he eth\_phy\_10g module is a **top-level wrapper** for a 10G Ethernet PHY (Physical Layer) that connects a **MAC** (via XGMII interface) to a **serial interface** (typically a transceiver/SerDes). It instantiates two submodules:

* eth\_phy\_10g\_rx: handles **receive (RX)** path
* eth\_phy\_10g\_tx: handles **transmit (TX)** path

Let's go through it in detail:

**🧩 Module Parameters**

| **Parameter** | **Description** |
| --- | --- |
| DATA\_WIDTH | Width of the data bus, usually 64 bits |
| CTRL\_WIDTH | Control signal width (DATA\_WIDTH / 8) |
| HDR\_WIDTH | Width of header signals (typically 2 for 64b/66b encoding) |
| BIT\_REVERSE | Enable bit reversal (used for debugging or compliance) |
| SCRAMBLER\_DISABLE | Disable scrambler (bypass for testing) |
| PRBS31\_ENABLE | Enable PRBS31 test pattern |
| TX\_SERDES\_PIPELINE | Number of pipeline stages in TX path |
| RX\_SERDES\_PIPELINE | Number of pipeline stages in RX path |
| BITSLIP\_HIGH\_CYCLES / LOW\_CYCLES | Timing controls for bitslip FSM |
| COUNT\_125US | Number of cycles for ~125us timing (used in BER monitoring) |

**🔌 I/O Ports**

**🧾 Common Signals**

* rx\_clk, rx\_rst: Clock and reset for RX logic
* tx\_clk, tx\_rst: Clock and reset for TX logic

**🖧 XGMII Interface (Connection to MAC)**

| **Signal** | **Dir** | **Description** |
| --- | --- | --- |
| xgmii\_txd | In | Transmit data from MAC |
| xgmii\_txc | In | Transmit control from MAC |
| xgmii\_rxd | Out | Received data to MAC |
| xgmii\_rxc | Out | Received control to MAC |

**🔗 SERDES Interface (Connection to transceiver/PCS)**

| **Signal** | **Dir** | **Description** |
| --- | --- | --- |
| serdes\_tx\_data | Out | Parallel data to SERDES (TX) |
| serdes\_tx\_hdr | Out | Header for 64b/66b (TX) |
| serdes\_rx\_data | In | Parallel data from SERDES (RX) |
| serdes\_rx\_hdr | In | Header from SERDES (RX) |
| serdes\_rx\_bitslip | Out | Request to perform bitslip |
| serdes\_rx\_reset\_req | Out | Request SERDES reset on RX errors |

**📊 Status Signals**

* tx\_bad\_block: TX sent an invalid 64b/66b block
* rx\_error\_count: Number of detected RX errors
* rx\_bad\_block: RX detected an invalid 64b/66b block
* rx\_sequence\_error: RX detected a bad block sequence
* rx\_block\_lock: RX has locked onto block boundaries
* rx\_high\_ber: High bit error rate detected
* rx\_status: Indicates link status (typically 1 when locked and good)

**⚙️ Configuration Inputs**

* cfg\_tx\_prbs31\_enable, cfg\_rx\_prbs31\_enable: Enable PRBS31 generation/checking

**🧠 Module Functionality**

This top-level PHY module wires up the TX and RX datapaths:

**📥 eth\_phy\_10g\_rx**

This submodule:

* Accepts serdes\_rx\_data and serdes\_rx\_hdr
* Handles **64b/66b decoding**, descrambling, alignment, and PRBS checking (if enabled)
* Detects bad blocks, errors, and provides XGMII receive data and control (xgmii\_rxd, xgmii\_rxc)
* Implements bitslip and reset\_req to recover from loss-of-alignment or high BER

**📤 eth\_phy\_10g\_tx**

This submodule:

* Accepts xgmii\_txd and xgmii\_txc from MAC
* Converts to **64b/66b** format and scrambles the data
* Optionally generates PRBS31 test patterns
* Outputs serdes\_tx\_data and serdes\_tx\_hdr to the SERDES

XGMII (from MAC)

┌───────────────┐

│ │

│ eth\_phy\_10g\_tx│

│ │

└──────┬────────┘

│

64b/66b Encoded, Scrambled

│

┌──────▼────────┐

│ SERDES TX IF │ --> optical/electrical medium

└───────────────┘

┌───────────────┐

│ SERDES RX IF │

└──────┬────────┘

│

64b/66b Decoded, Descrambled

│

┌──────▼────────┐

│ eth\_phy\_10g\_rx│

│ │

└───────────────┘

XGMII (to MAC)

**🔷 1. taxi\_eth\_phy\_10g\_tx.sv — Top-Level TX PHY Wrapper**

This is the **top wrapper module** that connects:

* XGMII interface (input)
* SERDES interface (output)
* Control/status/configuration signals

**🔸 Components inside:**

* taxi\_xgmii\_baser\_enc: Encodes XGMII data and control into 64b/66b format.
* taxi\_eth\_phy\_10g\_tx\_if: Handles scrambling, PRBS31 injection, optional bit reversal, and pipelines before transmitting to SERDES.

**🔸 Data Flow:**

[xgmii\_txd/txc] ──► [taxi\_xgmii\_baser\_enc]

│

▼

[encoded\_tx\_data, encoded\_tx\_hdr]

│

▼

[taxi\_eth\_phy\_10g\_tx\_if]

│

▼

[serdes\_tx\_data, serdes\_tx\_hdr]

**🔷 2. taxi\_xgmii\_baser\_enc.sv — XGMII to 10GBASE-R Encoder**

**🔸 Purpose:**

Encodes 64-bit XGMII (data + control) into 64b/66b blocks:

* Adds 2-bit sync header: 01 for control, 10 for data.
* Handles all 64b/66b block types: START, TERM, IDLE, OS, ERROR, etc.
* Detects special control characters and aligns them into legal block formats.

**🔸 Logic:**

* Based on the **control vector (xgmii\_txc)**, data lanes are interpreted.
* Builds valid 66-bit blocks by packing control/data + appropriate block type (like BLOCK\_TYPE\_TERM\_0, BLOCK\_TYPE\_OS\_START, etc.)
* Sets tx\_bad\_block when illegal combinations are detected.

**🔸 Output:**

* encoded\_tx\_data: 64-bit data block
* encoded\_tx\_hdr: 2-bit sync header
* tx\_bad\_block: High if illegal/unencodable XGMII sequence

**🔷 3. taxi\_eth\_phy\_10g\_tx\_if.sv — 10GBASE-R to SERDES Formatter**

**🔸 Purpose:**

Converts encoded\_tx\_data and encoded\_tx\_hdr into SERDES-ready output.

**🔸 Key Features:**

* **Scrambler**: 58-bit LFSR (IEEE 802.3 compliant), improves spectral characteristics.
* **PRBS31 Generator**: Generates pseudo-random bit streams for test modes.
* **Bit Reversal**: Optionally reverses bit order (useful for some SERDES configurations).
* **SERDES Pipelining**: Adds registered stages before SERDES output (configurable depth).
* **Gearbox Synchronization (GBX\_IF\_EN)**: Coordinates word boundaries when gearbox interfaces are used.

**🔸 Outputs:**

* serdes\_tx\_data: Final data to be serialized
* serdes\_tx\_hdr: 2-bit sync headers
* serdes\_tx\_\*\_valid: Indicates valid output
* serdes\_tx\_gbx\_sync: Sync pulse for gearbox alignment

**🔧 Configuration Parameters Summary**

| **Parameter** | **Description** |
| --- | --- |
| DATA\_W, CTRL\_W | Data and control widths (usually 64 and 8) |
| HDR\_W | Sync header width (fixed at 2 bits for 10GBASE-R) |
| GBX\_IF\_EN | Enables gearbox sync features |
| BIT\_REVERSE | Reverses bit order before output to SERDES |
| SCRAMBLER\_DISABLE | Disables 58-bit scrambler |
| PRBS31\_EN | Enables PRBS31 test pattern generation |
| SERDES\_PIPELINE | Sets number of pipeline stages to SERDES |

**🧠 Key Concepts**

* **XGMII** (10 Gigabit Media Independent Interface): Used between MAC and PHY.
* **64b/66b Encoding**: Reduces overhead compared to 8b/10b, adds sync headers.
* **Scrambling**: Prevents long runs of 0s/1s and reduces EMI.
* **PRBS31**: Pseudo-random binary sequence (test mode, ensures link integrity).
* **SERDES**: High-speed serial interface for transmitting bits over physical medium.

**✅ Final Output**

* Valid 66-bit blocks (64 data + 2 header) delivered through serdes\_tx\_data and serdes\_tx\_hdr to the physical SERDES.
* **The receive side of a 10G Ethernet PHY**:
* taxi\_eth\_phy\_10g\_rx.sv – Top-level wrapper for the RX PHY.
* taxi\_eth\_phy\_10g\_rx\_if.sv – Receive interface logic, including PRBS check, descrambler, and lane alignment.
* taxi\_xgmii\_baser\_dec.sv – Implements the **64b/66b decoder** converting serial 66-bit words to XGMII-style data and control signals.

**🔧 Top-Level Architecture Overview**

A screenshot of a computer code

AI-generated content may be incorrect.

**✅ 1. taxi\_eth\_phy\_10g\_rx.sv — Top-Level RX PHY**

**📌 Functionality**

* **Wraps** the interface and decoding blocks.
* Handles the RX clock and reset domain.
* Accepts 66-bit encoded data from SERDES.
* Outputs 64-bit rx\_data and 8-bit rx\_ctrl to higher layers (like MAC).

**🧱 Structure**

module taxi\_eth\_phy\_10g\_rx (

input wire clk,

input wire rst,

input wire [65:0] encoded\_rx\_data,

output wire [63:0] rx\_data,

output wire [7:0] rx\_ctrl,

output wire block\_lock

);

**🔗 Instantiates:**

* taxi\_eth\_phy\_10g\_rx\_if → processes encoded\_rx\_data, handles descrambling and block lock.
* taxi\_xgmii\_baser\_dec → performs 64b/66b decoding to generate rx\_data and rx\_ctrl.

**✅ 2. taxi\_eth\_phy\_10g\_rx\_if.sv — RX Interface (Descrambler, PRBS, Alignment)**

**📌 Functionality**

This module handles:

* **Block lock detection** (based on sync headers).
* **Descrambling** the 64-bit payload using a self-synchronous descrambler (x^58 + x^39 + 1).
* Optionally **checking for PRBS31 pattern** (useful in test mode).

**🧱 Key Elements**

sv

CopyEdit

input wire clk,

input wire rst,

input wire [65:0] encoded\_rx\_data, // 66-bit blocks from PCS

output wire [65:0] descrambled\_data, // To be decoded by 64b/66b decoder

output wire block\_lock // Indicates valid 64b/66b alignment

**⚙️ Main Processing:**

* **Sync Header Extraction:** encoded\_rx\_data[65:64] → should be 2'b10 (Data block) or 2'b01 (Control block)
* **Block Locking Logic:** Tracks valid headers over time to determine when to assert block\_lock.
* **Descrambling:**
  + Implements a **self-synchronous descrambler**.
  + Each new 64-bit data is XORed with a shifted LFSR value.

**✅ 3. taxi\_xgmii\_baser\_dec.sv — 64b/66b Decoder**

**📌 Functionality**

* Accepts **descrambled 66-bit blocks** from PCS RX interface.
* Decodes them into:
  + 64-bit data (rx\_data)
  + 8-bit control signals (rx\_ctrl) — to differentiate between data and control characters (e.g., IDLE, START, TERM).

**🧱 Main Ports**

input wire clk,

input wire rst,

input wire block\_lock,

input wire [65:0] encoded\_rx\_data,

output wire [63:0] rx\_data,

output wire [7:0] rx\_ctrl

**⚙️ Decoding Process**

1. **Sync Header (encoded\_rx\_data[65:64])**
   * 2'b10 → Data block
   * 2'b01 → Control block
2. **Data Block**: Interpreted directly as 64-bit payload.
3. **Control Block**: Contains control characters in specific byte positions.
4. **Control Character Mapping**: Converts specific 8-bit patterns into XGMII control symbols (e.g., 0x07 = IDLE).

**💡 Block Lock Signal**

* **Why important?**
  + Ensures that 66-bit word boundaries are correct.
  + Without lock, data may be misaligned, resulting in garbage decode.
* **When is block\_lock asserted?**
  + After seeing a certain number of valid sync headers in a row (as per IEEE 802.3 Clause 49).

**📦 Data Flow Summary**

A screenshot of a computer code

AI-generated content may be incorrect.

**✅ Final Outputs from Top-Level RX Module**

* rx\_data[63:0]: Decoded Ethernet frame data.
* rx\_ctrl[7:0]: Control signals for each byte (0 = data, 1 = control).
* block\_lock: High when the 66b decoder is locked and decoding is reliable.