1. Introduction

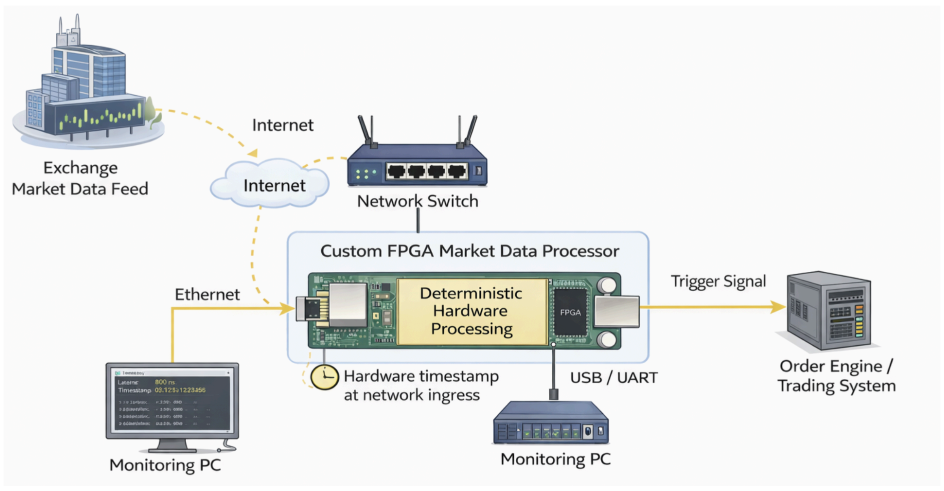
Problem and Solution

Modern electronic trading systems rely on the ability to process financial market data with extremely low latency. Exchanges broadcast continuous streams of market data containing updates such as trade prices, order book changes, and quote updates. Trading systems must parse and react to this information as quickly as possible in order to remain competitive. Traditional software-based processing on general-purpose CPUs introduces variable delays due to operating system scheduling, cache behavior, and network stack overhead. These delays can significantly impact the responsiveness of systems that rely on real-time financial data.

To address this challenge, many high-performance networking applications utilize hardware acceleration using field-programmable gate arrays (FPGAs). FPGAs allow packet processing logic to be implemented directly in hardware, enabling deterministic processing pipelines and significantly lower latency compared to software implementations. In this project, we propose the design of a low-latency FPGA-based market data feed handler capable of receiving network packets, parsing relevant trading information, maintaining simplified market state, and generating hardware trigger signals when predefined conditions occur.

The proposed system improves upon traditional CPU-based processing by implementing packet parsing and decision logic entirely within an FPGA pipeline. This architecture allows incoming data to be processed as a continuous byte stream with predictable timing. The system also integrates ingress timestamping to measure packet arrival time and evaluate processing latency. By demonstrating this architecture on a custom hardware platform, the project highlights how FPGA-based designs can be used to implement deterministic, high-performance data processing systems.

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
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.

High-Level Requirements

The following requirements define the essential characteristics that the system must achieve in order to successfully solve the problem described above.

- The system must receive and process Ethernet market data streams in real time using an FPGA-based hardware processing pipeline. Incoming packets must be accepted through the Ethernet interface and converted into a digital data stream that can be parsed by the FPGA logic.

- The system must correctly parse market data messages and extract key fields such as symbol identifier, price, and order size from the incoming data stream. These values must be captured and made available to the internal processing logic.

- The system must maintain an internal representation of market state for each processed symbol and update this state whenever new market data messages are received. This state information must be used to evaluate trigger conditions.

- The system must generate a hardware trigger signal when predefined conditions are satisfied by the processed market data. The trigger output must occur within a bounded and deterministic processing latency after the relevant data is received.

Failure to meet any of these requirements would prevent the system from performing its intended function as a low-latency hardware-based market data processing device.

2 Design

2.1 Block Diagram

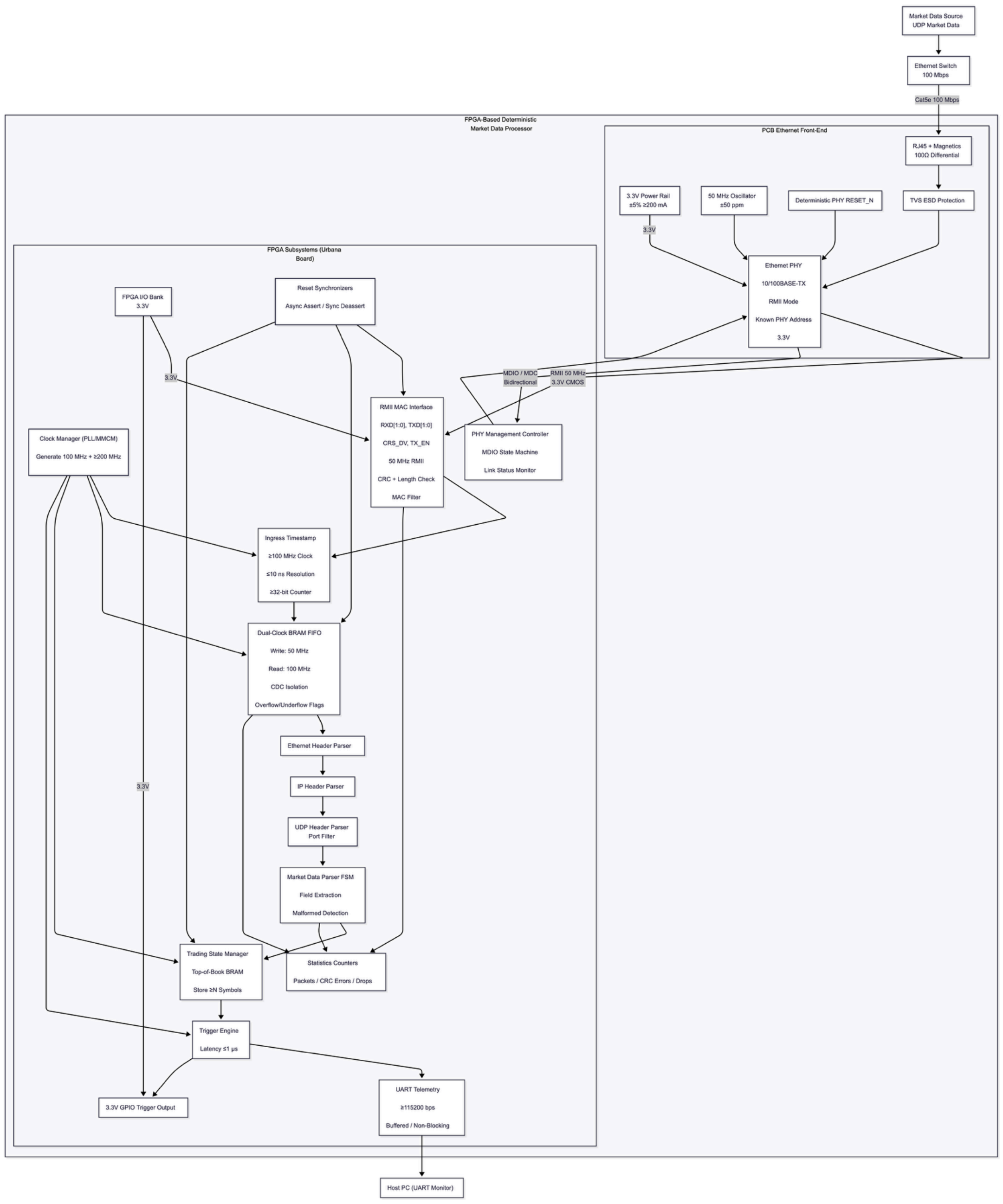


Figure 2: System level block diagram with detailed subsystems and requirements

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

Setup Time

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

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

- $t_{co(PHY)} \approx 3\text{ ns}$ (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 $t_{pd(data)} \leq 0.5\text{ ns}$
- Routing mismatch: if you length-match RMII lines within **1 inch**, skew from routing is:
  - $\Delta t_{route} \leq 0.167\text{ ns}$
- Add buffer/jitter margin:
  - $t_{skew} \approx 1.0\text{ ns}$ (includes routing mismatch, connector variation, and clock jitter margin)
- FPGA setup requirement (conservative):
  - $t_{setup(FPGA)} \approx 2\text{ ns}$

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

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.

Hold Time

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:

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

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

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. Cost and Schedule

3.1 Cost Analysis

The cost of the proposed system consists of both labor costs and material costs required to design and implement the final hardware prototype. The parts cost includes only the components necessary for the final integrated system. Hardware purchased purely for development purposes, such as temporary FPGA development boards used for testing during early implementation, is not included in the final product cost.

Labor Cost

Labor costs are estimated based on an assumed starting salary equivalent to $25 per hour for an electrical or computer engineering graduate. Following the recommended cost estimation guidelines for engineering projects, the hourly rate is multiplied by 2.5 to account for additional employment costs such as benefits, overhead, and administrative expenses.

Each team member is expected to contribute approximately 120 hours toward the project. These hours include time spent on system design, FPGA development, simulation, hardware design, debugging, testing, and documentation.

The labor cost per team member is calculated as:

25 ($/hour)×2.5×120 hours=7,500 USD25\ (\$/hour) \times 2.5 \times 120\ hours = 7,500\ USD25 ($/hour)×2.5×120 hours=7,500 USD

Since the project team consists of three members, the total labor cost is:

7,500×3=22,500 USD7,500 \times 3 = 22,500\ USD7,500×3=22,500 USD

Therefore, the total estimated labor cost for the project is $22,500.

Parts Cost

The following components are required to build the final hardware system. These components include the FPGA device, networking interface hardware, printed circuit board, and supporting electronic components required for normal operation.

The estimated parts cost includes:

- FPGA device (Artix-7 class) used to implement the packet processing pipeline and trading logic.
  Estimated cost: $45

- Ethernet PHY chip used to convert the Ethernet physical layer signals into digital signals that can be processed by the FPGA.

Estimated cost: $4

- RJ45 connector with integrated magnetics, which provides the physical Ethernet interface for network connectivity.
  Estimated cost: $3

- SPI flash memory used to store the FPGA configuration bitstream for system startup.
  Estimated cost: $1

- Voltage regulators required to generate the multiple voltage rails needed by the FPGA and supporting components (typically 1.0 V, 1.8 V, and 3.3 V).
  Estimated cost: $6

- Crystal oscillator (50 MHz reference clock) used for Ethernet timing and system clock generation.
  Estimated cost: $1

- Passive components including resistors, capacitors, and filtering components necessary for power stabilization and signal conditioning.
  Estimated cost: $5

- Custom PCB fabrication for the final hardware design. The board will likely be a four-layer PCB to support proper signal routing and power distribution.
  Estimated cost: $20

- Miscellaneous connectors and assembly materials required for interfacing and board integration.
  Estimated cost: $5

The total estimated parts cost for the final system is approximately $90.

Total Project Cost

Combining both labor and parts costs:

- Total labor cost: $22,500

- Total parts cost: $90

This results in an overall estimated project cost of approximately $22,590.

It should be noted that the labor cost reflects the engineering effort required to design and implement the system. In a commercial manufacturing scenario, this cost would be distributed across many units, resulting in a much lower per-unit cost.

3.2 Schedule

The project schedule is organized into several stages that correspond to the major phases of system development: architectural design, subsystem implementation, hardware design, system integration, and final validation.

The planned development schedule is outlined below:

- Week 1
  Finalize system architecture and subsystem definitions
  Establish project milestones and development plan

- Week 2
  Design Ethernet communication pipeline
  Define packet format and parsing logic

- Week 3
  Implement the frame parser for incoming data streams
  Begin simulation testing of packet parsing logic

- Week 4
  Implement the timestamping subsystem used for ingress latency measurement

- Week 5
  Implement the trading state manager and trigger generation logic

- Week 6
  Integrate the core FPGA subsystems and perform full simulation testing

- Week 7
  Begin PCB schematic design and component selection

- Week 8
  Complete PCB layout and prepare design for fabrication

- Week 9
  Assemble hardware and begin initial hardware testing

- Week 10
  Integrate FPGA logic with hardware platform

Perform debugging and functional verification

- Week 11
  Measure system latency and optimize performance

- Week 12
  Conduct final system validation and prepare final documentation and demonstration

Work will be distributed among the three team members to allow parallel progress on FPGA logic development, hardware design, and system testing. This parallel development approach helps reduce project risk and ensures sufficient time for debugging and performance validation before the final demonstration.

4. Discussion of Societal Impact, Engineering Standards, Ethics, and Safety Considerations

Societal Impact

The proposed system contributes to the development of high-performance computing infrastructure used in modern financial markets. Electronic trading systems rely on extremely low-latency data processing in order to react to rapidly changing market conditions. By implementing a deterministic hardware pipeline capable of processing market data streams in real time, the project demonstrates techniques used in high-performance network processing and hardware acceleration.

Beyond financial applications, the techniques used in this system—such as hardware-accelerated packet parsing, deterministic processing pipelines, and FPGA-based networking—are widely applicable in other fields including telecommunications infrastructure, network security appliances, and high-frequency data acquisition systems. These technologies improve the performance and efficiency of data processing systems that support modern digital infrastructure.

Engineering Standards

Several engineering standards are relevant to the design and implementation of this system.

First, the networking interface is designed to operate according to the **IEEE 802.3 Ethernet standard**, which defines the physical and data link layer protocols for Ethernet communication. Compliance with this standard ensures interoperability with existing network infrastructure and hardware devices.

Second, the electrical design of the hardware system must follow standard electronic design practices including appropriate signal integrity considerations, voltage regulation, and PCB layout techniques. These practices help ensure reliable operation of high-speed digital circuits and reduce the likelihood of electromagnetic interference.

Additionally, the design process follows recommended engineering documentation and development practices to ensure that the system architecture is modular, verifiable, and maintainable.

Ethical Considerations

Engineers have an ethical responsibility to design systems that operate reliably, transparently, and safely. In accordance with the IEEE and ACM codes of ethics, the development of this project emphasizes accuracy, reliability, and responsible engineering practices.

Although the system processes financial market data, the project itself does not execute trading decisions or interact directly with financial markets. Instead, it serves as a research and educational demonstration of hardware-accelerated networking techniques. Care is taken to ensure that the system processes data accurately and does not introduce unintended behavior that could lead to incorrect outputs.

The design process also emphasizes proper attribution of all external resources, including reference materials and technical documentation used during the development of the system.

Safety Considerations

The hardware system operates entirely within standard low-voltage digital electronics ranges, typically between **1.0 V and 3.3 V**, which significantly reduces electrical safety risks. Nonetheless, several safety considerations must be addressed during development and testing.

First, proper handling procedures must be followed to avoid electrostatic discharge (ESD), which can damage sensitive semiconductor components such as the FPGA and Ethernet PHY. Team members will use grounded workstations and appropriate handling techniques when assembling and testing hardware.

Second, the power supply circuits must be carefully designed to ensure that voltage regulators provide stable and correct voltage levels to all components. Incorrect voltage levels could damage integrated circuits or cause unreliable operation.

Finally, all testing and debugging procedures will follow the laboratory safety guidelines established for the course. This includes using appropriate lab equipment, verifying connections before applying power, and monitoring system behavior during testing to prevent component damage.

Through careful design practices and adherence to established safety guidelines, the project minimizes risks to both developers and equipment while ensuring safe operation of the hardware prototype.

References

[1] IEEE Standards Association, *IEEE Standard for Ethernet*, IEEE Std 802.3, 2018.

[2] A. Putnam et al., “A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services,” *Proceedings of the 41st Annual International Symposium on Computer Architecture (ISCA)*, 2014.

[3] J. W. Lockwood, N. Naufel, and J. Turner, “Reconfigurable Network Processing on Field Programmable Gate Arrays,” *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2001.

[4] AMD/Xilinx, *Artix-7 FPGA Data Sheet: DC and AC Switching Characteristics*, DS181, 2023.

[5] Microchip Technology Inc., *LAN8720A Ethernet Transceiver Datasheet*, 2022.

[6] OpenAI, “ChatGPT,” OpenAI, 2026. [Online].