Critical Design Review (CDR)

QPSK SDR over Optical Aurora Link with Forward Error Correction
JHU EN.525.743 Embedded Systems Development Lab
Alex Wozneak — Fall 2025

Table of Contents

---

1. Project Description

This project implements a QPSK/BPSK software-defined radio (SDR) system over an optical Aurora 64B/66B link on the Alinx AXU5EVB-P (AMD/Xilinx Zynq UltraScale+ MPSoC). The design integrates:

- Forward error correction (FEC) via convolutional codes (K=7, rate-1/2).
- Adaptive modulation and coding (AMC) for switching between {BPSK+FEC, QPSK+FEC, QPSK}.
- Host-side data visualization (constellation and BER).
- Preamble-based timing recovery (non-iterative, frame-synchronous).

Primary objective: Demonstrate robust end-to-end optical SDR chain with hardware-accelerated PHY functions, host visualization, and BER evaluation.

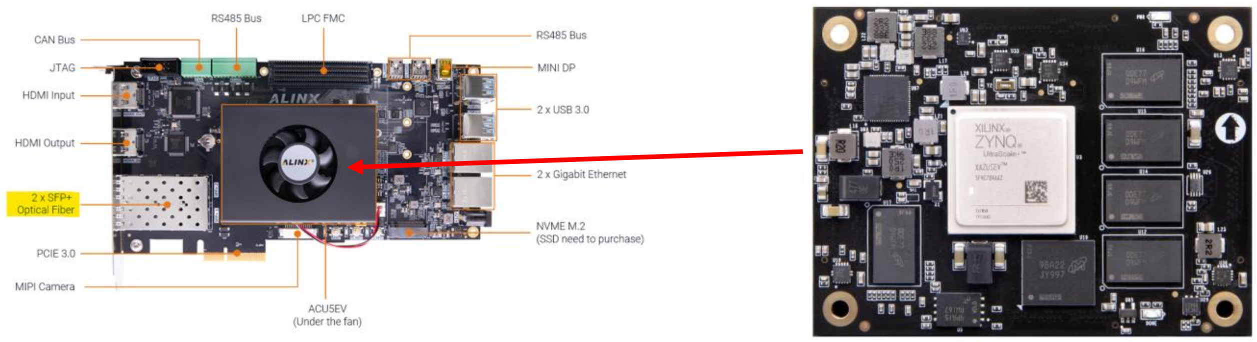


Figure 2-1-1: ACU5EV Core Board (Front View)

2. Functional Overview

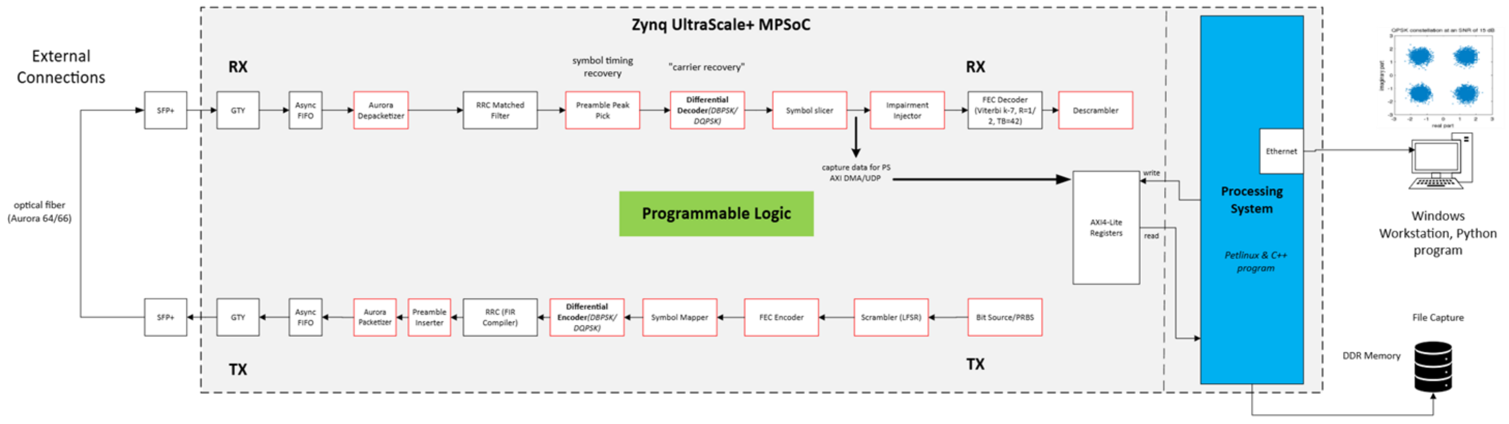
2.1 Capabilities

- QPSK/BPSK modulation/demodulation with RRC filtering.
- AMC FSM with SNR/EVM-based mode switching.
- Convolutional FEC encoding/decoding.
- Aurora optical loopback at 10.3125 Gb/s.
- Software registers for control/status.
- Constellation Plots + BER statistics.

2.2 Limitations

- Optical loopback only (no RF front-end).
- Only BPSK/QPSK modes.
- Hard-decision Viterbi only (soft decision out of scope).
- CLI-driven control (no GUI on PS).
- Snap-shot data visualization, not live stream

3. System-Level Block Diagram



4. Implementation Details

4.1 Transmit Path

4.1.1 PRBS Generator

Purpose:

Produce deterministic pseudo-random bits/bytes (PRBS) for bring-up, BER baselines, and end-to-end regression.

Description:

1. **Control/Config (AXI4-Lite):**
    - Software sets polynomial type (PRBS7/15/23/31), enables/disables output, writes seed, and optionally sets frame length (for TLAST).
    - Status bits show whether seed was loaded, running, and counters (bytes/bits emitted).
2. **LFSR Core (Shift Register):**
    - A 31-bit shift register produces one pseudo-random bit each clock.
    - Feedback is the XOR of two taps chosen by the polynomial.
    - If seed=0, it auto-corrects to 1 to avoid lock-up.
3. **Byte Packer (Optional):**
    - If enabled, groups 8 bits into a byte (TDATA[7:0]).
    - First generated bit goes into bit 0 (little-endian within the byte).
    - If not enabled, just outputs single bits on TDATA[0].
4. **Framing (Optional TLAST):**
    - A counter tracks how many bytes have been sent in the current frame.
    - When the count reaches FRAME_LEN_BYTES-1, it asserts TLAST.
    - Counter resets for the next frame.
5. **AXI-Stream Interface (Output):**
    - Provides TDATA, TVALID, TREADY, and optional TLAST.
    - If downstream deasserts TREADY, the generator stalls so the bitstream sequence is preserved (no skipped or duplicated bits).

Block Diagram:

Generalized LFSR structure below, taps will differ

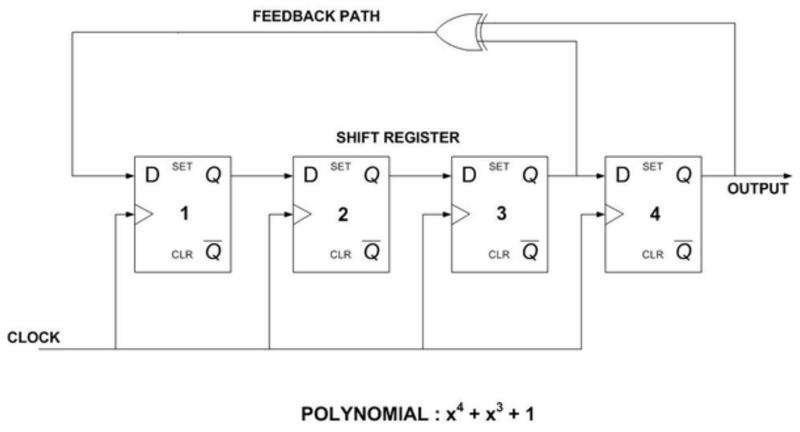


image source

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb` (baseband, e.g., 125 MHz), `rst_n` (sync to `clk_bb`).
- **AXI4-Lite (control/status):**
  - `CTRL`: `[0] enable`, `[3:1] poly_sel (PRBS7/15/23/31)`, `[4] pack_bytes`, `[5] tlast_en`
  - `SEED`: load value (width = longest poly, zero-guarded)
  - `FRAME_LEN_BYTES`: TLAST interval when `tlast_en=1`
  - `STATUS`: sticky `running`, `seed_loaded`, `underrun_err` (should never trip), `frame_bytes_rem`
- **AXI-Stream (master):**
  - If **bitstream mode**: `tdata[0]` (bit), `tvalid`, `tready`, `tlast` (optional on byte boundaries)
  - If **byte-packed mode**: `tdata[7:0]`, MSB last per spec below, `tvalid`, `tready`, `tlast` on `FRAME_LEN_BYTES-1`
- **Parameters (generics):**
  - `G_SUPPORT = {PRBS7, PRBS15, PRBS23, PRBS31}` (can trim to 7/15/31)
  - `G_DEFAULT_POLY = PRBS31`
  - `G_ENDIAN = LITTLE` (bit 0 first into byte packer)

Micro-Architecture:

LFSR Polynomials & Taps

- PRBS7: $x^7 + x^6 + 1$ (taps: 6,5)
- PRBS15: $x^{15} + x^{14} + 1$ (14,13)
- PRBS23: $x^{23} + x^{18} + 1$ (22,17)
- PRBS31: $x^{31} + x^{28} + 1$ (30,27)

Use a single 31-bit shift register; shorter polynomials mask off upper bits. Guard against all-zero state (on SEED write, if zero → force `1`).

1. **Core LFSR (1 bit/clk):**
   - `lfsr_next = {lfsr[N-2:0], feedback}`
   - `feedback = XOR(lfsr[tap_a], lfsr[tap_b])`
   - Output bit (before shift): `prbs_bit = lfsr[0]` (or MSB—just be consistent with your golden vectors)
   - When `enable=0`, hold state; when `seed_wr=1`, load `SEED & mask`.
2. **Bit → Byte Packer (optional):**
   - Shift in `prbs_bit` each cycle; after 8 bits, present a byte on AXIS.
   - **Bit ordering (recommended):** first generated bit goes to bit 0 of `tdata[7:0]` (little-endian within the byte). Document this—your testbench and PC tools must match.
   - If downstream deasserts `tready`, **stall the entire generator** to avoid dropping/duplicating bits (valid/ready back-pressure).
3. **Framing (optional TLAST):**
   - Decrement `frame_bytes_rem` on each accepted byte.
   - Assert `tlast` on last byte; reload counter to `FRAME_LEN_BYTES` next cycle when `tvalid & tready`.
   - In **bitstream mode**, assert `tlast` every 8*`FRAME_LEN_BYTES` bits (or disable).
4. **Throughput:**
   - 1 bit/clk → 125 Mb/s at 125 MHz. With byte packing, 1 byte/8 clks → 15.625 MB/s. Plenty for baseband testing.
5. **Reset/Seed semantics:**
   - On `rst_n=0`: load a non-zero default seed (e.g., `0x1`), clear counters/flags.
   - On `SEED` write: latch, and arm a one-shot `seed_load` that takes effect on next cycle when `enable=0` (or immediately if you choose; just document).

Verification Plan:

- **Golden vector match (per polynomial):**
  - For each poly and a few seeds (e.g., `1`, `0x5A`, `0xACE1u`), generate N=4096 bits in MATLAB/Python and compare bit-exact.
- **Packetizer loopback:** PRBS bytes → packetizer → depacketizer → scoreboarding vs generator's expected sequence (align on frame boundary).

Definition of Done (DoD):

- Bit-exact match to golden vectors for PRBS7/15/31 across multiple seeds.
- AXIS protocol clean under random back-pressure (assertions hold).
- Framing (`tlast`) aligns to configured byte length with no drift.
- Seed behavior deterministic and documented (including all-zero guard).

- Hardware capture round-trips with **0 mismatches** over ≥1 MB in both direct-to-DMA and Aurora-loopback paths.
- Resource/timing: comfortably meets baseband Fmax (≤125 MHz), LUT/FF trivial.

4.1.2 Scrambler

Purpose:

Randomize bit sequences to break up long runs of 0/1, reduce spectral lines, and whiten data before FEC and modulation. Provides deterministic, reversible scrambling for BER and regression testing.

Description:

1. **Control/Config (AXI4-Lite):**
    - Software selects polynomial (or bypass).
    - Enable/disable control.
    - Status: running flag, bypass flag.
2. **LFSR Core:**
    - Same principle as PRBS but used as **scramble sequence generator**.
    - XORs PRBS bit with input data bit each clock.
    - Configurable polynomial (default IEEE 802.3 x^7 + x^4 + 1).
3. **Byte-Aligned Operation:**
    - Data arrives as AXI-Stream bytes.
    - Each bit in the byte is XORed with the scrambler sequence in-order (bit 0 first).
    - Keeps byte/word alignment deterministic.
4. **Bypass Mode:**
    - AXI4-Lite control bit bypasses scrambling (data out = data in).
    - Useful for debug and regression.
5. **AXI-Stream Interface (Input/Output):**
    - Input: AXIS bytes (from PRBS or packetizer).
    - Output: AXIS bytes scrambled.
    - Fully back-pressure compliant (stalls scrambler state advancement until tvalid & tready).

Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite (control/status):**
    - CTRL: [0] enable, [1] bypass, [3:2] poly_sel
    - STATUS: running, seed_loaded
- **AXI-Stream (slave → master):**
    - s_axis_tdata[7:0], s_axis_tvalid, s_axis_tready, s_axis_tlast
    - m_axis_tdata[7:0], m_axis_tvalid, m_axis_tready, m_axis_tlast
- **Parameters (generics):**
    - G_POLY = {x^7+x^4+1, x^15+x^14+1, …}

    - G_ENDIAN = LITTLE (bit 0 scrambled first within byte)

Micro-Architecture:

1. **Scrambler LFSR:**
    - Generates one pseudo-random bit each cycle.
    - XOR applied per input bit.
    - Held stable if back-pressured.
2. **Bitwise XOR Engine:**
    - For each incoming byte, process 8 cycles.
    - OR — implement an 8-bit parallel XOR for efficiency.
3. **Bypass Path:**
    - Simple mux selects between scrambled data and pass-through.
4. **State Handling:**
    - On reset: load default seed (non-zero).
    - Seed reload on AXI4-Lite write (like PRBS).

Verification Plan:

- RTL simulation/self-checking testbench
- ILA core
- PRBS → Scrambler → Descrambler → compare with original PRBS stream.
- Mode switch (bypass enable/disable) → confirm expected behavior.

Definition of Done (DoD):

- Scrambler matches golden software reference sequence bit-for-bit.
- Bypass mode verified in sim + hardware.
- Back-pressure handling preserves correct sequence.
- Integration with descrambler yields original data (zero mismatches over 1 MB).

4.1.3 FEC Encoder

Purpose:

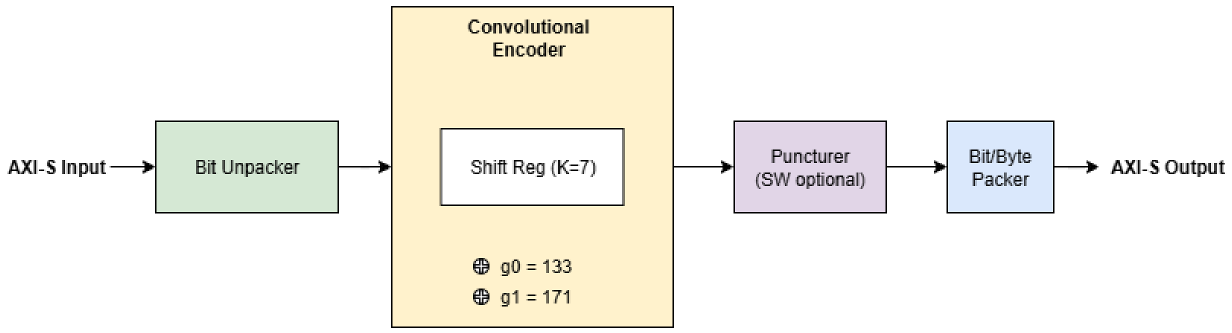
Apply forward error correction (rate-1/2 convolutional code, K=7) to improve link robustness and enable BER below uncoded limits. Encodes input bitstream into redundant output stream for recovery at receiver.

Description:

1. **Control/Config (AXI4-Lite):**
    - Enable/disable encoder.
    - Bypass mode for debug.

2. **Convolutional Encoder Core:**
   - Implements constraint length 7, generator polynomials $(171, 133)_8$.
   - Input: 1 bit per cycle.
   - Output: 2 coded bits per input bit.
3. **AXI-Stream Interface (Input/Output):**
   - Input: AXIS bytes.
   - Output: AXIS symbols (packed as bytes/words).
   - Maintains alignment; back-pressure halts state machine.
4. **Bypass Mode:**
   - Output data = input data (rate = 1:1).

Block Diagram:



Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite:** enable, bypass.
- **AXI-Stream:**
   - s_axis_tdata[7:0], s_axis_tvalid, s_axis_tready, s_axis_tlast
   - m_axis_tdata[15:0] (2 bits per input bit, packed)
- **Parameters (generics):** K=7, RATE=1/2, POLY={171,133}

Micro-Architecture:

1. **Shift Register (6-bit state + input):** Maintains history for constraint length 7.
2. **XOR Networks:** Compute parity outputs from generator polynomials.
3. **Output Formatter:** Packs 2 coded bits into byte stream (AXIS aligned).
4. **Bypass Path:** Direct input → output when bypass=1.

Verification Plan:

- Compare encoder output against MATLAB/`poly2trellis/convenc` golden vectors.
- Integration check: PRBS → Encoder → Viterbi (decoder) → compare to PRBS.

- Hardware bring-up: capture encoded data → decode offline in MATLAB/Python → confirm bit-exact.
- Bypass mode: confirm 1:1 passthrough.

Definition of Done (DoD):

- Output matches MATLAB golden vectors for multiple seeds and input lengths.
- Clean AXIS protocol under stalls.
- Encoder + Viterbi roundtrip reproduces input with 0 mismatches over ≥1 MB.
- Bypass mode confirmed.

4.1.4 Mapper

Purpose:

Map binary input bits into complex modulation symbols (BPSK or QPSK). Provides the digital baseband representation used for transmission over the optical Aurora link.

Description:

1. **Control/Config (AXI4-Lite):**
   - Select modulation: BPSK or QPSK.
   - Enable/disable module.
   - AMC override option (force mode regardless of AMC FSM).
2. **Mapping Logic:**
   - **BPSK:** maps `0 → +1`, `1 → -1` (real axis only).
   - **QPSK:** groups input bits into pairs; Gray-coded mapping to ±1 ± j1 (normalized).
3. **AXI-Stream Interface (Input/Output):**
   - Input: AXIS bits/bytes.
   - Output: AXIS symbols `{I[15:0], Q[15:0]}` fixed-point (Q1.15).
   - Supports back-pressure with deterministic symbol alignment.
4. **Bypass Mode:**
   - Optional passthrough (input bits → I only).

Block Diagram:

```
AXIS (bits) ──► [Bit Grouping]
               └► [Mapper: BPSK / QPSK LUT] ──► {I,Q} symbols (Q1.15) ──►
AXIS
```

Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite:** mode_select, enable, bypass, AMC override.
- **AXI-Stream:**

  - s_axis_tdata[7:0], s_axis_tvalid, s_axis_tready, s_axis_tlast
  - m_axis_tdata[31:0] = {I[15:0], Q[15:0]}, plus tvalid/tready/tlast
- **Parameters (generics):** G_FIXEDPT=Q1.15, G_MAPPING={Gray-coded}.

Micro-Architecture:

1. **Bit Grouping:** Collects 1 bit (BPSK) or 2 bits (QPSK).
2. **Mapping LUT:** Simple combinational logic for symbol assignment.
3. **Output Formatter:** Expands to signed fixed-point `{I,Q}`.
4. **Bypass Mux:** Directly routes input bits if bypass enabled.

Verification Plan:

- Compare output symbols against MATLAB constellation tables.
- PRBS → Mapper → check that constellation matches expected points.
- Hardware bring-up: capture symbols → plot in Python (constellation snapshot).

Definition of Done (DoD):

- BPSK/QPSK mapping verified against MATLAB/NumPy golden vectors.
- Symbol alignment deterministic under stalls.
- Roundtrip with slicer reproduces input bits with 0 mismatches.
- AMC override correctly forces selected mode.

4.1.5 Differential Encoder

Purpose:

Encode information in **phase differences** so the RX can recover data despite constant carrier phase offsets.

Description:

1. **Control/Config (AXI4-Lite):** enable; mode = **DBPSK** (1 bit) or **DQPSK** (2 bits).
2. **Core Function:** keep previous output symbol; map input bits → phase increment (0°,90°,180°,270°); output = previous_symbol × e^{jΔφ}.
3. **AXI-Stream:** In `{I,Q}` from Mapper; Out `{I,Q}` to RRC; honors back-pressure; propagates `tlast`.

Block Diagram:

```
AXIS {I,Q} ➔ [Phase Inc LUT] ➔ [Symbol State z^-1] ➔ [Complex Mult (prev ×
phase)] ➔ AXIS {I,Q}
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `mode` (DBPSK/DQPSK)
- **AXIS:** `s_axis_tdata[31:0]={I,Q}`, `m_axis_tdata[31:0]={I,Q}`, valid/ready/last
- **Params:** `G_FIXEDPT=Q1.15`, `G_INIT_SYM=(1,0)` (reset state)

Micro-Architecture:

1. **Phase LUT:** maps input bits to unit phasors (Q1.15).
2. **State Reg:** previous output symbol; reset to (1,0).
3. **Complex Multiplier:** fixed-point multiply with rounding/saturation.
4. **Control:** advance only on `tvalid && tready`; propagate `tlast`.

Verification Plan:

- Compare DBPSK/DQPSK output vs MATLAB/NumPy golden vectors.
- Loop: PRBS→Mapper→DiffEnc→(offline)DiffDec → expect bit-exact.
- Hardware capture: plot Δphase histogram; check only allowed increments.

Definition of Done (DoD):

- Golden-match for DBPSK/DQPSK; no overflow/saturation artifacts.
- AXIS clean under stalls; deterministic reset behavior.
- RX Differential Decoder recovers original bits losslessly.

4.1.6 RRC Filter (TX)

Purpose:

Apply **root-raised cosine** pulse shaping to limit bandwidth and control ISI before the link.

Description:

1. **Control/Config (AXI4-Lite):** enable/bypass; optional coeff reload.
2. **Core Function:** FIR with symmetric RRC taps (from `rcosdesign`), per-channel (I & Q). Optional **2× interpolation**.
3. **AXI-Stream:** In `{I,Q}` from DiffEnc; Out `{I,Q}` to Preamble Inserter; back-pressure compliant; propagate `tlast`.

Block Diagram:

```
AXIS {I,Q} ➔ [FIR (RRC) for I] ─┐
                   [FIR (RRC) for Q] ─┤──➔ AXIS {I,Q}
                                  [Bypass Mux]
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `bypass`, `(opt) coeff_wr`
- **AXIS:** `s_axis_tdata[31:0]={I,Q}`, `m_axis_tdata[31:0]={I,Q}`, valid/ready/last
- **Params:** `G_TAPS=48-64`, `G_ROLL_OFF=0.25`, `G_OSR=2`, `G_FIXEDPT=Q1.15`

Micro-Architecture:

1. **FIR Engines (I & Q):** Xilinx FIR Compiler IP; symmetric taps; internal scaling to avoid overflow; fixed latency L.
2. **Bypass Mux:** latency-aware path selection; `tuser/tlast` pipelined to match FIR latency.
3. **Flow Control:** counters advance only on `tvalid && tready`.

Verification Plan:

- Impulse/frequency response vs MATLAB RRC (within quantization).
- End-to-end eye opening improves post-filter; matched filter (RX) yields minimal ISI.
- Hardware capture: FFT of filtered symbols matches target roll-off.

Definition of Done (DoD):

- Response matches design; latency documented; no overflow.
- AXIS clean under stalls and bypass toggles.
- Matched TX/RX RRC produces expected ISI-free constellation at slicer.

4.1.7 Preamble Inserter

Purpose:

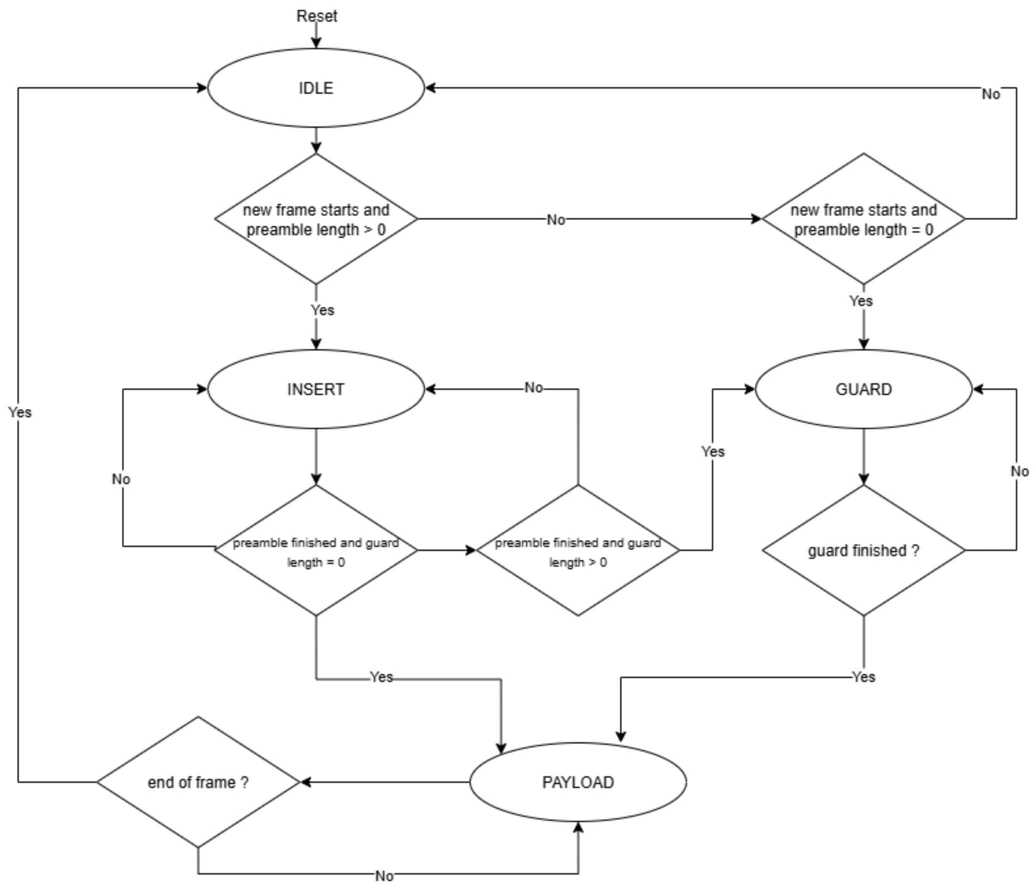
Insert a known symbol sequence at the start of each frame to enable RX timing acquisition (correlation-based frame sync) and provide a reference for EVM/SNR checks.

Description:

1. **Control/Config (AXI4-Lite):**
   - Enable/disable insertion; bypass option.
   - Select preamble length (e.g., 32/48/64 symbols).
   - Optional guard symbols after preamble (zeros or known pattern).
   - Optional `tuser` flagging for preamble samples.
2. **Core Function:**
   - On each frame start, output a ROM-stored complex preamble (I/Q) for `N` symbols.
   - After the preamble (and optional guard), pass through payload symbols unchanged.
   - Honors AXIS back-pressure so insertion never overruns downstream.
3. **AXI-Stream Interface (Input/Output):**
   - Input: AXIS `{I[15:0], Q[15:0]}` symbols (from RRC).

   - Output: AXIS `{I[15:0], Q[15:0]}` with preamble inserted at frame start.
   - `tlast` from upstream defines frame boundaries; `tuser[0]` optionally marks preamble.

Block Diagram (FSM):


Reset
IDLE
new frame starts and preamble length > 0
No
new frame starts and preamble length = 0
No
Yes
Yes
INSERT
No
GUARD
Yes
Yes
No
No
preamble finished and guard length = 0
preamble finished and guard length > 0
guard finished ?
No
Yes
Yes
end of frame ?
PAYLOAD
No

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`.
- **AXI4-Lite:** `enable`, `bypass`, `preamble_len`, `guard_len`, `(opt) scale`.
- **AXI-Stream:**
   - In: `s_axis_tdata[31:0]={I,Q}`, `s_axis_tvalid`, `s_axis_tready`, `s_axis_tlast`.
   - Out: `m_axis_tdata[31:0]={I,Q}`, `m_axis_tvalid`, `m_axis_tready`, `m_axis_tlast`, `(opt) m_axis_tuser[0]=preamble`.
- **Parameters (generics):**
   - `G_IQ_FMT = Q1.15`, `G_MAX_LEN = 64`, `G_GUARD_MAX = 16`.

   - `G_PREAMBLE_INIT[]`: compiled-in default sequence (can be ROM-initialized).

Micro-Architecture:

1. **FSM:** `IDLE` (waiting for frame start) → `INSERT` (emit `N` ROM symbols) → `GUARD` (emit zeros or pattern) → `PAYLOAD` (pass-through until `tlast`).
2. **ROM:** Dual-port or time-multiplexed ROM holding `{I,Q}` preamble samples.
3. **Mux & Flow Control:** Selects ROM vs payload; advances counters only on `tvalid && tready`.
4. **Frame Start Detect:** Uses upstream `tlast` (or a dedicated start signal) to arm the next preamble.
5. **Bypass Path:** Full pass-through with no timing modifications.

Verification Plan:

- Check correlation peak at RX using the chosen preamble (offline Python or FPGA correlator).
- Confirm `N` exact preamble symbols precede each payload frame; guard length honored.
- Back-pressure: stall during insertion and ensure no symbol duplication/loss.
- Bypass: output equals input; no added latency at frame boundaries.

Definition of Done (DoD):

- Exact `N`-symbol preamble inserted at every frame start; `tuser` flagging (if enabled) aligns with those samples.
- No AXIS protocol violations under stalls; payload boundaries preserved (`tlast` unchanged).
- Correlation in RX produces stable acquisition with deterministic offset.
- Bypass verified lossless.

4.1.8 Packetizer

Purpose:

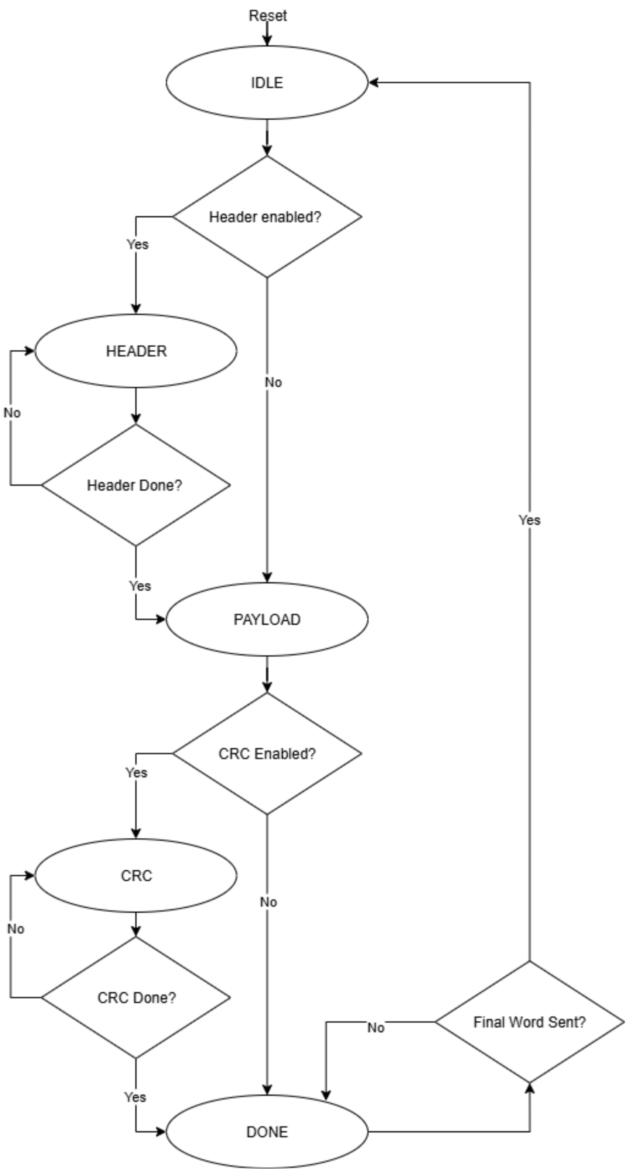
Encapsulate payload data (preamble + symbols) into packets with headers for synchronization, mode identification, and error checking. Provides structure for framing over Aurora and enables depacketization at RX.

Description:

1. **Control/Config (AXI4-Lite):**
   - Enable/disable; bypass option.
   - Header fields configurable: sync word, mode, sequence number, length, CRC, flags.

    - Optional CRC enable/disable.
2. **Core Function:**
    - At frame start, insert fixed header fields before payload.
    - Append CRC (if enabled) at frame end.
    - Maintains sequence counter for debug and BER stats.
    - Back-pressure compliant; header/payload emitted only on `tvalid && tready`.
3. **AXI-Stream Interface (Input/Output):**
    - Input: AXIS `{I,Q}` symbols (preamble + payload).
    - Output: AXIS with header prepended and CRC appended.
    - `tlast` marks end of packet; `tuser` optionally flags header region.

Block Diagram (FSM):



Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite:**
  - CTRL: enable, bypass, crc_en.
  - CFG: sync word, mode field, base sequence number, header format.
- **AXI-Stream:**
  - In: `{I,Q}`, valid/ready/last.
  - Out: `{I,Q}`, valid/ready/last, optional `tuser` header flag.

Micro-Architecture:

1. **FSM:** `IDLE → HEADER → PAYLOAD → CRC → IDLE.`
2. **Header Generator:** Concatenates sync + mode + seq + length + flags.
3. **CRC Engine (optional):** Running CRC computed over header+payload.
4. **Mux Logic:** Select header, then payload, then CRC onto output stream.
5. **Sequence Counter:** Auto-incremented on each packet.
6. **Bypass Path:** Payload forwarded unchanged if bypass=1.

Verification Plan:

- Packet capture → parse header fields in Python → confirm sync/mode/seq/len match configuration.
- CRC check passes in software for enabled runs; disabled runs skip CRC field.
- Roundtrip (Packetizer → Depacketizer) reproduces payload exactly.
- Back-pressure: no dropped/duplicated symbols.

Definition of Done (DoD):

- Correct header inserted with configurable fields.
- CRC validated vs software golden.
- Payload integrity verified through Packetizer+Depacketizer loop.
- AXIS compliance under stalls.
- Bypass confirmed lossless.

4.1.9 Aurora TX

Purpose:

Serialize packetized baseband symbols and transport them across the optical link using Aurora 64B/66B protocol at 10.3125 Gb/s. Provides a lightweight, high-throughput PHY/MAC layer between FPGA and SFP+ optics.

Description:

1. **Control/Config (AXI4-Lite):**

    - Enable/disable channel.
    - Lane/polarity config.
    - Status: channel_up, soft_err, hard_err.
2. **Core Function:**
    - Uses Xilinx Aurora 64B/66B IP core.
    - Accepts AXIS packets from Packetizer.
    - Encodes into 64B/66B blocks, serializes via GTH transceiver.
    - Drives SFP+ module over 10.3125 Gb/s optical link.
3. **AXI-Stream Interface (Input):**
    - Input: AXIS `{I,Q}` packet stream.
    - Handshake compliant; back-pressure applied if Aurora not ready.

Interfaces & Parameters:

- **Clock/Reset:**
    - `clk_aurora = 156.25 MHz` (from recovered PLL).
    - Async FIFO bridges between `clk_bb` and `clk_aurora`.
- **AXI4-Lite:** channel enable, lane config, error/status readback.
- **AXI-Stream:**
    - In: `{I,Q}`, valid/ready/last.
- **External:**
    - SFP+ cage, optical fiber.
- **Parameters (generics):**
    - Lane width (e.g., 64b).
    - GEARBOX factor for serialization.

Micro-Architecture:

1. **Async FIFO (CDC):** Buffers between baseband clock domain and Aurora core.
2. **Aurora Core (Xilinx IP):** Handles 64B/66B line coding, channel bonding, error detection.
3. **GTH Wrapper:** Serializes encoded stream to 10.3125 Gb/s.
4. **Link Management:** Monitors `channel_up`, resets on error.

Verification Plan:

- Internal loopback: TX → RX within Aurora IP → confirm payload integrity.
- ILA probes: channel_up, error counters stable during run.
- Packetizer → Aurora TX → Aurora RX → Depacketizer → compare payloads.
- Hardware bring-up: capture decoded packets in DDR, check vs transmitted PRBS.

Definition of Done (DoD):

- Aurora link comes up (`channel_up=1`) and stays stable.
- Internal loopback shows 0 packet mismatches.
- End-to-end payload integrity confirmed through Aurora link.

- Error counters remain 0 under normal conditions.

4.2 Receive Path

4.2.1 Aurora RX

Purpose:

Receive serialized packets from the optical link via Aurora 64B/66B protocol. Recovers clock, deserializes data, and outputs AXIS packet stream for downstream baseband processing.

Description:

1. **Control/Config (AXI4-Lite):**
   - Enable/disable channel.
   - Status: `channel_up`, `soft_err`, `hard_err`.
   - Reset/retrain control.
2. **Core Function:**
   - Uses Xilinx Aurora 64B/66B IP core.
   - GTH transceiver recovers serial clock and data at 10.3125 Gb/s.
   - Aurora core decodes 64B/66B stream → AXIS words.
   - Provides error detection signals for monitoring.
3. **AXI-Stream Interface (Output):**
   - Output: AXIS `{I,Q}` packetized symbols.
   - Propagates `tlast` to mark end of packet.
   - Back-pressure supported; upstream Aurora core applies stall.

Block Diagram:

Interfaces & Parameters:

- **Clock/Reset:**
  - `clk_aurora = 156.25 MHz` recovered.
  - Async FIFO bridges Aurora domain → baseband domain.
- **AXI4-Lite:** enable, reset, status readback.
- **AXI-Stream:**
  - Out: `{I,Q}` packet stream, valid/ready/last.
- **External:**
  - SFP+ module input via OM3 fiber.
- **Parameters (generics):**
  - Lane width (e.g., 64b).
  - GEARBOX factor for deserialization.

Micro-Architecture:

1. **GTH Receiver:** recovers clock/data at 10.3125 Gb/s and forwards to Aurora
2. **Aurora 64B/66B core** performs 66b block alignment using sync headers and error detection.
3. **Async FIFO:** Crosses from Aurora clock → baseband clock.
4. **AXIS Interface:** Outputs clean packetized symbols to Depacketizer.

Verification Plan:

- Internal loopback (Aurora TX→RX) confirms packet integrity.
- Monitor `channel_up` stable, error counters remain 0.
- End-to-end: PRBS → Packetizer → Aurora TX→RX → Depacketizer → compare with PRBS.
- Hardware capture: 0 mismatches over ≥1 MB.

Definition of Done (DoD):

- Link reliably establishes (`channel_up=1`).
- AXIS packets emitted with correct framing and no corruption.
- Stable operation under long runs with no errors.
- End-to-end PRBS test through link passes with 0 mismatches.

4.2.2 Depacketizer

Purpose:

Parse incoming packets, validate the header/CRC, and recover the raw payload (preamble + symbols) for baseband RX processing.

Description:

1. **Control/Config (AXI4-Lite):** enable/bypass; expected sync word; CRC enable; (opt) drop-on-error vs pass-with-flag.
2. **Core Function:** detects sync, parses header (mode, seq, length, flags), checks CRC (if enabled), strips header/CRC, outputs payload only.
3. **AXI-Stream:** In: AXIS packets from Aurora RX. Out: AXIS payload stream with preserved `tlast`; optional `tuser` flags for header/preamble regions.

Block Diagram:

```
AXIS Packets ➔ [Sync Detect + Header Parser] ➔ [Len/Seq Checks]
               └➔ [CRC Checker (opt)] ➔ [Error Flags / Policy]
               └➔ [Mux: Payload Only] ➔ AXIS Payload
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `bypass`, `crc_en`, `sync_word`, `(opt) drop_on_err`, `(opt) expected_mode`
- **AXI-Stream:**
  - o In: `s_axis_tdata[31:0]={I,Q}`, `tvalid`, `tready`, `tlast`
  - o Out: `m_axis_tdata[31:0]={I,Q}`, `tvalid`, `tready`, `tlast`, `(opt) tuser[0]=header/preamble`
- **Params:** header layout (field widths), CRC polynomial (default 0x1021)

Micro-Architecture:

1. **FSM:** `SEARCH_SYNC → READ_HDR → PAYLOAD → (CRC)`; advances only on `tvalid && tready`.
2. **Header Parser:** accumulates fixed header length; extracts `mode/seq/len/flags`.
3. **Length/Seq Trackers:** enforce payload length; (opt) track sequence gaps.
4. **CRC Engine (opt):** streaming CRC over header+payload; compares on trailer.
5. **Policy Mux:** drop or forward with error flag on header/CRC failure.
6. **Bypass Path:** pass input to output unchanged.

Verification Plan:

- Packetizer→Aurora→Depacketizer loop: payload equals original (0 mismatches).
- CRC enabled: software recompute matches hardware pass/fail; disabled path skips check.
- Error injection: bad sync/len/CRC → module raises flag and applies configured policy.
- Back-pressure: no symbol duplication/loss; `tlast` aligns with payload end.

Definition of Done (DoD):

- Correct sync detect and header parse across runs; payload recovered exactly.
- CRC check behavior correct; policy respected (drop vs flag).
- AXIS protocol clean under stalls; `tlast` preserved.
- Seamless loop with Packetizer (end-to-end integrity).

4.2.3 RRC Matched Filter (RX)

Purpose:

Apply receive-side root-raised cosine filtering matched to TX pulse shape to maximize SNR and minimize ISI before timing/phase recovery.

Description:

1. **Control/Config (AXI4-Lite):** enable/bypass; optional coeff reload; decimation/OSR select (if applicable).
2. **Core Function:** FIR with symmetric RRC taps (same roll-off as TX), per-channel on I and Q; optional decimation to symbol rate.

3. **AXI-Stream:** In `{I,Q}` from Depacketizer; Out `{I,Q}` to Preamble Correlator; honors back-pressure and propagates `tlast`.

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `bypass`, `(opt) coeff_wr`, `(opt) osr_sel`
- **AXIS:** `s_axis_tdata[31:0]={I,Q}`, `m_axis_tdata[31:0]={I,Q}`, `tvalid/tready/tlast`
- **Params:** `G_TAPS=48-64`, `G_ROLL_OFF=0.25`, `G_OSR=2`, `G_FIXEDPT=Q1.15`

Micro-Architecture:

- Dual FIR engines (I/Q) (Xilinx IP), fixed latency; optional decimator; latency-align `tlast/tuser`; bypass path with latency compensation.

Verification Plan:

- Impulse/frequency response ≈ MATLAB RRC; TX+RX cascade ≈ Nyquist.
- End-to-end constellation tightens after matched filtering.
- AXIS clean under stalls; bypass toggles safe.

Definition of Done (DoD):

- Response matches design within quantization; latency documented.
- Clean AXIS; improved EVM vs pre-filter capture.

4.2.4 Preamble Correlator

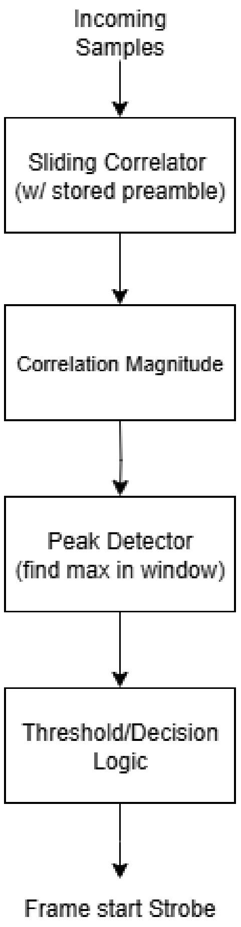
Purpose:

Detect the inserted preamble, establish frame/timing alignment, and provide a reliable strobe (and optional coarse CFO/phase estimate) for downstream demodulation.

Description:

1. **Control/Config (AXI4-Lite):** enable; threshold; preamble length; (opt) max offset search window; (opt) output `tuser` strobe.
2. **Core Function:** Sliding complex correlation of `{I,Q}` stream against stored preamble; find peak → emit `sym_strobe` and frame start index; (opt) compute peak phase for coarse rotation.
3. **AXI-Stream:** In `{I,Q}` from RX RRC; Out `{I,Q}` to Differential Decoder; `tuser` can mark symbol timing; `tlast` propagated.

Block Diagram:



Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `thresh`, `preamble_len`, `(opt) win_len`, `(opt) coarse_rot_en`
- **AXIS:** `s_axis_tdata[31:0]={I,Q}`, `m_axis_tdata[31:0]={I,Q}`, `tvalid/tready/tlast`, `(opt) tuser`
- **Params:** `G_MAX_LEN=64`, fixed-point widths for correlator accumulation

Micro-Architecture:

- Complex MAC window with running sum; magnitude^2 comparator for peak; optional CORDIC/atan2 for peak phase; generates one-cycle `sym_strobe` and frame-start flag; hold-off to avoid multiple detections per frame.

Verification Plan:

- Inject known preamble → single strong correlation peak at correct offset.
- Vary SNR → verify detection probability vs threshold; false-alarm rate low.
- If coarse rotator enabled, measured residual phase ≪ slicer decision angle.

Definition of Done (DoD):

- Deterministic, single detection per frame with correct alignment.
- Stable strobe aligning payload symbols; optional coarse rotation reduces phase error.
- Clean AXIS behavior; no payload corruption.

4.2.5 Differential Decoder

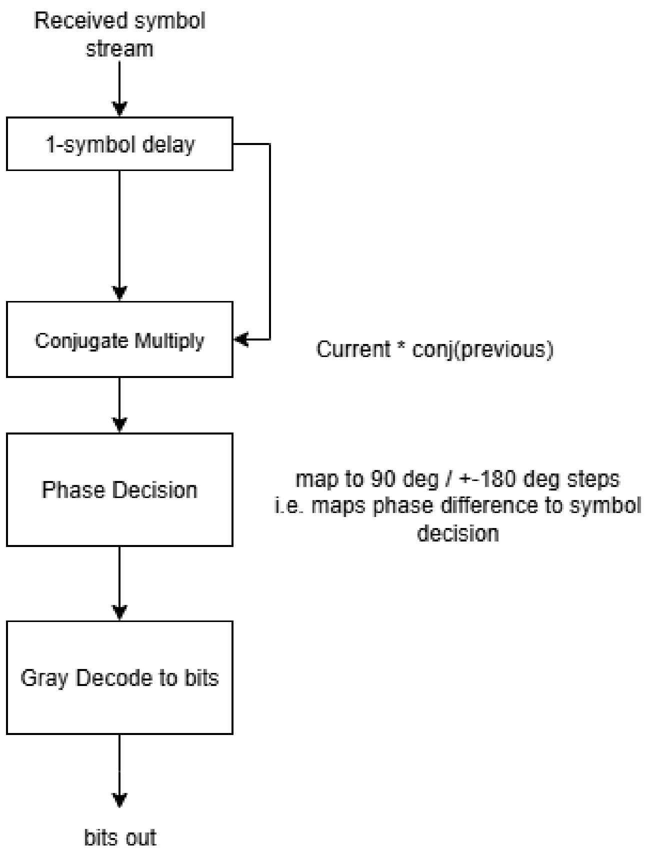
Purpose:

Recover bits from **phase differences** between successive symbols to remove absolute carrier phase ambiguity.

Description:

1. **Control/Config (AXI4-Lite):**
   Enable/disable differential decoding.
   Mode select: DBPSK (1 bit/sym) or DQPSK (2 bits/sym).
2. **Core Logic:**
   Maintain a 1-symbol delay register for the previous received symbol.
   Compute d[k] = y[k] * conj(y[k-1]).
   Map the phase of d[k] to the nearest valid increment (0, ±90, 180 deg).
   Gray-decode that phase increment into 1 or 2 bits.
3. **AXI-Stream Interface (Input/Output):**
   Input: AXIS symbols {I,Q} from RX RRC / Preamble Correlator.
   Output (configurable): AXIS decision bits (packed) or AXIS {I,Q} of the differenced symbol for a downstream slicer.
   Handshake: tvalid/tready/tlast.

Block Diagram:



Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite:** mode_select (DBPSK/DQPSK), enable.
- **AXI-Stream:**
  - s_axis_tdata[31:0] = {I[15:0], Q[15:0]}
  - m_axis_tdata[31:0] = {I[15:0], Q[15:0]}
  - Standard handshake signals.
- **Parameters (generics):** G_FIXEDPT=Q1.15, G_MODES={DBPSK, DQPSK}.

Micro-Architecture:

- Symbol Delay: FIFO/SRL holding the previous symbol.
- Complex Multiply: current * conjugate(previous).
- Phase Detector: quadrant/atan2-style decision to nearest allowed phase step.
- Gray Decoder: maps detected phase step to 1–2 bits.
- Reset Handling: initialize previous symbol to (1,0).

Verification Plan:

- Golden compare vs MATLAB/NumPy for DBPSK/DQPSK.
- Check delta-phi histograms show only allowed phase steps.
- Confirm decoded bits match TX bitstream in clean conditions.

Definition of Done (DoD):

- Bit-exact on golden vectors; stable under stalls; deterministic reset at frame boundaries.

4.2.6 Slicer

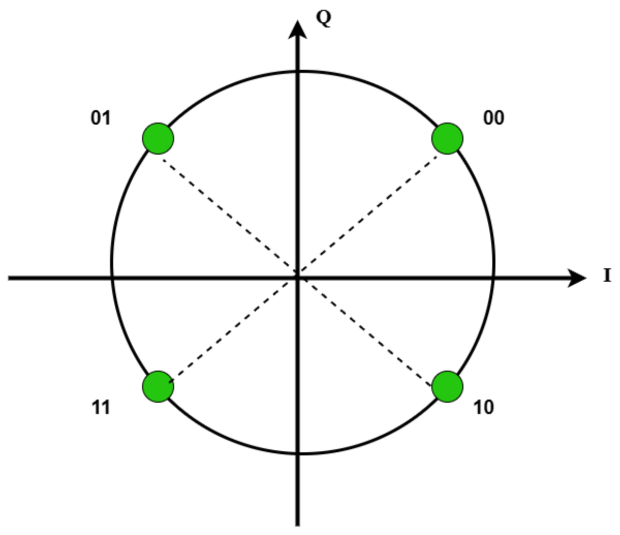
Purpose:

Make hard decisions on received constellation symbols, converting noisy `{I,Q}` values into estimated transmitted bits. Supports both **BPSK** and **QPSK (Gray-coded)** detection.

Description:

1. **Control/Config (AXI4-Lite):**
   - Mode select: BPSK or QPSK.
   - Enable/disable; bypass (optional).
2. **Core Function:**
   - **BPSK:** decision on the real axis (`I ≥ 0 → bit=0; I < 0 → bit=1`).
   - **QPSK:** quadrant-based Gray-coded mapping to 2 bits per symbol.
   - Threshold at 0 for both I and Q.
3. **AXI-Stream Interface (Input/Output):**
   - Input: AXIS `{I,Q}` symbols (from Differential Decoder).
   - Output: AXIS bits (packed into bytes).
   - Honors back-pressure; propagates `tlast`.

Block Diagram (QPSK Constellation) :



Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`.
- **AXI4-Lite:** `mode` (BPSK/QPSK), `enable`, `bypass`.
- **AXI-Stream:**
  - In: `s_axis_tdata[31:0]={I,Q}`, `tvalid/tready/tlast`.
  - Out: `m_axis_tdata[7:0]` (bits packed), `tvalid/tready/tlast`.
- **Parameters (generics):**
  - `G_FIXEDPT=Q1.15`.
  - `G_ENDIAN=LITTLE` (bit order within output byte).

Micro-Architecture:

1. **Decision Logic:** sign check on I (BPSK) or both I/Q (QPSK).
2. **Gray Decoder:** maps quadrants to bit pairs (00,01,11,10).
3. **Bit Packer:** groups decisions into 8-bit words.
4. **Bypass Mux:** optional passthrough of raw symbols.

Verification Plan:

- PRBS → Mapper → DiffEnc → RRC → … → DiffDec → Slicer → compare bits vs PRBS golden.
- Constellation scatter with noise → decisions land in correct bit bins.
- Hardware bring-up: capture slicer output, compare with transmitted PRBS stream.

Definition of Done (DoD):

- Correct Gray-coded mapping confirmed against golden.
- Deterministic bit packing and alignment.
- Clean AXIS protocol under stalls.
- Full TX→RX loop recovers PRBS with 0 mismatches (no channel impairments).

4.2.7 Impairment Injector

Purpose:

Introduce controlled bit errors or bursts into the RX stream to validate **FEC performance** and BER counter operation under known error patterns.

Description:

1. **Control/Config (AXI4-Lite):**
   - Enable/disable injector.
   - Mode: random bit flips, fixed interval flips, or burst errors.
   - Parameters: error probability (p), interval (N), burst length (L).
2. **Core Function:**

    - Monitors incoming bitstream.
    - Based on configuration, flips bits (XOR with error mask).
    - Burst errors applied by consecutive flips.
3. **AXI-Stream Interface (Input/Output):**
    - Input: AXIS bits (from Slicer).
    - Output: AXIS bits with impairments injected.
    - Fully back-pressure compliant; propagates `tlast`.

Block Diagram:

```
AXIS Bits ➔ [Error Mask Generator: rand / counter / burst FSM]
           └➔ [XOR with Incoming Bits] ➔ AXIS Bits (impaired)
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`.
- **AXI4-Lite:**
    - CTRL: enable, mode (random/fixed/burst).
    - CFG: error_prob, interval, burst_len, seed.
- **AXI-Stream:**
    - In: `s_axis_tdata[7:0]`, `tvalid/tready/tlast`.
    - Out: `m_axis_tdata[7:0]`, `tvalid/tready/tlast`.
- **Parameters (generics):** RNG type (LFSR default), G_ENDIAN=LITTLE.

Micro-Architecture:

1. **RNG Core:** LFSR or PRBS-based random generator.
2. **Mask Logic:** generates single-bit flip, periodic flip, or burst window.
3. **XOR Stage:** applies mask to incoming data bits.
4. **Bypass Path:** forwards data unchanged if disabled.

Verification Plan:

- Random mode: measure injected BER vs configured p → matches expected rate.
- Interval mode: flips occur at exact N-th bit; burst mode: contiguous L-bit flips.
- End-to-end: PRBS → Encoder → Channel (with Injector) → Viterbi → compare BER vs baseline.

Definition of Done (DoD):

- Configurable impairment patterns injected correctly.
- BER counters reflect expected error rates.
- AXIS compliance under stalls.
- Bypass path confirmed lossless.

4.2.8 FEC Decoder (Viterbi)

Purpose:

Decode convolutionally encoded data (K=7, rate-1/2) using the Viterbi algorithm. Provides maximum-likelihood recovery of original bitstream, correcting errors introduced by the channel.

Description:

1. **Control/Config (AXI4-Lite):**
   - Enable/disable decoder.
   - Bypass mode for debug.
   - Configurable traceback depth (default: 5–7 × K).
2. **Core Function:**
   - Uses Xilinx Viterbi Decoder IP (hard-decision mode).
   - Input: coded bitstream (2 bits per input symbol).
   - Output: recovered 1-bit stream.
3. **AXI-Stream Interface (Input/Output):**
   - Input: AXIS packed coded bits.
   - Output: AXIS decoded bits (repacked into bytes).
   - Fully back-pressure compliant.
4. **Bypass Mode:**
   - Input data forwarded unchanged.

Block Diagram:

```
AXIS (coded bits) → [Branch Metric Calc] → [ACS / Trellis State Machine]
                  → [Survivor Memory] → [Traceback Unit] → AXIS (decoded
bits)
```

Interfaces & Parameters:

- **Clock/Reset:** clk_bb, rst_n.
- **AXI4-Lite:** enable, bypass, traceback depth.
- **AXI-Stream:**
  - s_axis_tdata[15:0], s_axis_tvalid, s_axis_tready, s_axis_tlast
  - m_axis_tdata[7:0], m_axis_tvalid, m_axis_tready, m_axis_tlast
- **Parameters (generics):** K=7, RATE=1/2, POLY={171,133}, TRACEBACK=42 (default).

Micro-Architecture:

1. **Branch Metric Unit:** Computes Hamming distance of received bits vs expected codewords.
2. **Add-Compare-Select (ACS) Units:** Update path metrics for all states.
3. **Survivor Memory:** Stores best paths.
4. **Traceback Logic:** Recovers most likely input sequence after delay.
5. **Bypass Path:** Simple passthrough mux.

Verification Plan:

- Golden vector comparison vs MATLAB `vitdec`.
- Integration: PRBS → Encoder → Viterbi → compare to PRBS.
- Hardware bring-up: capture → decode offline in MATLAB/Python → confirm bit-exact.
- Bypass mode: confirm passthrough.

Definition of Done (DoD):

- Decoder output matches MATLAB golden vectors across multiple input sequences.
- Roundtrip (Encoder → Decoder) recovers original bitstream with 0 mismatches over ≥1 MB.
- Stable under back-pressure and configurable traceback depth.
- Bypass verified.

4.2.9 Descrambler

Purpose:

Invert the TX scrambling to recover the original bitstream before BER/FEC/metrics. Ensures deterministic, reversible whitening.

Description:

1. **Control/Config (AXI4-Lite):** enable/disable; bypass; polynomial select; seed load.
2. **Core Function:** generate scramble sequence (LFSR) and **XOR** with incoming bits (bit-for-bit inverse of TX).
3. **AXI-Stream:** In bits (from Viterbi or Slicer if uncoded); Out bits to BER/EVM; honors back-pressure; propagates `tlast`.

Block Diagram:

```
AXIS Bits ➔ [LFSR (same poly/seed as TX)] ➔ [Bitwise XOR] ➔ AXIS Bits
(descrambled)
                                           └➔ [Bypass Mux]
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite:** `enable`, `bypass`, `poly_sel`, `seed`
- **AXIS:** `s_axis_tdata[7:0]`, `m_axis_tdata[7:0]`, `tvalid/tready/tlast`
- **Params:** supported polynomials (e.g., 802.3 x^7+x^4+1), `G_ENDIAN=LITTLE`

Micro-Architecture:

- LFSR advances **only when `tvalid && tready`**.

- 8-bit parallel XOR path (or serial 1-bit ×8 cycles) with deterministic bit order.
- Seed/zero-guard like TX; seed load when disabled for reproducibility.

Verification Plan:

- TX Scrambler → RX Descrambler round-trip = original PRBS (0 mismatches).
- Bypass equals input; stalls preserve alignment; multiple polys/seeds match golden.

Definition of Done (DoD):

4.2.10 BER Counters

Purpose:

Measure bit-error rate by comparing recovered bits to a locally regenerated reference (PRBS or CRC-verified payload). Provides quantitative link quality for demo.

Description:

1. **Control/Config (AXI4-Lite):** enable/clear; mode select (PRBS-referenced vs CRC-gated payload); window length; optional auto-resync.
2. **Core Function:** align to reference, XOR received vs reference, accumulate **error_count** and **bit_count**; expose instantaneous and cumulative BER.
3. **AXI-Stream:** In bits after descrambler; optional sideband from depacketizer (frame boundaries) for alignment.

Block Diagram:

```
AXIS Bits ➔ [Reference Generator (PRBS or payload gate)]
             └➔ [Aligner (frame/seed sync)] ➔ [XOR Compare]
             └➔ [Accumulators: bit_count, err_count] ➔ AXI-Lite Status (BER)
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI4-Lite (status/ctrl):** `enable`, `clear`, `mode`, `window_len`, `aligned`, `bit_count`, `err_count`, `(opt) ber_q16_16`
- **AXIS In:** `s_axis_tdata[7:0]`, `tvalid/tready/tlast`, `(opt) tuser.frame_start`
- **Params:** supported PRBS polynomials; alignment depth; resync policy

Micro-Architecture:

- **Reference path:** PRBS LFSR seeded from known frame/seed OR CRC-gated “trusted payload” mode.
- **Aligner:** uses frame boundary and/or small slip search to lock bit phase; optional auto-resync on high error burst.

- **Counters:** wide saturating counters; optional fixed-window latched snapshot for host read.

Verification Plan:

- Clean channel: BER→0; inject known error rates via Impairment Injector and verify measured BER ≈ expected (binomial tolerance).
- Resync behavior correct after intentional slip; windowed snapshots stable under back-pressure.

Definition of Done (DoD):

- Accurate BER within statistical bounds over configured windows.
- Robust alignment/resync across frames; status readable without races.
- AXIS protocol clean; no impact on data path timing.

4.2.11 SNR/EVM Estimator

Purpose:

Compute signal quality metrics for diagnostics and AMC decisions: **per-frame EVM%**, **SNR (dB)**, and optional symbol-error stats.

Description:

1. **Control/Config (AXI4-Lite):** enable; mode (BPSK/QPSK); window = per-frame or fixed-N symbols; (opt) decimation for stats rate.
2. **Core Function:** compare received decisions/symbols against ideal constellation; accumulate error power and signal power → derive **EVM%** and **SNR dB**; latch results per frame.
3. **AXI-Stream:** In `{I,Q}` (or post-slicer bits plus re-mapper to ideal); Out passes `{I,Q}` through unchanged; results readable via AXI-Lite.

Block Diagram:

```
AXIS {I,Q} ➔ [Ideal Mapper (mode)] ➔ [Error: e = x - x^]
             ↳➔ [Accumulators: Σ|e|², Σ|x^|², count] ➔ [EVM%, SNR dB]
             ↳➔ (Frame latch) ➔ AXI-Lite status
             ↳➔ Pass-through AXIS {I,Q}
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI-Lite:** `enable`, `mode`, `win_len`, `evm_q16_16`, `snr_q8_8`, `sym_cnt`, `valid_flag`, `overflow`
- **AXIS:** `s_axis_tdata[31:0]={I,Q}`, `m_axis_tdata[31:0]={I,Q}`, `tvalid/tready/tlast`

- **Params:** fixed-point widths for power sums; scale factors for Q1.15

Micro-Architecture:

- **Ideal Mapper:** regenerates reference constellation from slicer bits or mode select.
- **Error Metrics:** complex subtract → $|e|^2$; reference power $|\hat{x}|^2$.
- **Accumulators:** wide, saturating; gated by `tvalid&&tready`; latch on `tlast` (frame).

Metrics:
Let `x^[k]` be the ideal constellation point (from mode), `x[k]` the received symbol after equal processing.

- Error power: `Σ|e|² = Σ|x[k] - x^[k]|²`
- Ref power: `Σ|x^|² = Σ|x^[k]|²`
- **EVM%** = `100 · sqrt( Σ|e|² / Σ|x^|² )`
- **SNR (dB)** ≈ `10·log10( Σ|x^|² / Σ|e|² )`
  Latch results on frame `tlast`. Fixed-point accumulators sized to avoid overflow for max frame length.

Verification Plan:

- Noiseless run → EVM≈0%, SNR→∞ (bounded by quantization).
- Inject AWGN at known Es/N0 → measured SNR tracks within tolerance.
- Frame latching aligns with `tlast`; no counter rollovers for max frame.

Definition of Done (DoD):

- EVM/SNR within ±0.5 dB (typ) vs MATLAB over representative SNRs.
- Stable per-frame reporting; AXI-Lite readable without race conditions.
- Zero impact to payload timing (transparent pass-through).

4.2.12 AMC FSM

Purpose:

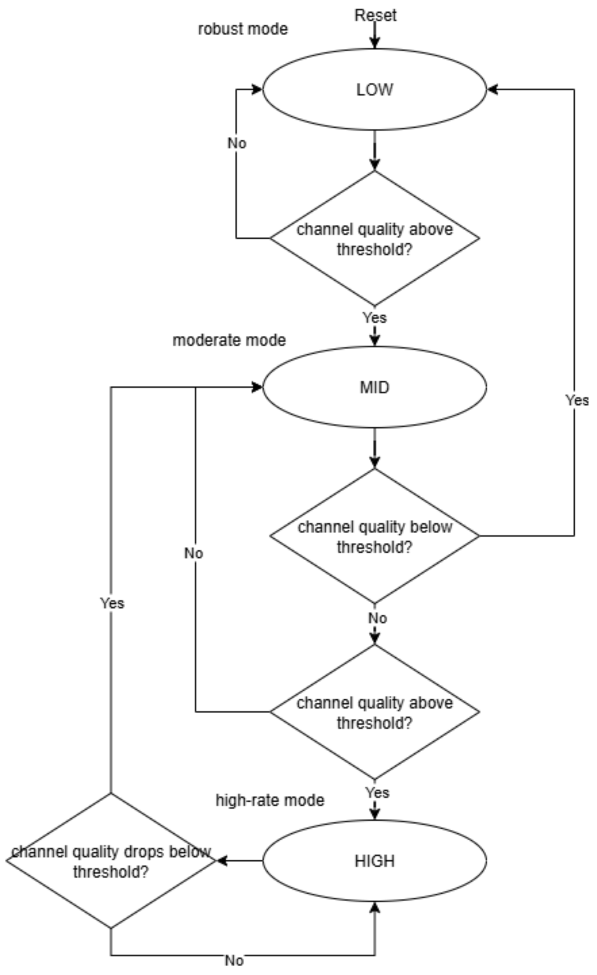
Select **modulation and coding mode** (e.g., {BPSK+FEC, QPSK+FEC, QPSK}) based on measured link quality with hysteresis and dwell to avoid mode flapping.

Description:

1. **Control/Config (AXI4-Lite):** thresholds for EVM/SNR; hysteresis margins; min dwell frames; enable/disable AMC; manual override.
2. **Core Function:** consume per-frame metrics → evaluate thresholds → update mode subject to dwell/hysteresis → publish current mode to TX Mapper/FEC and RX expectations.

3. **Interfaces:** No AXIS data path; AXI-Lite + sideband outputs (mode bus, change interrupt).

Block Diagram (FSM):


Reset
robust mode
LOW
No
channel quality above threshold?
Yes
moderate mode
MID
Yes
channel quality below threshold?
No
No
channel quality above threshold?
Yes
Yes
high-rate mode
HIGH
channel quality drops below threshold?
No

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI-Lite:** `enable`, `override_en`, `override_mode`, thresholds, hysteresis, dwell_frames, status (current_mode, last_change_frame)
- **Sideband Outputs:** `mode_out` to TX Mapper/FEC and to RX controls; `irq_mode_change`

Micro-Architecture:

- **Comparator Bank:** compares metrics to up/down thresholds.
- **FSM:** applies hysteresis; only transitions after `dwell_frames` satisfied.
- **Override Mux:** forces mode when enabled; logs reason codes.

Verification Plan:

- Sweep SNR across thresholds → verify transitions and no chatter (hysteresis).
- Dwell enforcement with oscillating SNR; override preempts FSM immediately.
- End-to-end: mode_out drives TX/RX consistently; no mismatches across a frame.

Definition of Done (DoD):

- Correct, stable mode transitions per configured thresholds and dwell.
- Override works and reports status; interrupts on change generated once per event.
- Clean handoff to TX/RX modules without pipeline hazards.

4.2.13 Constellation Snapshot

Purpose:

Capture periodic windows of `{I,Q}` symbols for host-side visualization (Python/pyqtgraph) without disturbing the data path.

Description:

1. **Control/Config (AXI4-Lite):** enable; trigger mode (periodic, manual, preamble-gated); decimation factor; capture length (samples).
2. **Core Function:** tap `{I,Q}` stream → decimate if configured → write samples into BRAM FIFO → AXI DMA burst to PS DDR → UDP to host.
3. **AXI-Stream:** Non-intrusive snoop of `{I,Q}`; main stream continues unmodified.

Block Diagram:

```
AXIS {I,Q} ➔ [Tap] ➔ (opt) [Decimator] ➔ [BRAM FIFO]
                               ↳ [AXI-DMA S2MM] ➔ PS DDR ➔ UDP → Host
(pyqtgraph)
```

Interfaces & Parameters:

- **Clock/Reset:** `clk_bb`, `rst_n`
- **AXI-Lite:** `enable`, `trig_mode`, `decim`, `cap_len`, `status (busy, frames, dropped)`
- **AXI-Stream Tap:** `s_axis_tdata[31:0]={I,Q}`, `tvalid/tready/tlast` (read-only tap)
- **AXI-DMA:** S2MM channel descriptors and completion flags (handled by PS software)
- **Params:** FIFO depth; max capture length; Q1.15 format

Micro-Architecture:

- **Tap & Decimator:** pick every `N`-th sample when `decim>1`; gate by preamble if selected.
- **Capture Controller:** finite state machine arms, counts samples, signals DMA start/stop.

- **Flow Control:** back-pressure never applied to main path; FIFO overflow flagged and capture aborted (status bit).

Verification Plan:

- Timing: trigger on preamble → snapshots start within known latency.
- Data integrity: captured I/Q matches online stream (spot-checked); decimation factor honored.
- End-to-end: UDP viewer renders constellation; no effect on RX BER when capturing.

Definition of Done (DoD):

- Reliable capture with zero impact to main data path; overflow conditions reported.
- PS app receives buffers and plots constellations correctly.
- Configurable triggers/decimation operate as specified.

4.1 IP vs Custom RTL

Module	Type	Notes
Transmit Path		
PRBS Generator	Custom RTL	AXI4-Lite ctrl, AXIS output; optional byte packer & TLAST
Scrambler	Custom RTL	LFSR XOR, byte-aligned
FEC Encoder	Custom RTL	Rate-1/2, K=7 convolutional $(171,133)_8$
Mapper	Custom RTL	BPSK/QPSK Gray; Q1.15 output
Differential Encoder	Custom RTL	DBPSK/DQPSK; 1-symbol state
RRC Filter (TX)	Mixed (IP + RTL wrapper)	FIR Compiler IP datapath; RTL wrapper for bypass, latency align
Preamble Inserter	Custom RTL	ROM sequence, guard, tuser flagging
Packetizer	Custom RTL	Sync/mode/seq/len/CRC header insertion
Aurora TX	IP Core	Aurora 64B/66B with GTH; custom CDC/FIFO wrappers
Receive Path		
Aurora RX	IP Core	Aurora 64B/66B RX; async FIFO to baseband domain
Depacketizer	Custom RTL	Header parse, CRC check, error policy
RRC Matched Filter (RX)	Mixed (IP + RTL wrapper)	FIR Compiler IP; wrapper adds bypass/decimation
Preamble Correlator	Custom RTL	Sliding correlation, peak detect, optional coarse rotator
Differential Decoder	Custom RTL	Undo DBPSK/DQPSK
Slicer	Custom RTL	Hard decision; Gray decode; bit packer
Impairment Injector	Custom RTL	Random/interval/burst bit flips
FEC Decoder (Viterbi)	IP Core	Xilinx Viterbi Decoder IP, hard-decision
Descrambler	Custom RTL	Inverse of scrambler, LFSR XOR

BER Counters	Custom RTL	PRBS/CRC-referenced error count
SNR/EVM Estimator	Custom RTL	Σ
AMC FSM	Custom RTL	Thresholds, hysteresis, dwell, override
Constellation Snapshot	Mixed (RTL + AXI DMA)	RTL tap/decimator + AXI DMA IP to PS DDR
Support / Infrastructure / Debug		
AXI DMA	IP Core	For constellation capture and host streaming
Clocking Wizard / MMCM	IP Core	125 MHz baseband, 156.25 MHz Aurora
AXI Interconnect / SmartConnect	IP Core	AXI4-Lite + AXIS fabrics
AXIS FIFOs / Clock Converters	IP Core	CDC & elasticity buffers
CRC Engine	Custom RTL	CRC-16-CCITT in Packetizer/Depacketizer (could use IP alternative)
ILA (debug)	IP Core	Debug only, not in final design

5. Interfaces

5.1 Internal (PL)

AXI4-Lite Control/Status Bus

- One AXI4-Lite slave per block (PRBS, Scrambler, Mapper, Packetizer, etc.).
- Runs in the **baseband clock domain (`clk_bb`, 125 MHz)**.
- Accessed from PS via AXI Interconnect.
- Each block includes:
  - `CTRL` register with enable/bypass/mode bits.
  - `STATUS` register with sticky flags (running, aligned, overflow).
  - **Software reset bit**: writing `1` generates a synchronous one-shot reset inside the block, equivalent to hardware reset for that domain.

AXI-Stream Data Fabrics

- Complex streams: `{I[15:0], Q[15:0]}` (32-bit).
- Bitstream paths: 8-bit wide, byte-packed.
- All AXIS paths handshake with `tvalid/tready/tlast`.
- CDC boundaries bridged with Async FIFOs.

DMA / Memory Interfaces

- AXI DMA S2MM: constellation snapshot → PS DDR.
- Optional MM2S for playback/testing.

Debug/Observability

- ILA cores on Aurora TX/RX, PRBS loopback, and key AXIS buses.
- Optional debug FIFO for pre-FEC capture.

Clock/Frequency Plan:

Domain	Name	Freq (MHz)	Source / Notes
Baseband	clk_bb	125	MMCM from 25 MHz ref; drives PRBS, Mapper, FEC, RRC, AMC FSM, etc.
Aurora User	clk_aur	156.25	GTY QPLL; required for Aurora 64B/66B @ 10.3125 Gb/s.
AXI-Lite	clk_axi	100	PS→PL AXI interconnect. Default processing system clock domain.
Debug/ILA	clk_dbg	same as debug domain	Optional high-speed debug domain for ILA/VIO probes.

CDC Boundaries

- Async FIFOs bridge `clk_bb` ↔ `clk_aur` (Packetizer/Depacketizer).
- AXI Interconnect bridges `clk_axi` ↔ `clk_bb`.

Reset Strategy

- **Global reset:** All PL custom blocks can be synchronously reset by writing `CTRL.GLOBAL_SW_RESET = 1` in the Global Reset/GPIO block (Appendix 10.1). This issues a one-shot reset pulse to every block's reset input, then auto-clears.
- **Per-block reset:** Each block also supports a local `CTRL.SW_RESET` bit in its own register map. This allows targeted resets without disturbing the full chain.
- **Aurora resets:** The Aurora TX/RX wrappers also accept resets via both the global tree and their own RETRAIN control bit for link-level retrain sequences.
- **AXI-Lite stability:** The AXI-Lite domain remains active during global reset, ensuring that software can reliably re-configure blocks after reset.

Note: tie GLOBAL_SW_RESET to each block's reset input (synchronously in clk_bb), and also OR it with the Aurora wrappers' local reset where appropriate. Keep the **AXI-Lite domain stable** during global reset so software can reliably deassert/verify.

5.2 External (PS/Optical)

Optical Link (Aurora over SFP+)

- **Physical:** SFP+ cages populated with 10GBASE-SR optical modules, connected via OM3 duplex LC fiber (1–3 m).
- **Protocol:** Aurora 64B/66B, single lane at 10.3125 Gb/s.
- **Clocks:** Refclk ~156.25 MHz from GTY QPLL, recovered at RX.

- **Signals:** All optical serialization/deserialization handled by Aurora IP and GTY transceivers; PL logic only sees AXI-Stream.
- **Status/Debug:** `channel_up`, `soft_err`, `hard_err` reported to PS via AXI4-Lite; optional Aurora ILA taps for bring-up.

PS Ethernet Interface

- **Purpose:** Used for UDP streaming of constellation snapshots and telemetry (BER/SNR/EVM/AMC state) to host PC.
- **Implementation:** Gigabit Ethernet from Zynq PS GEM MAC, routed to Linux (PetaLinux). C++ program handles UDP framing and sends to host.
- **Connection:** Direct link (Cat6 cable) or via switch; host runs Python + pyqtgraph for visualization.

PS UART / JTAG

- **UART:** USB-UART cable to PS console for Linux shell access and debug messages.
- **JTAG:** Used for FPGA programming and low-level debug during bring-up.

microSD Card

- **Purpose:** Boot media for PetaLinux image and configuration files.
- **Requirements:** ≥32 GB, UHS-I card recommended for reliability.

Resets and Control

- **Hardware resets** sourced from PS and exposed on board headers.
- **Software resets** triggered by AXI4-Lite control registers (per-block).
- **Aurora link retrain** controlled by PS register writes if channel errors are detected.

---

6. Software

6.1 PS Software (PetaLinux + CLI)

Purpose:
Provide a minimal software control layer running on the Processing System (PS) to configure PL blocks, arm captures, and dump results for host-side analysis. The emphasis is on simplicity and reproducibility.

Environment:

- PetaLinux (board boot image)
- User-space C/C++ programs compiled with GCC toolchain

- Access to PL registers via `/dev/mem` or UIO drivers
- Optional shell scripts to wrap common test flows

Core Responsibilities:

1. **Configuration of AXI4-Lite Registers**
   - Set PRBS polynomial, scrambler bypass, modulation mode, FEC enable, preamble length, AMC thresholds, etc.
   - Provide **software reset** by writing to the CTRL.reset bit in each block.
2. **Test Orchestration**
   - Arm PRBS generator, packetizer, Aurora channel.
   - Loopback modes (internal or optical) selectable by register writes.
   - Start/stop a frame run via one command.
3. **Capture Control**
   - Program Constellation Snapshot registers: destination DDR address, capture length, trigger mode.
   - Poll STATUS.done or interrupt flag.
   - Dump buffer to file (`cap_iq_i16.bin`) with a small user program.
4. **Status / Telemetry**
   - Read EVM/SNR metrics, BER counters, and AMC state via AXI4-Lite STATUS registers.
   - Print results to console or append to a log file.

CLI Commands (conceptual format)

- `sdr init` – Reset all blocks, apply default configuration.
- `sdr prbs --poly 31 --seed 1` – Configure PRBS generator.
- `sdr mode --mod qpsk --fec on --diff dqpsk` – Set modem mode.
- `sdr preamble --len 64` – Configure preamble inserter.
- `sdr start` / `sdr stop` – Enable or halt frame processing.
- `sdr capture --len 16384 --out cap_iq_i16.bin` – Arm DMA capture, dump file to PS filesystem.
- `sdr stats` – Print BER, SNR/EVM, AMC mode.

Implementation Notes:

- Access method: simplest is `/dev/mem` mapping to PL base addresses.
- Registers defined in **Appendix (10.1)**. Each block follows common layout (CTRL, STATUS, CFGx, RESULTx).
- File output: use standard Linux file I/O. Dump files to `/run/media/mmcblk0p1/` (microSD) for easy transfer.
- Optional: Provide simple shell scripts to bundle sequences (e.g., `run_qpsk_test.sh`).

Acceptance Criteria:

- All major PL blocks configurable and resettable via CLI.

- Able to start/stop PRBS traffic and confirm Aurora link stability.
- Successfully capture ≥16k symbols to file and transfer to host PC.
- Correctly read and display BER/SNR/EVM counters from hardware.

6.2 Host Tools (Python + Visualization)

Purpose:
Provide simple, low-risk visualization of constellation data using offline snapshots. No networking is required. This functionality will be used for demonstration purposes as well as for debug.

Workflow Overview:

1. On the board, the Constellation Snapshot block captures N complex samples via AXI-DMA into DDR.
2. A tiny PS userspace tool dumps the interleaved int16 I/Q buffer to a file (e.g., cap_iq_i16.bin).
3. Copy the file to the host PC (microSD, SCP, or USB/serial).
4. Plot locally in Python or MATLAB.

File Format (documented for repeatability):

- Type: little-endian int16, interleaved I, Q, I, Q, …
- Samples: N complex samples -> 2*N int16 values
- Scaling: Q1.15 fixed-point (full-scale approx ±32767)

Minimum Host Tooling:

- Python 3.x with numpy and matplotlib, or MATLAB R2024b+
- No GUI framework required (no PyQt/PySide); scripts are single-file

Python (constellation plot):

- Save as plot_cap.py on the host.
- Usage: python plot_cap.py cap_iq_i16.bin

```python
import sys, numpy as np
import matplotlib.pyplot as plt

if len(sys.argv) < 2:
    print("usage: python plot_cap.py cap_iq_i16.bin"); sys.exit(1)
fn = sys.argv[1]

raw = np.fromfile(fn, dtype='<i2')     # little-endian int16
```

```python
if raw.size % 2 != 0:
    raise RuntimeError("file length not even (I/Q pairs expected)")
iq  = raw.reshape(-1, 2).astype(np.float32)
I, Q = iq[:, 0], iq[:, 1]

plt.figure()
plt.scatter(I, Q, s=2)
plt.gca().set_aspect('equal', 'box')
plt.title(f'Constellation: {fn}')
plt.xlabel('I'); plt.ylabel('Q'); plt.grid(True)
plt.show()
```

MATLAB (constellation plot):

- Save as plot_cap.m on the host.
- Usage: plot_cap('cap_iq_i16.bin')

```matlab
function plot_cap(fname)
  fid = fopen(fname, 'rb');
  if fid < 0, error('cannot open %s', fname); end
  iq = fread(fid, [2, Inf], 'int16=>double'); fclose(fid);
  plot(iq(1,:), iq(2,:), '.'); axis equal; grid on;
  title(sprintf('Constellation: %s', fname));
  xlabel('I'); ylabel('Q');
end
```

Operational Notes:

- Capture size: 16k complex samples is a good default (fast to move, dense enough to see clusters).
- Transfer method: microSD is simplest; SCP is fastest if the board is on LAN.
- Versioning: include N, scaling (Q1.15), and modulation mode in the filename or a small sidecar .txt to avoid confusion (example: cap_qpsk_n16384_q115.bin).
- Reproducibility: seed the PRBS and document any RX processing (preamble align, differential decode) active during capture.

7. Materials & Tools

7.1 Hardware

- Alinx AXU5EVB-P (Zynq UltraScale+ MPSoC) board + 12 V PSU (obtained)
- 2× SFP+ 10GBASE-SR optical transceivers (e.g., FS brand)
- OM3 LC–LC duplex fiber patch (1–3 m)(fiber cable)
- microSD card (≥32 GB, UHS-I) for PetaLinux image
- USB-UART cable (board console)
- Ethernet (Cat6) cable for PS networking/UDP
- Cooling fan (over GTY area)

7.2 Software

- **FPGA / SoC tools**
  - AMD/Xilinx Vivado 2023.2
  - AMD/Xilinx Vitis 2023.2
  - PetaLinux SDK (matching version; build on Ubuntu 20.04/22.04)
- **Embedded (board side)**
  - PetaLinux image for AXU5EVB-P (PS Ethernet, UART enabled)
  - GCC/G++ toolchain in PetaLinux (for C++ UDP sender)
  - iperf/ethtool/net-tools (bring-up/debug)
- **Host PC (Windows) — Visualization & Debug**
  - Python 3.11+
    - pyqtgraph, numpy, PySide6 (or PyQt6), struct/socket (std lib)
  - VS Code
  - Wireshark (UDP sanity checks)
- **Analysis / Modeling**
  - MATLAB R2024b + Communications Toolbox (BER vs SNR, FEC golden vectors)
  - poly2trellis, convenc, vitdec, RRC tap design

- **IP cores (used in design)**
  - **Free AMD IP:** Aurora 64B/66B, FIR Compiler (RRC Tx/Rx), DDS Compiler (NCO), Viterbi Decoder (LogiCORE IP), CORDIC/Rotator, AXI DMA, AXI Interconnect, Clocking Wizard, AXI-Stream FIFO, ILA/VIO

---

7. Schedule & Development Plan

Week	Focus Area	Milestones / Deliverables
W1	**Bring-Up & PRBS**	Repo + TCL automation finalized. Clocks/resets stable. PRBS generator verified vs golden vectors. Aurora IP example synthesized, loopback channel_up=1.
W2	**Aurora Link Bring-Up**	Aurora TX/RX wrappers integrated with async FIFOs. Optical loopback working at 10.3125 Gb/s. ILA confirms packet framing.
W3	**Scrambler / Descrambler**	RTL complete + testbenches. Loopback test PRBS → Scrambler → Descrambler matches bit-exact.
W4	**Mapper + Differential Encoder**	BPSK/QPSK mapping validated. DBPSK/DQPSK encoding verified vs MATLAB/NumPy golden. TX chain runs through Aurora with stable constellations.
W5	**Preamble Inserter + Correlator**	TX preamble insertion. RX sliding correlator detects peak reliably at correct offset. Symbol strobe gates payload correctly.
W6	**FEC Integration**	Convolutional encoder + Viterbi IP wrapper integrated. Golden vectors confirmed. BER counters measure injected error rates correctly.
W7	**Metrics + AMC FSM**	SNR/EVM estimator validated vs MATLAB. AMC FSM thresholds/hysteresis exercised in sim and hardware. Stable mode transitions proven.
W8	**Constellation Capture & Demo Prep**	Snapshot + AXI-DMA → DDR path validated. PS userspace dump tool produces cap_iq_i16.bin. Host plots constellations offline. End-to-end optical demo runs with BER/EVM sweeps logged. Demo scripts + report finalized.

---

8. Risks & Mitigation

Risk	Mitigation

Aurora bring-up fails or is unstable	Start with Aurora internal loopback example; verify clocks/GTY settings early; use ILAs to monitor channel_up and AXIS traffic.
Clocking / CDC issues	Insert async FIFOs at domain boundaries; review with RTL simulation under random stalls; check timing closure reports.
AXI-Stream protocol bugs (back-pressure, TLAST alignment)	Self-checking RTL testbenches with random stall injection; add assertions; observe tvalid/tready/tlast with ILAs in hardware.
FEC integration errors	Use MATLAB golden vectors for convolutional encode/decode; verify with RTL simulation before hardware; confirm BER gain in loopback tests.
Metrics (BER/EVM/SNR) inaccurate	Cross-check against MATLAB/Python post-processing; validate with injected impairments in simulation; compare hardware results to software references.
AMC FSM unstable (flapping or wrong transitions)	Add hysteresis and dwell timers; verify FSM transitions with RTL simulation sweeps; log state transitions via ILAs and PS reads.
Resource / timing closure problems	Target moderate baseband clock (125 MHz); pipeline long datapaths; check incremental synthesis results early.
Schedule risk / integration delays	Keep internal loopback path as fallback for end-to-end validation; prioritize risky blocks (Aurora, DMA) early.

9. References

- Alinx Technology. (2023). AXU5EVB-P User Manual and Schematics. Retrieved from https://www.en.alinx.com/Product/SoC-Development-Boards/Zynq-UltraScale-plus-MPSoC/AXU5EVB-P.html
- AMD/Xilinx. (2023). Aurora 64B/66B LogiCORE IP Product Guide (PG074). San Jose, CA: Advanced Micro Devices, Inc.
- AMD/Xilinx. (2023). Zynq UltraScale+ MPSoC Technical Reference Manual (UG1085). San Jose, CA: Advanced Micro Devices, Inc.
- AMD/Xilinx. (2023). FIR Compiler LogiCORE IP Product Guide (PG149). San Jose, CA: Advanced Micro Devices, Inc.
- AMD/Xilinx. (2023). Viterbi Decoder LogiCORE IP Product Guide (PG109). San Jose, CA: Advanced Micro Devices, Inc.
- AMD/Xilinx. (2023). Clocking Wizard LogiCORE IP Product Guide (PG065). San Jose, CA: Advanced Micro Devices, Inc.
- MathWorks. (2024). Communications Toolbox Documentation (poly2trellis, convenc, vitdec, rcosdesign, awgn). Retrieved from https://www.mathworks.com/help/comm
- Proakis, J. G. (2008). Digital Communications (5th ed.). New York, NY: McGraw-Hill.
- Sklar, B. (2001). Digital Communications: Fundamentals and Applications (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

- Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, 13(2), 260–269. https://doi.org/10.1109/TIT.1967.1054010
- PetaLinux Tools Documentation (https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/embedded-software/petalinux-sdk.html)
- Alinx AXU5EV-P Github: https://github.com/awozjhu/AXU4EV-P_AXU5EV-P/tree/master
- Python docs: socket (UDP), PySide/PyQt; **pyqtgraph** documentation (real-time plotting). (https://docs.python.org/3/)

---

10. Appendix

10.1 Draft AXI4-Lite Register Map

10.1.1 Common layout (per block, offsets from that block's BASE)

Offset	Name	R/W	Description
0x00	**CTRL**	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET (self-clears), [6:4] MODE, [7] START, [15] CLEAR (counters) — bits unused by a given block are ignored
0x04	**STATUS**	R/W1C	[0] RUNNING, [1] ALIGNED/LOCKED (if applicable), [2] OVERFLOW, [3] UNDERFLOW, [4] ERROR, [8] DONE (for one-shot ops)
0x08	**CFG0**	R/W	Block-specific configuration (e.g., lengths, thresholds)
0x0C	**CFG1**	R/W	Block-specific configuration
0x10	**CFG2**	R/W	Block-specific configuration
0x14	**CFG3**	R/W	Block-specific configuration
0x18	**RESULT0**	RO	Latched result (e.g., counters/metrics)
0x1C	**RESULT1**	RO	Latched result
0x20	**RESULT2**	RO	Latched result
0x24	**RESULT3**	RO	Latched result

Notes:

* **SW_RESET** asserts the block's internal reset for ≥1 clk_bb cycle, then auto-clears. Works even if ENABLE=0.

* **START** is used by one-shot engines (e.g., capture) and can share the CTRL word with ENABLE.

* **CLEAR** zeroes counters/accumulators synchronously at the next safe boundary (e.g., frame end).

10.1.2 Base addresses

Block	BASE
Global Reset / GPIO	0xA001_6000
PRBS Generator	0xA000_0000
Scrambler / Descrambler	0xA000_1000 / 0xA000_7000
FEC Encoder	0xA000_2000

Mapper (TX)	0xA000_3000
Differential Encoder (TX)	0xA000_4000
RRC Filter TX / RX	0xA000_5000 / 0xA000_9000
Preamble Inserter (TX)	0xA000_6000
Packetizer / Depacketizer	0xA000_8000 / 0xA000_A000
Aurora TX / RX (status wrap)	0xA000_B000 / 0xA000_C000
Preamble Correlator (RX)	0xA000_D000
Differential Decoder (RX)	0xA000_E000
Slicer	0xA000_F000
Impairment Injector	0xA001_0000
Viterbi (IP wrapper)	0xA001_1000
BER Counters	0xA001_2000
SNR/EVM Estimator	0xA001_3000
AMC FSM	0xA001_4000
Constellation Snapshot	0xA001_5000

10.1.3 Block-specific fields

Note: Register Access Abbreviations (R/W column):

R = Read-only
W = Write-only
R/W = Read/Write
RO = Read-only (same as R, but used for emphasis on results/counters)
R/W1C = Readable, *Write-1-to-Clear* (writing 1 clears that bit, writing 0 has no effect)

Global Reset / GPIO (BASE: 0xA001_6000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] GLOBAL_SW_RESET (one-shot, self-clears)
0x04	GPIO_OUT	R/W	32 user outputs (LEDs, test selects, optional Aurora retrain line)
0x08	GPIO_IN	RO	32 user inputs
0x0C	STATUS	RO	Optional sticky flags (impl-defined)

PRBS Generator (BASE: 0xA000_0000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [6:4] MODE (0=PRBS7,1=15,2=23,3=31), [15] CLEAR
0x04	STATUS	R/W1C	[0] RUNNING, [2] OVER/UNDERFLOW (diag), [8] DONE (frame)
0x08	SEED (CFG0)	R/W	31:0 seed (0 coerced to 1)
0x0C	FRAME_LEN_BYTES (CFG1)	R/W	TLAST interval (byte-packed mode)
0x18	BYTE_COUNT (RESULT0)	RO	Accepted bytes
0x1C	BIT_COUNT (RESULT1)	RO	Accepted bits

Scrambler (BASE: 0xA000_1000) / Descrambler (BASE: 0xA000_7000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [6:4] MODE (poly select)
0x04	STATUS	R/W1C	[0] RUNNING, [2] OVERFLOW (should be 0)
0x08	SEED (CFG0)	R/W	Non-zero seed
0x0C	CFG1	R/W	Impl-specific (e.g., bit order)

FEC Encoder (BASE: 0xA000_2000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [6:4] MODE (future puncture)
0x04	STATUS	R/W1C	[0] RUNNING
0x18	OUT_BITS (RESULT0)	RO	Encoded bits sent
0x1C	IN_BITS (RESULT1)	RO	Input bits consumed

Mapper (TX) (BASE: 0xA000_3000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [6:4] MODE (0=BPSK,1=QPSK), [8] AMC_OVERRIDE
0x04	STATUS	R/W1C	[0] RUNNING, [2] OVERFLOW (should be 0)

Differential Encoder (TX) (BASE: 0xA000_4000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET (prev ← (1,0)), [6:4] MODE (0=DBPSK,1=DQPSK)
0x04	STATUS	R/W1C	[0] RUNNING

RRC Filter TX (BASE: 0xA000_5000) / RX (BASE: 0xA000_9000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET
0x04	STATUS	R/W1C	[2] OVERFLOW, [8] DONE (coeff reload)
0x08	CFG0	R/W	Coeff set ID / scale
0x0C	CFG1	R/W	OSR/decimation (RX)
0x10	CFG2	R/W	Reserved
0x14	CFG3	R/W	Reserved

Preamble Inserter (TX) (BASE: 0xA000_6000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [7] START (arm next frame)

0x04	STATUS	R/W1C	[8] DONE (insert finished)
0x08	PREAMBLE_LEN (CFG0)	R/W	Symbols (32–64)
0x0C	GUARD_LEN (CFG1)	R/W	Symbols
0x10	FLAGS (CFG2)	R/W	Bit0: mark tuser on preamble

Packetizer (BASE: 0xA000_8000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [8] CRC_EN
0x04	STATUS	R/W1C	[0] RUNNING, [4] ERROR
0x08	SYNC_WORD (CFG0)	R/W	16-bit sync (placed in header)
0x0C	MODE_DEFAULT (CFG1)	R/W	Mode field default
0x10	SEQ_BASE (CFG2)	R/W	Start sequence
0x14	HEADER_FMT (CFG3)	R/W	Field widths/flags

Aurora TX / RX Status Wrappers (BASE: 0xA000_B000 / 0xA000_C000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [8] RETRAIN (pulse Aurora reset)
0x04	STATUS	R/W1C	[0] CHANNEL_UP, [1] SOFT_ERR, [2] HARD_ERR
0x18	SOFT_ERR_CNT (RESULT0)	RO	Soft errors
0x1C	HARD_ERR_CNT (RESULT1)	RO	Hard errors

Depacketizer (BASE: 0xA000_A000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [8] CRC_EN, [9] DROP_ON_ERR
0x04	STATUS	R/W1C	[4] ERROR, [8] DONE (packet)
0x08	EXPECTED_SYNC (CFG0)	R/W	16-bit
0x0C	EXPECTED_MODE (CFG1)	R/W	Mode mask/expectation

Preamble Correlator (RX) (BASE: 0xA000_D000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [7] START (arm detect)
0x04	STATUS	R/W1C	[0] PEAK_FOUND, [1] STROBE_ACTIVE
0x08	PREAMBLE_LEN (CFG0)	R/W	Symbols (32–64)
0x0C	THRESHOLD (CFG1)	R/W	Compare value
0x10	WIN_LEN (CFG2)	R/W	Search window
0x18	PEAK_IDX (RESULT0)	RO	Peak index
0x1C	PEAK_MAG2 (RESULT1)	RO	Peak magnitude^2

Differential Decoder (RX) (BASE: 0xA000_E000)

Offset	Name	R/W	Description

0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET (prev ← (1,0)), [6:4] MODE (0=DBPSK,1=DQPSK)
0x04	STATUS	R/W1C	[0] RUNNING, [4] ERROR (invalid phase bin, diag)

Slicer (BASE: 0xA000_F000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [6:4] MODE (0=BPSK,1=QPSK)
0x04	STATUS	R/W1C	[0] RUNNING
0x18	BIT_PACKED_COUNT (RESULT0)	RO	Output bytes produced

Impairment Injector (BASE: 0xA001_0000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [6:4] MODE (0=OFF,1=RANDOM,2=INTERVAL,3=BURST)
0x04	STATUS	R/W1C	[0] RUNNING
0x08	ERR_PROB_ppm (CFG0)	R/W	0–1e6 ppm
0x0C	INTERVAL_N (CFG1)	R/W	Flip every N bits
0x10	BURST_LEN (CFG2)	R/W	Contiguous flips
0x14	SEED (CFG3)	R/W	RNG seed
0x18	APPLIED_COUNT (RESULT0)	RO	Total flips applied

Viterbi (IP Wrapper) (BASE: 0xA001_1000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [1] BYPASS, [2] SW_RESET, [6:4] TRACEBACK_SEL
0x04	STATUS	R/W1C	[0] RUNNING, [2] OVERFLOW (should be 0)

BER Counters (BASE: 0xA001_2000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [6:4] MODE (0=PRBS-ref,1=CRC-gated), [15] CLEAR
0x04	STATUS	R/W1C	[0] ALIGNED
0x08	PRBS_POLY (CFG0)	R/W	Match TX
0x0C	WINDOW_LEN_BITS (CFG1)	R/W	Measurement window
0x18	BIT_COUNT (RESULT0)	RO	Bits compared
0x1C	ERR_COUNT (RESULT1)	RO	Errors counted
0x20	BER_Q16_16 (RESULT2)	RO	Fixed-point BER

SNR / EVM Estimator (BASE: 0xA001_3000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [6:4] MODE (0=BPSK,1=QPSK), [15] CLEAR
0x04	STATUS	R/W1C	[0] VALID (latched), [1] OVERFLOW
0x08	WIN_LEN_SYMS (CFG0)	R/W	Symbols per frame/window
0x18	EVM_Q16_16 (RESULT0)	RO	EVM% fixed-point
0x1C	SNR_Q8_8 (RESULT1)	RO	SNR dB fixed-point
0x20	SYM_COUNT (RESULT2)	RO	Symbols accumulated

AMC FSM (BASE: 0xA001_4000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE, [2] SW_RESET, [8] OVERRIDE_EN, [6:4] OVERRIDE_MODE
0x04	STATUS	R/W1C	CURRENT_MODE (bits[2:0])
0x08	THRESH_UP (CFG0)	R/W	SNR/EVM up thresholds
0x0C	THRESH_DOWN (CFG1)	R/W	Down thresholds
0x10	DWELL_FRAMES (CFG2)	R/W	Min frames per state
0x18	TIME_IN_STATE (RESULT0)	RO	Frames spent in current
0x1C	SWITCH_COUNT (RESULT1)	RO	Total transitions

Constellation Snapshot (BASE: 0xA001_5000)

Offset	Name	R/W	Description
0x00	CTRL	R/W	[0] ENABLE/ARM, [2] SW_RESET, [7] START (one-shot), [15] CLEAR
0x04	STATUS	R/W1C	[0] BUSY, [1] DONE, [2] DROPPED
0x08	DST_ADDR_L (CFG0)	R/W	DDR phys addr (low)
0x0C	DST_ADDR_H (CFG1)	R/W	DDR phys addr (high)
0x10	CAP_LEN_SYMS (CFG2)	R/W	Complex samples to capture
0x14	DECIM (CFG3)	R/W	Decimation factor
0x18	FRAMES_CAPT (RESULT0)	RO	Successful captures

10.2 Proposed Packet Format

Overview:
Each packet encapsulates one SDR frame (preamble + payload). The header provides synchronization, mode, and length; an optional CRC protects integrity. The payload is complex baseband symbols (`{I[15:0], Q[15:0]}`, Q1.15).

Field	Bits	Description
Sync Word	16	Fixed pattern (default 0xA5A5) for header alignment

Header Length	8	Length of header in bytes (excluding sync + CRC)
Mode	4	Modulation/FEC mode: 0=BPSK+FEC, 1=QPSK+FEC, 2=QPSK uncoded
Flags	4	[0]=CRC_EN, [1]=Preamble present, [2]=Reserved, [3]=Reserved
Sequence Number	16	Incrementing per frame; wraps at 2^16
Payload Length	16	Number of {I,Q} symbols in payload (excluding preamble)
Reserved	16	Reserved for future use (e.g., pilot index)
CRC-16-CCITT (optional)	16	Covers header + payload when CRC_EN=1

Packet Format:

[Header (96 b)] | [Preamble (M × 32 b, fixed ROM seq)] | [Payload (N × 32 b)] | [CRC (16 b, only if CRC_EN=1)]

Header (96 bits)	**Preamble (Mx32 bits)**	**Payload (Nx32 bits)**	**CRC (16 bits)**

Header: (96 bits)

Field	**Sync**	**Hlen**	**Mode/Flags**	**Seq**	**PayloadLen**	**Reserved**
Bits	16	8	8	16	16	32

Payload Format: (32 × N bits)

Pair #	**0**		**1**		**…**	**N**	
32-Bit IQ Pair	**I0**	**Q0**	**I1**	**Q1**	**…**	**IN**	**QN**
Bits	16	16	16	16	…	16	16

Total header size: 12 bytes (96 bits). If CRC is enabled, 2 additional bytes (16 bits) are appended at the packet end.

Payload Section:

- **Preamble:** Fixed sequence of 32–64 complex symbols, inserted at the start of every frame by the Preamble Inserter. Each symbol is {I[15:0], Q[15:0]} in Q1.15.

- **Data symbols:** Variable length, as specified in the header. Each symbol is {I[15:0], Q[15:0]} in Q1.15.

Trailer Section:

- **CRC (optional):** 16-bit CRC-16-CCITT appended at the end of the payload when CRC_EN=1.

Alignment:

- Entire packet is transmitted over AXI-Stream with tvalid/tready.
- tlast asserted at end of CRC (if CRC enabled) or at end of payload (if CRC disabled).
- Packet order is always: **Header → Preamble → Data → (CRC if enabled)**.

10.3 Fixed-Point Scaling Notes

Fixed-Point Format Primer:
Throughout this design, numeric signals are expressed in **Q-format notation**. A label such as **Qm.n** means the number has **m integer bits** (not including the sign) and **n fractional bits**.

- **Q1.15** = signed 16-bit number, 1 integer bit + 15 fractional bits.
  • Range ≈ −1.0 to +0.99997
  • Resolution ≈ 3×10^{-5}
  • Example: $0.75 \times 2^{15} = 24{,}576$ → stored as `0x6000`
- **Q2.30** = signed 32-bit number, 2 integer bits + 30 fractional bits.
  • Range ≈ −4.0 to +3.99999
  • Resolution ≈ 9×10^{-10}
  • Example: $0.75 \times 2^{30} \approx 805{,}306{,}368$ → stored as `0x30000000`

This convention makes scaling explicit and allows deterministic hardware/software comparison.

Constellation Symbols:

- `{I,Q}` represented in **Q1.15** signed fixed-point (16-bit).
- Symbol amplitudes ±1.0 map to ±32767.

PRBS / Scrambler / Descrambler:

- Operate on raw bits (AXI-Stream bytes). No fixed-point math.

Mapper:

- BPSK: {+32767, –32767} on I, Q=0.
- QPSK: {±23170, ±23170} (scaled √½ so unit power).

Differential Encoder/Decoder:

- Complex multiply in Q1.15.
- Saturating arithmetic with rounding.
- Reset state initialized to (32767, 0).

RRC Filters (TX/RX):

- FIR Compiler IP with 16-bit inputs/outputs.
- Coefficients pre-scaled to keep peak magnitude < 1.0.
- Internal accumulation width ≥32 bits; output truncated/rounded to Q1.15.

Preamble Inserter / Correlator:

- Preamble stored in Q1.15 ROM.
- Correlator accumulators ≥32 bits to avoid overflow on 64-symbol windows.

SNR/EVM Estimator:

- Error power accumulation in Q2.30 or wider.
- Ratios reported in fixed-point: EVM as Q16.16 (%), SNR as Q8.8 (dB).

BER Counters:

- 32-bit bit_count and err_count.
- Optional Q16.16 BER register for host convenience.

10.4 Simulation Verification Plan

Unit-Level Tests:

- **PRBS Generator:** Match against MATLAB/NumPy golden sequences (PRBS7/15/23/31). Verify TLAST intervals.
- **Scrambler/Descrambler:** Round-trip = original PRBS. Toggle bypass mid-frame.
- **Mapper + Slicer:** PRBS→Mapper→Slicer recovers original bits in noiseless sim. Check BPSK/QPSK Gray mapping.
- **Differential Encoder/Decoder:** Golden compare vs MATLAB DBPSK/DQPSK. Histograms show allowed Δφ only.
- **FEC Encoder/Decoder:** Encode PRBS → inject random errors → decode; match MATLAB vitdec results.
- **Preamble Inserter/Correlator:** Correlator peak aligns to known offset; single detection per frame.
- **RRC Filters:** Impulse/frequency response matches MATLAB within quantization error.

System-Level Tests (Sim):

- **End-to-End Loopback (No Errors):** PRBS → full TX chain → RX chain → recovered PRBS. Expect 0 mismatches.
- **Impairment Injection:** Inject controlled BER; verify counters measure expected BER within binomial tolerance.
- **CRC / Packetizer/Depacketizer:** Bad header/CRC injected → depacketizer flags error, respects drop/forward policy.
- **Back-Pressure:** Randomize tready in testbench; confirm no symbol duplication/loss; TLAST alignment preserved.
- **Metrics:** AWGN channel model at known Es/N0; measured SNR/EVM within ±0.5 dB vs MATLAB reference.
- **AMC FSM:** Sweep SNR across thresholds; confirm hysteresis and dwell timers behave as configured; FSM transitions logged once per event.

10.5 Petalinux Build & Setup Notes

TBD (TODO)

10.6 Acronyms

Acronym	Meaning
AMC	Adaptive Modulation and Coding
API	Application Programming Interface
AXI	Advanced eXtensible Interface
AXIS	AXI-Stream (AXI streaming protocol)
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
CDC	Clock Domain Crossing
CLI	Command-Line Interface
CRC	Cyclic Redundancy Check
DDR	Double Data Rate (memory)
DMA	Direct Memory Access
DQPSK	Differential Quadrature Phase-Shift Keying
EVM	Error Vector Magnitude
FEC	Forward Error Correction
FIR	Finite Impulse Response
FSM	Finite State Machine
GTY	GigaTransceiver (high-speed serial transceiver in UltraScale+)
ILA	Integrated Logic Analyzer
IP	Intellectual Property (core)
LUT	Lookup Table

LFSR	Linear Feedback Shift Register
MPSoC	Multiprocessor System-on-Chip
PL	Programmable Logic
PS	Processing System
PRBS	Pseudo-Random Binary Sequence
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
ROM	Read-Only Memory
RRC	Root Raised Cosine (filter)
RTL	Register-Transfer Level
SCP	Secure Copy Protocol
SDR	Software-Defined Radio
SFP+	Small Form-Factor Pluggable, 10 Gb/s
SNR	Signal-to-Noise Ratio
SW_RESET	Software Reset
TBD	To Be Determined
TLAST	AXI-Stream signal marking end of frame
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UIO	Userspace I/O (Linux device access)
VHDL	VHSIC Hardware Description Language
VIO	Virtual Input/Output (debug IP)