FPGA BASED STOCK MARKET DATA FEED HANDLER

By

Sara Sehgal

Saksham Jairath

Neel Ghoshal

Project Proposal for ECE 445, Senior Design, Spring 2026

TA: Gerasimos Gerogiannis

13 Feb 2026

Project No. 76

Contents

1. Introduction

Modern electronic trading systems require deterministic, low-latency processing of high-rate market data streams, which general-purpose software systems cannot reliably guarantee under heavy load. This proposal outlines the design of a hardware-based market data feed handler using a custom Ethernet front-end PCB and FPGA logic to achieve predictable packet processing and ingress timestamping. The report will present the planned system architecture, PCB and FPGA design approach, and the methodology that will be used to evaluate functionality, throughput, and latency performance.

1.1 Problem

Electronic financial markets generate extremely high-rate streams of market data that must be processed with very low and predictable latency. Exchanges continuously broadcast order book updates, trades, and price changes to participants, and trading firms rely on this information to make time-sensitive decisions. As markets have become increasingly electronic and competitive, even microsecond-scale delays can materially affect execution quality and risk exposure. Software-based feed handlers running on general-purpose processors often experience nondeterministic delays due to caching, interrupts, and operating system scheduling, making it difficult to guarantee bounded latency under heavy load. This has motivated the use of hardware-accelerated and FPGA-based systems in industry to achieve deterministic, low-latency performance.

The importance of reliable and timely market data processing extends beyond individual trading firms. Modern financial markets support capital allocation, price discovery, and liquidity provision across global economies. If market data is delayed, corrupted, or inconsistently processed, it can contribute to pricing inefficiencies, increased volatility, or systemic instability. Events such as flash crashes have highlighted how tightly coupled, automated systems can amplify errors or latency imbalances, raising concerns about market fairness and resilience. Ensuring predictable and verifiable processing of market data is therefore linked to broader issues of financial stability, economic welfare, and trust in electronic markets.

From a societal and global perspective, financial markets underpin retirement systems, corporate financing, government debt markets, and international trade. The infrastructure that processes market data must therefore meet high standards of reliability and determinism to protect public welfare and economic stability. Hardware-based, deterministic processing architectures reduce the risk of data loss and unpredictable behavior during peak trading activity, contributing to safer and more robust financial systems. As global markets operate continuously across regions and time zones, improving the reliability and transparency of electronic trading infrastructure has economic and societal implications that extend well beyond individual firms.

1.2 Solution

The proposed solution is a fully hardware-based market data feed handler implemented on an FPGA, coupled with a custom-designed Ethernet front-end PCB. The system ingests live market data over a physical Ethernet connection, processes packetized trading updates in real time, maintains per-symbol top-of-book state, and generates low-latency trigger events when predefined trading conditions are met. By implementing the complete data path in hardware—from physical network ingress to trigger generation—the design eliminates operating system and software-induced jitter, enabling deterministic and bounded processing latency suitable for latency-sensitive trading applications.

At a high level, the custom PCB provides the physical-layer network interface, including an RJ45 connector with magnetics, Ethernet PHY, clocking, power conditioning, and protection circuitry. The PHY communicates with the FPGA over an RMII interface through the Pmod+ expansion connector. Inside the FPGA, a hardware

Ethernet MAC receives frames and immediately captures an ingress timestamp at the physical boundary. The packet payload is then passed through FIFO buffers into a finite-state machine that parses market data messages and extracts relevant fields. A trading state manager implemented using on-chip memory updates best bid and best ask values per symbol, and a trigger engine evaluates predefined conditions to generate output signals and diagnostic metrics. This architecture creates a deterministic, end-to-end hardware pipeline that closely mirrors real-world FPGA-based trading systems while remaining suitable for controlled academic implementation and evaluation.

1.3 Visual Aid

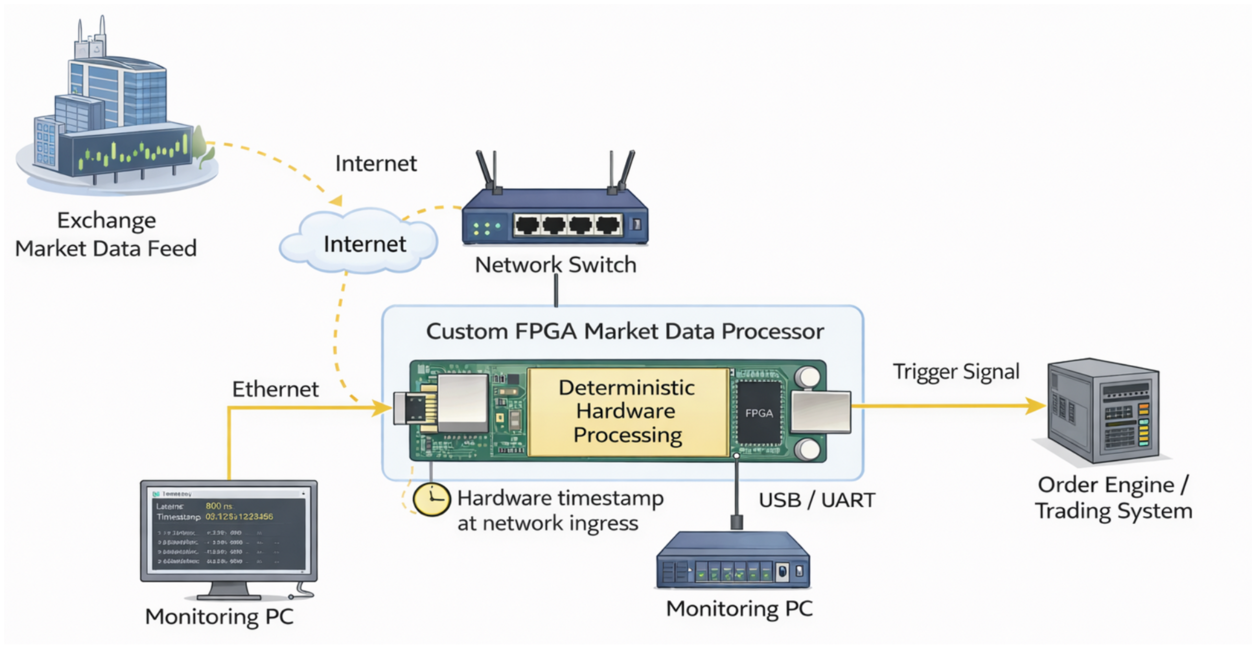
Exchange
Market Data Feed
Internet
Internet
Network Switch
Custom FPGA Market Data Processor
Deterministic Hardware Processing
FPGA
Ethernet
Trigger Signal
Order Engine / Trading System
Hardware timestamp at network ingress
USB / UART
Latency 800 ns
Monitoring PC
Monitoring PC

Figure 1: Contextual diagram showing the FPGA-based market data processor deployed between exchange market data infrastructure and a trading engine, with hardware-level deterministic processing and timestamped trigger generation.

1.4 High Level Requirements List

- The system shall sustain continuous reception and processing of Ethernet market data at **100 Mbps line rate** without packet loss under back-to-back minimum-sized frame conditions.
- The end-to-end hardware latency from detection of packet ingress at the Ethernet interface (rising edge of CRS_DV) to trigger signal assertion shall be **bounded and deterministic, not exceeding 1 microsecond** under nominal operating conditions.
- The ingress timestamping mechanism shall provide a time resolution of **10 nanoseconds or better** (corresponding to a 100 MHz clock or higher) and shall correctly capture timestamps for 100% of valid received frames.

2 Design

2.1 Block Diagram

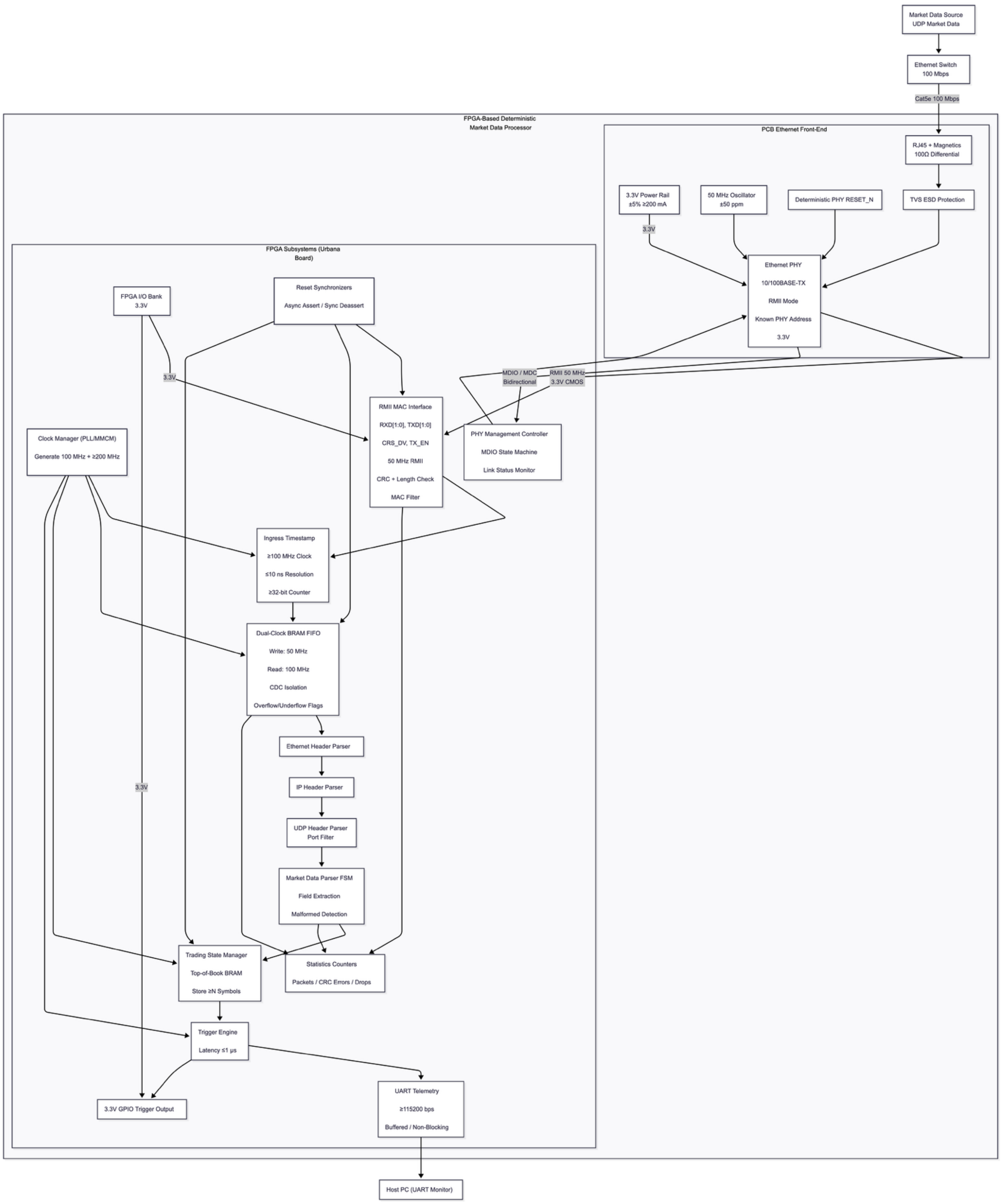

Figure 2: System level block diagram with detailed subsystems and requirements

2.2 Subsystem Overview

2.2.1 PCB Ethernet Front-End Subsystem

This subsystem provides the physical network entry point for market data. It accepts a standard Ethernet cable through an RJ45 connector (with magnetics), protects the interface against electrostatic discharge (ESD), and uses a 10/100 Ethernet PHY to convert the analog cable signaling into digital RMII signals. It connects to the FPGA subsystem through the Pmod+ connector using RMII data/control signals plus MDIO/MDC for configuration and a 50 MHz reference clock for synchronous operation.

2.2.2 PCB Power, Clock, and Debug Subsystem

This subsystem makes the PCB independently testable and ensures stable operation of the PHY. It distributes 3.3 V power (from the Pmod+ 3.3 V rail or an equivalent bench supply), filters and decouples the supply, provides a 50 MHz clock source for the PHY, and exposes link/activity LEDs and test points. It connects to the Ethernet PHY (power, clock, reset, LEDs) and indirectly supports the FPGA subsystem by ensuring the PHY is stable and observable during bring-up.

2.2.3 FPGA Ethernet MAC + RMII Interface Subsystem

This subsystem is the FPGA's hardware network interface to the PCB PHY. It receives RMII signals from the PHY, reconstructs Ethernet frames, and optionally transmits frames back through RMII. It provides a clean byte/packet stream to downstream logic and exposes MDIO/MDC transactions for PHY configuration and status monitoring. It connects upstream to the PCB via RMII/MDIO and downstream to the buffering subsystem via a streaming interface.

2.2.4 FPGA Ingress Timestamp Subsystem

This subsystem timestamps market-data arrival at the ingress boundary to measure deterministic latency. A free-running counter clocked by a stable FPGA clock latches a timestamp on the start-of-frame event (e.g., rising edge of CRS_DV or equivalent frame-start indication) before buffering and parsing. It connects to the MAC/RMII subsystem for the frame-start signal and to the FIFO/buffer subsystem by attaching a timestamp metadata word to each received frame.

2.2.5 FPGA FIFO / Buffer Subsystem

This subsystem absorbs bursty arrivals and prevents packet loss by buffering incoming frame bytes and metadata in BRAM-based FIFOs. It also serves as a clock-domain crossing boundary if the RMII side and processing side use different clocks. It connects upstream to the MAC/timestamp subsystems and downstream to the packet parser subsystem, ensuring continuous throughput and protecting the design's lossless requirement.

2.2.6 FPGA Packet Parser Subsystem

This subsystem converts raw packet payload bytes into structured market-data messages. Implemented as a finite state machine, it detects frame boundaries, validates basic formatting, and extracts fields (e.g., symbol ID, side, price, size, sequence number). It connects upstream to the FIFO/buffer subsystem and downstream to the trading state and trigger subsystems by outputting parsed message fields with a valid/ready handshake.

2.2.7 FPGA Trading State Manager Subsystem

This subsystem maintains per-symbol top-of-book state (best bid and best ask). It updates BRAM-resident state using parsed messages and provides the current top-of-book to the trigger engine. It connects upstream to the packet parser and downstream to the trigger engine, forming the core "market state" needed for trading-condition evaluation.

2.2.8 FPGA Trigger + Telemetry Output Subsystem

This subsystem evaluates predefined conditions on updated market state and produces low-latency trigger outputs. It also exports telemetry (timestamps, latency measurements, error counters) over UART and optionally asserts a GPIO trigger pin. It connects upstream to the trading state manager (state values) and timestamp subsystem (ingress timestamps) and provides outputs to external measurement/monitoring equipment.

2.3 Subsystem Requirements

2.3.1 PCB Ethernet Front-End Subsystem — Requirements

Contribution to high-level requirements: This subsystem is required to sustain **100 Mbps line-rate market data ingestion** and provide a stable, deterministic physical ingress boundary for timestamping and trigger-latency measurement. If it fails, the system cannot meet throughput or determinism goals.

Interfaces (quantitative):

- **Ethernet:** 10/100BASE-TX over RJ45.
- **RMII to FPGA (3.3 V CMOS):** RXD[1:0], CRS_DV, TXD[1:0], TX_EN, REF_CLK=50 MHz.
- **Management:** MDIO (bidirectional), MDC (output), PHY RESET_N.
- **Electrical levels:** 3.3 V logic, compatible with FPGA I/O standards.
- **Clock:** 50 MHz reference must meet RMII duty-cycle and stability constraints (typical ±50 ppm oscillator is sufficient for 10/100).

Minimum subsystem requirements:

- The subsystem shall support **100 Mbps** 10/100BASE-TX link operation (auto-negotiation or forced 100 Mbps).
- The subsystem shall provide RMII signals synchronized to a **50 MHz** reference clock.
- The subsystem shall include magnetics (integrated or discrete) suitable for 10/100BASE-TX.
- The subsystem shall include ESD protection on the RJ45 line side.
- The PHY configuration straps shall set the PHY into **RMII mode** and a known PHY address.
- The PHY shall expose link/activity indication (either dedicated LED pins or status register via MDIO).

2.3.2 PCB Power, Clock, and Debug Subsystem — Requirements

Contribution to high-level requirements: Stable power and clocking are necessary to meet **line-rate throughput** and to avoid jitter or link drops that would violate deterministic operation. Debug visibility is required to validate "fully functional" operation independently.

Interfaces (quantitative):

- **Power input:** 3.3 V from Pmod+ (or bench), distributed to PHY and support parts.
- **Clock output:** 50 MHz clock to PHY `REF_CLK`.
- **Debug:** LINK/ACT LEDs, test points for 3.3 V, GND, CLK, RESET, MDIO.

Minimum subsystem requirements:

- The subsystem shall supply **3.3 V ± 5%** to the PHY continuously.
- The subsystem shall support at least **200 mA continuous current** available to PHY + LEDs + oscillator (conservative sizing).

- The subsystem shall provide a **50 MHz** reference clock with stable amplitude compatible with PHY input.
- The subsystem shall provide a deterministic reset behavior: PHY `RESET_N` held low on power-up and released after power stabilizes (RC or supervisor).
- The subsystem shall provide at least **one link-status indicator** (LED or MDIO-readable register) to confirm independent operation.

2.3.3 FPGA Ethernet MAC + RMII Interface Subsystem — Requirements

Contribution to high-level requirements: This subsystem must reconstruct Ethernet frames at **100 Mbps** and deliver payload bytes without loss to meet the throughput requirement and provide the reference event used by timestamping.

Interfaces (quantitative):

- **RMII input:** `RXD[1:0]`, `CRS_DV`, `REF_CLK=50 MHz`.
- **RMII output:** `TXD[1:0]`, `TX_EN` (optional for testing; required for full-duplex capability).
- **MDIO/MDC:** ability to read PHY ID/status and configure link settings.
- **Downstream stream:** byte stream + frame boundary marker; recommended valid/ready handshake (or FIFO write interface).

Minimum subsystem requirements:

- The subsystem shall receive Ethernet frames at **100 Mbps** and pass payload bytes to downstream logic with no internal drops.
- The subsystem shall detect start-of-frame and end-of-frame conditions (frame boundary markers).
- The subsystem shall support PHY management via MDIO: read PHY ID and link status at minimum.
- The subsystem shall operate correctly with a **50 MHz RMII clock** and meet FPGA timing closure on that interface.

2.3.4 FPGA Ingress Timestamp Subsystem — Requirements

Contribution to high-level requirements: Provides the measurement mechanism to verify deterministic latency. It directly supports the requirement of **≤10 ns timestamp resolution** and enables bounded latency verification.

Interfaces (quantitative):

- **Input:** frame-start indication (e.g., `CRS_DV` rising edge or MAC start-of-frame pulse).
- **Clock:** uses FPGA clock (recommended **100 MHz** or higher).
- **Output:** timestamp metadata attached to each received frame or first byte.

Minimum subsystem requirements:

- The subsystem shall maintain a free-running counter clocked at **≥100 MHz** (10 ns or better resolution).
- The subsystem shall latch a timestamp on every received frame's ingress event with **100% capture** for valid frames.
- The timestamp value width shall be at least **32 bits** to avoid rapid wraparound during tests.
- The timestamp shall be paired deterministically with the correct frame (no reordering).

2.3.5 FPGA FIFO / Buffer Subsystem — Requirements

Contribution to high-level requirements: This subsystem is the primary mechanism preventing loss during bursts and is required to sustain **100 Mbps** without packet drops while preserving determinism.

Interfaces (quantitative):

- **Upstream write:** byte stream + valid, optional `last`, plus timestamp metadata at frame start.
- **Downstream read:** same byte stream + frame markers.
- **Clocking:** may implement clock-domain crossing from RMII clock domain to processing clock domain.

Minimum subsystem requirements:

- The subsystem shall buffer enough data to absorb back-to-back frames at **100 Mbps** without overflow for the expected burst length.
- If clock domains differ, the subsystem shall perform safe clock-domain crossing (asynchronous FIFO or equivalent).
- FIFO shall preserve ordering and frame boundaries exactly.
- FIFO shall expose overflow/underflow indicators for debugging.

2.3.6 FPGA Packet Parser Subsystem — Requirements

Contribution to high-level requirements: Correct parsing is required to match the reference model and to ensure triggers are computed from correct state, supporting both correctness and deterministic trigger timing.

Interfaces (quantitative):

- **Input:** byte stream + start/end-of-frame markers, plus ingress timestamp for the frame.
- **Output:** parsed fields (symbol, side, price, size, sequence), plus valid/ready handshake.

Minimum subsystem requirements:

- The subsystem shall parse all valid packets according to the defined message format with **100% correctness** under test vectors.
- The subsystem shall detect malformed packets and assert an error flag/counter.
- The subsystem shall maintain alignment between frame boundaries and parsed messages.
- The subsystem shall output parsed messages at a rate sufficient to keep up with **100 Mbps** input traffic (no backpressure-induced drops).

2.3.7 FPGA Trading State Manager Subsystem — Requirements

Contribution to high-level requirements: Maintains the real-time top-of-book state needed for triggers. If state updates lag or corrupt, correctness and trigger timing become invalid.

Interfaces (quantitative):

- **Input:** parsed message fields + valid/ready.
- **Memory:** BRAM-based storage indexed by symbol ID.
- **Output:** updated best bid/ask and optional state-read interface for trigger evaluation.

Minimum subsystem requirements:

- The subsystem shall update top-of-book for each message within a fixed bounded number of cycles (deterministic update latency).
- The subsystem shall store at least **N symbols** (choose N explicitly in your doc, e.g., 256 or 1024) with bid/ask price and size fields.
- The subsystem shall resolve updates deterministically (e.g., ignore out-of-sequence or apply last-write policy based on sequence number).
- The subsystem shall expose state to trigger engine in the same clock domain to avoid additional CDC complexity.

2.3.8 FPGA Trigger + Telemetry Output Subsystem — Requirements

Contribution to high-level requirements: Generates the final output within the project's latency bound and provides measurable evidence (timestamps/latency) that the system meets determinism targets.

Interfaces (quantitative):

- **Inputs:** top-of-book state updates, ingress timestamp, optional sequence/error flags.
- **Outputs:**
    - GPIO trigger (3.3 V) with deterministic assertion timing.
    - UART telemetry (e.g., **115200 bps** or faster if supported).

Minimum subsystem requirements:

- The subsystem shall assert a trigger within the project latency bound (e.g., **≤1 µs from ingress event**, as defined in your high-level requirements).
- The subsystem shall output telemetry including at least: ingress timestamp, trigger time (or delta), and error counters.
- UART output shall not stall the real-time pipeline (telemetry must be buffered or rate-limited).
- The subsystem shall provide at least one external observable trigger signal (GPIO or LED) for measurement.

2.4 Tolerance Analysis

Risky aspect: The highest risk to successful completion is reliable **100 Mbps Ethernet reception** over the **RMII (Reduced Media Independent Interface)** connection between the Ethernet PHY on the custom PCB and the Urbana FPGA through the Pmod+ header. RMII runs a **50 MHz synchronous interface** with multiple single-ended signals (RXD[1:0], CRS_DV, TXD[1:0], TX_EN plus REF_CLK). If the **clock-to-data skew** and propagation delays are too large (from PCB routing, connector, and I/O buffer variation), the FPGA can sample incorrect bits, causing frame corruption and packet loss.

2.4.1 Feasibility via timing budget

RMII uses a **50 MHz clock**, so the clock period is:

- $T = \frac{1}{50\text{ MHz}} = 20\text{ ns}$

For correct sampling at the FPGA, the received data must meet:

- **Setup constraint:** data must be stable at least t_{setup} before the sampling clock edge
- **Hold constraint:** data must remain stable at least t_{hold} after the edge

<u>**Setup Time**</u>

A conservative worst-case setup budget can be written as:

$$t_{co(PHY)} + t_{pd(data)} + t_{skew} + t_{setup(FPGA)} < T$$

Where:

- $t_{co(PHY)}$: PHY clock-to-out delay (data launched relative to REF_CLK)
- $t_{pd(data)}$: PCB/connector propagation delay on data
- t_{skew}: mismatch between clock and data arrival (routing mismatch + buffer skew + jitter)
- $t_{setup(FPGA)}$: FPGA input setup requirement (including internal input path)

Plugging in conservative numbers

At 10/100 speeds, typical PHY and FPGA I/O delays are on the order of a few ns. Use conservative assumptions:

- $t_{co(PHY)} \approx 3$ ns (safe upper estimate for RMII output timing)

- PCB trace propagation: FR-4 signal speed ≈ **6 in/ns ⇒ 0.167 ns/in**
    - If traces are short (≤ 3 inches from PHY to Pmod+), then $t_{pd(data)} \leq 0.5$ ns
- Routing mismatch: if you length-match RMII lines within **1 inch**, skew from routing is:
    - $\Delta t_{route} \leq 0.167$ ns
- Add buffer/jitter margin:
    - $t_{skew} \approx 1.0$ ns (includes routing mismatch, connector variation, and clock jitter margin)
- FPGA setup requirement (conservative):
    - $t_{setup(FPGA)} \approx 2$ ns

$$3+0.5+1.0+2=6.5\text{ns}<20\text{ns}$$

Setup slack (margin):

$$20 - 6.5 = 13.5 \text{ ns}$$

This is a large margin at 50 MHz, indicating the interface is feasible if routing is kept short and reasonably matched.

<u>Hold Time</u>

Hold violations occur if data changes too soon after the clock edge at the receiver. A conservative hold condition is:

$$t_{co(PHY,min)} + t_{pd(data,min)} - t_{skew} > t_{hold(FPGA)}$$

Using safe values:

- $t_{co(PHY,min)} \approx 1$ ns
- $t_{pd(data,min)} \approx 0.2$ ns
- $t_{skew} \approx 1.0$ ns
- $t_{hold(FPGA)} \approx 0.5$ ns

$$1 + 0.2 - 1.0 = 0.2 \text{ ns} \not> 0.5 \text{ ns}$$

This shows **hold is the tighter risk** under worst-case skew assumptions, which is typical for synchronous interfaces.

Mitigation (design choices that fix hold risk)

To make hold robust in practice, the design will include at least one of the following (standard hardware practice):

1. **Clock/data co-routing and length matching**
    - Match RMII data lines to REF_CLK within **≤ 0.5 inch** (≤ 0.084 ns mismatch)
2. **Add small series resistors (22–33 Ω) near the PHY** on RMII lines
    - Reduces ringing/edge-rate issues that worsen effective sampling uncertainty
3. **FPGA input registering and timing constraints**
    - Register RMII inputs immediately on the REF_CLK domain at the I/O boundary and constrain timing accordingly (ensures tools optimize the input path)
4. **Keep the PHY close to the Pmod+ connector**
    - Minimize trace lengths so t_{pd} is small and predictable

With reasonable routing (short traces and matched lengths), t_{skew} drops substantially (e.g., from 1.0 ns to ~0.3–0.5 ns), and the hold inequality becomes satisfied with margin.

The RMII interface is feasible at **50 MHz** with significant setup margin. The main tolerance risk is **hold sensitivity to clock/data skew**, which is mitigated through controlled PCB routing (short, matched traces), optional series damping resistors, and I/O-boundary registering in the FPGA. This analysis supports that a Pmod+-connected RMII PHY front-end can reliably sustain **100 Mbps** operation, enabling the project's throughput and deterministic-latency requirements.

3. Ethics, Safety and Societal Impact

3.1 Ethical considerations during development (IEEE/ACM)

This project involves designing hardware and FPGA logic that processes financial market data with very low latency. The primary ethical obligation is to prioritize public welfare, avoid harm, and be honest about performance claims and limitations. The IEEE Code of Ethics emphasizes responsibility for the safety, health, and welfare of the public, honesty in stating claims, and avoiding deceptive acts. The ACM Code of Ethics similarly frames the "public good" as a central consideration and calls for avoiding harm, being honest and trustworthy, and respecting intellectual property. In practice, we will avoid ethical breaches by (1) basing PCB and FPGA design decisions on datasheets and legitimate reference designs rather than copying proprietary work; (2) maintaining clear design logs and test evidence so performance numbers (throughput/latency) are reproducible and not misleading; and (3) properly citing external sources and any borrowed nontrivial design ideas.

3.2 Ethical risks from misuse (intentional or accidental)

Low-latency trading technology can be misused to create unfair informational advantages, contribute to an "arms race," or support abusive trading behaviors (e.g., manipulative strategies that degrade market quality). While this project is an educational prototype and not a production trading platform, responsible engineering still requires anticipating misuse. In alignment with IEEE/ACM expectations to minimize harm and act with integrity , we will (1) scope the system to a research/educational feed-handling and measurement tool, not an automated order-entry system; (2) avoid providing guidance on manipulative trading tactics; and (3) frame evaluation around determinism, correctness, and safety/reliability rather than "beating other market participants."

3.3 Safety concerns and mitigations (lab, electrical, and process safety)

The PCB operates at low voltage (3.3 V), so shock hazard is minimal, but there are still relevant safety risks: burns from soldering irons, fire risk if tools are left on, inhalation of flux fumes, and damage/injury from improper tool use. University guidance for soldering emphasizes preventing burns, using proper stands, and safe handling practices. Campus/course senior-design safety materials also emphasize planning for emergencies, knowing exits/extinguishers, and safe lab conduct. We will mitigate these risks by using fume extraction/ventilation during soldering, wearing eye protection, keeping flammables away from hot tools, powering down equipment when unattended, and following course lab sign-in/training procedures where applicable.

3.4 Regulatory and standards considerations (electrical, EMC, networking)

Even as a prototype, the PCB is a digital device that can radiate electromagnetic interference. In the U.S., many digital devices fall under FCC Part 15 rules for radio-frequency devices (including unintentional radiators), which govern operation without an individual license and set conditions for compliance/authorization when marketed. While formal FCC compliance testing is typically out of scope for a class prototype (especially if not marketed), we will follow good EMC practices: short return paths, proper decoupling, controlled edge rates where needed (e.g., series resistors on fast GPIO), and ESD protection at the RJ45 interface. The Ethernet interface is designed to interoperate with standardized Ethernet behavior as defined by IEEE 802.3 (physical layer and MAC specifications).

3.5 Societal, economic, environmental, and global context

Economically, reliable low-latency market data processing supports the infrastructure of modern electronic markets, which influences liquidity, price discovery, and risk management globally. At the same time, highly optimized trading infrastructure can contribute to unequal access and market fragmentation if deployed irresponsibly. Societally, failures in automated market infrastructure can amplify volatility and erode trust, so emphasizing determinism, validation, and transparent measurement (ingress timestamps and reproducible latency testing) supports safer engineering practice. Environmentally, the main impacts are small-scale: PCB fabrication and eventual electronics waste. We will minimize waste by limiting PCB respins through careful review (schematic/ERC, DRC, peer design checks), selecting widely available components to avoid unnecessary reorders, and reusing hardware where possible.

Summary of how we will avoid ethical breaches

- Use datasheets/standards and cite sources; avoid copying proprietary designs; document design provenance.
- Report performance with reproducible methodology and disclose limitations; avoid exaggerated latency/throughput claims.
- Build and test safely following campus soldering/lab safety guidance; use ventilation and proper PPE.
- Keep the project focused on feed handling/measurement, not order execution or guidance on manipulative trading behavior (harm minimization).

References

[1] IEEE, “IEEE Code of Ethics,” IEEE, Piscataway, NJ, USA. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8. [Accessed: Mar. 8, 2025].

[2] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” ACM, New York, NY, USA. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Mar. 8, 2025].