Ultra Ethernet PHY Layer

Under the supervision of:
Dr. Emad Badry

Ultra Ethernet PHY Layer: RTL Design & Verification

- Team members: 5 students
- Supervisors: University + Siemens Mentor
- Duration: Dec 2025 – May 2026 (~6 months)
- Tools: SystemVerilog, UVM, QuestaSim/Icarus, Vivado(FPGA prototyping), GitHub/CI

Motivation & Background

Why Ultra Ethernet?

- High-performance Ethernet for AI clusters, HPC, and data centers

- Target speeds: 100G (student-feasible instead of 400 - 800G)

- Low latency and high reliability

Why PHY Layer?

- Converts digital MAC frames to physical signals

- Critical for throughput, error correction, and clock alignment

Why Our Project Matters

- Rare topic in academia
- Real industry protocol

Project Objectives

Main Goal

- Implement a **synthesizable 100G Ultra Ethernet PHY** in RTL
- Build a **SystemVerilog + UVM verification environment**

Sub-Goals

- Understand PHY architecture: PCS, FEC, Gearbox
- Implement parallel-domain blocks:
  - 64b/66b Encoder
  - Scrambler
  - Alignment / Deskew
  - RS-FEC Encoder + Decoder
  - Gearbox (parallel → serial mapping)
- Build verification environment:
  - Constrained random tests
  - Scoreboard & coverage collection
  - Assertions

System Overview (High-Level)

Diagram (simplified for 100G): MAC Layer → PCS → FEC → Gearbox → PMA → SerDes → Fiber/Copper

Blocks Overview:

- **MAC Layer:** Provides 64-bit client words @ 100–156 MHz
- **PCS (Physical Coding Sublayer):**
  - 64b/66b line coding
  - Scrambling
  - Block framing
- **FEC (Reed-Solomon):** Adds redundancy to correct bit errors
- **Gearbox:** Converts narrow parallel → wide parallel (lowers clock)
- **SerDes:** Serializes parallel data into multiple lanes to reach 100G

Note: **SerDes & analog link also MAC layer are out of the scope of this project**
you focus on digital parallel-domain blocks.

How to Reach 100G from 100–156 MHz

Example Calculation for 100G PHY:

- Input: 64-bit @ 156.25 MHz → 10 Gb/s
- Encoding + FEC: 64b/66b + RS-FEC
- Gearbox: widen to 512-bit → lowers clock to ~12.5 MHz
- SerDes: 4 lanes × 25.78 Gb/s = 103 Gbps (raw)

Key Concept:

- Your RTL runs at **low MHz clock**
- SerDes handles **high-speed serialization** (multi-GHz internally, not student design)

Project Scope (Academic-Friendly)

Included (Student Tasks)

- 64b/66b Encoder
- Scrambler
- Block aligner & deskew
- RS-FEC encoder + decoder (simplified)
- Gearbox for lane mapping
- SystemVerilog + UVM testbench

Not Included

- High-speed SerDes design
- Analog PMA / PMD
- Fiber/copper physical modeling
- PLL/CDR circuits

Focus: 100G digital PHY logic + verification environment

PCS Detailed Architecture

Block	Function	Notes
64b/66b Encoder	Converts 64-bit → 66-bit frames	Adds sync headers for alignment
Scrambler	Randomizes data patterns	Maintains DC balance & reduces EMI
Block Aligner	Finds sync headers on receiver	Aligns bit boundaries correctly
Deskew	Multi-lane deskew	Maintains order across lanes

RS-FEC Overview (Simplified)

- Purpose: Achieve low BER, correct burst errors from SerDes
- Example: RS(544,514) — adds 30 parity symbols
- Encoder Tasks:
  - GF(2^m) multiplication
  - Polynomial generation
- Decoder Tasks:
  - Syndrome calculation
  - Error locator polynomial (Berlekamp–Massey)
  - Chien search

Student Tip: Simplify decoder for manageable RTL, focus on **error detection + telemetry counters**

References:

- Reed–Solomon Wikipedia
- Altera RS-FEC Tutorial PDF

Gearbox (Bus Width / Clock Mapper)

- Converts slow, narrow parallel input → wide parallel output for multi-lane serialization

- Example:

- Input: 64-bit @ 100 MHz
- Output: 512-bit @ 12.5 MHz

→ feeds 4 SerDes lanes for 100G

Key Concept:

•Parallel width scaling allows low-frequency digital logic to produce ultra-high-speed serial outputs

Reference: IEEE 802.3 Clause 82

Verification Environment (SystemVerilog/UVM)

Components:

- UVM Agent (Driver + Monitor)
- Scoreboard (checks FEC & frame correctness)
- Coverage collector & assertions

Test Types:

- Random frame injection
- Bit-flip error injection (FEC test)
- Scrambler reset / frame alignment corner cases
- Stress tests

Timeline (Dec – May, 6 Months)

Month	Tasks
Dec – Jan	Study PHY, PCS, FEC, 64b/66b encoding, UVM basics
Feb	Implement 64b/66b Encoder + Scrambler RTL, basic testbench
Mar	Implement RS-FEC Encoder, start UVM environment
Apr	Implement RS-FEC Decoder, full UVM verification
May	Gearbox + lane mapping, integration, full system simulation, coverage closure, report + presentation

Expected Deliverables

RTL:

- Synthesizable SystemVerilog
- Reset & CDC strategy

Verification:

- UVM testbench with coverage
- Assertions
- Error injection & FEC verification

Documentation:

- PHY block summary
- Test plan & verification report
- Final project report + slides

Challenges & Innovations

RS-FEC decoder is mathematically complex

- 64b/66b encoder + scrambler must be cycle-accurate
- Gearbox alignment for multiple lanes
- Verification coverage closure (UVM)
- Maintaining realistic 100G protocol behavior in a student environment

Tools & Environment

- QuestaSim / VCS + UVM
- Vivado / Quartus for FPGA prototyping
- Synopsys Design Compiler (RTL checks)
- GitHub + CI pipelines
- MATLAB / Python (for FEC polynomials & simulation verification)

Materials / References

- IEEE 802.3 Standard (100G PCS, FEC): https://standards.ieee.org/standard/802_3-2022.html
- Open-source PHY RTL: alexforencich/verilog-ethernet
- 64b/66b Encoding: Wikipedia
- RS-FEC Tutorial: Altera PDF
- PHY / SerDes Basics Video: YouTube
- UVM Guide: Accellera

Team Members

Mohamed Ahmed Mohamed

Mostafa Mohamed Ahmed

Aya Mohamed abdelmaqid

Hazem Yasser Mahmoud

Mahmoud Elian

Thank You