 3 hrs | |
| Build Verification Environment  (Integration with DUT) | | 1 hr | |
| Normal Scenarios | | | |
| Basic Operation | | | |
| Gen 4 | Development | 3 hrs | |
| Running |  | |
| Debugging | 81 hrs | |
| Gen 3 | Development | 5 hrs | |
| Running |  | |
| Debugging | 30 hrs | |
| Gen 2 | Development | 1 hr | |
| Running |  | |
| Debugging | 3 hrs | |
| Basic Operation with Disconnection (SBRX low) | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Basic Operation with Disconnection (LT FALL) | | | |
| Gen 4 | Development |  | |
| Running |
| Debugging |
| Gen 3 | Development |  | |
| Running |
| Debugging |
| Gen 2 | Development |  | |
| Running |
| Debugging |
| Basic Operation with Lane disable (C\_disable) | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Negative Testing | | | |
| Mixing up Ordered Sets (1) | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Mixing up Ordered Sets (2) | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Mixing up Ordered Sets (3) | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Re-Training Logical Layer | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Parameter Exchange Timeout | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| SBRX raised for less than tConnectRX time | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Sending AT command before | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Fault Injection Scenarios | | | |
| Sending wrong AT Command | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Sending Wrong AT Response | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |
| Sending Wrong Ordered Sets during Training | | | |
| Gen 4 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 3 | Development |  | |
| Running |  | |
| Debugging |  | |
| Gen 2 | Development |  | |
| Running |  | |
| Debugging |  | |

## Duration of each task (by AI):

|  |  |  |  |
| --- | --- | --- | --- |
| Task | | | Time |
| Build Verification Environment  (Implementation) | | | |
| Configuration Space’s Agent | | | 2 hrs |
| Configuration Space’s Sequence | | | 2 hrs |
| Configuration Space’s Scoreboard | | | 0.5 hr |
| Upper Layer’s Agent | Driver | | 5 hrs |
| Monitor | | 5 hrs |
| Upper Layer’s Sequence | | | 4 hrs |
| Upper Layer’s Scoreboard | | | 0.5 hrs |
| Electrical Layer’s Agent | Driver | | 31hrs |
| Monitor | | 43hrs |
| Electrical Layer’s Sequence | | | 15hrs |
| Electrical Layer’s Scoreboard | | | 2hrs |
| All Transactions and Interfaces | | | 2 hrs |
| Virtual Sequence | | | 1.5hrs |
| Build Reference Model | Reference Model Wrapper | | 2 hrs |
| Configuration Space Reference | | 2.5 hrs |
| Electrical Layer Reference | Phase 2 | 0.5 hrs |
| Phase 3 | 12 hrs |
| Phase 4 | 20 hrs |
| Phase 5 | 4 hrs |
| Upper Layer Reference | | 4 hrs |
| Build Verification Environment  (Integration) | | | 2 hrs |
| Build Verification Environment  (Integration with DUT) | | | 1 hrs |
| Normal Scenarios | | | |
| Basic Operation | | | |
| Gen 4 | Development | | 5hrs |
| Running | |  |
| Debugging | | 40hrs |
| Gen 3 | Development | | 6hrs |
| Running | |  |
| Debugging | | 10hrs |
| Gen 2 | Development | | 1hrs |
| Running | |  |
| Debugging | | 7hrs |