Low-Latency FPGA Market Data Parser and Hardware Order Book

CSEE 4840 Project Proposal

Shawn Kathuria (shk2199), Shiyao Lam (sml2286), Sarah Hagan (sah2267), Siddharth Raykar (sr4102)

Spring 2026

1 Introduction

High-frequency trading firms use FPGAs to process market data and execute trades at nanosecond-scale latencies. The critical performance advantage comes from eliminating the CPU entirely from the data path: raw Ethernet frames carrying exchange data are received, parsed, and acted upon in FPGA fabric, bypassing the operating system, kernel network stack, and all associated software overhead. This project implements the core of such a system on the DE1-SoC.

We will build a hardware market data processing pipeline that receives simulated NASDAQ ITCH 5.0 protocol messages, parses them entirely in SystemVerilog, and maintains a sorted hardware order book in on-chip memory. The system will support Add Order, Order Executed, Order Cancel, Order Delete, and Order Replace message types. A VGA display will show the live state of the order book: best bid/ask, depth at each price level, and real-time statistics including messages processed and parse latency. Simultaneously, we will implement the same parsing and book-building logic in C on the HPS ARM core and rigorously benchmark the two paths to quantify the latency advantage of the hardware implementation.

A prior 4840 project (HFT Book Builder, Spring 2024) attempted a similar system but routed all data through the ARM processor via TCP sockets and the Linux kernel before forwarding it to the FPGA over the Avalon bus. This defeated the purpose of hardware acceleration. Our project eliminates the ARM from the critical data path entirely: a market data simulator transmits ITCH messages over a direct serial link to the FPGA fabric via GPIO, where they are received, framed, deserialized, parsed, and used to update the order book—all without CPU involvement.

2 System Overview

The system consists of three major components: a market data simulator running on an external host (laptop or second DE1-SoC), the FPGA-based receive and processing pipeline,

and software on the HPS for benchmarking and VGA control.

The external host replays recorded NASDAQ ITCH 5.0 data (freely available historical sample files from NASDAQ) by serializing each message into a simple framed byte stream and transmitting it over a serial link to a GPIO pin on the FPGA which delivers complete ITCH messages into the parser.

The parser is a hardware state machine that identifies the message type from the first byte and extracts the relevant fields (order reference number, side, shares, stock, price) into a fixed-width internal representation. This parsed message is then dispatched to the order book module.

The order book maintains per-stock, per-side (bid/ask) price-level tables in on-chip BRAM. Unlike the 2024 project's linked-list approach ($O(n)$ insertion and lookup), we use a direct-mapped price-level array where each entry stores aggregate quantity at that price. This gives $O(1)$ update for Add/Cancel/Execute and $O(n)$ scan for best-bid/best-ask in the worst case, though we maintain a running best-price register that is updated incrementally to avoid scanning in the common case.

3 Hardware Design

Serial Receiver. The data link from the simulator to the FPGA uses a simple asynchronous serial protocol at a configurable baud rate (targeting 3 Mbps, adjustable down for debugging).

Frame Delineation. ITCH messages are variable-length. The simulator prepends each message with a 2-byte big-endian length field, matching the real NASDAQ MoldUDP64 framing. A small state machine reads the length, counts incoming bytes, and signals message-complete to the parser.

ITCH Parser. A state machine reads the message type byte and branches to the appropriate field extraction logic. The parser extracts: message type (1 byte), stock locate (2 bytes), timestamp (6 bytes), order reference number (8 bytes), side (1 byte), shares (4 bytes), stock symbol (8 bytes), and price (4 bytes).

Order Book. We track up to 4 stocks simultaneously, each with a bid book and an ask book. Each book is a price-level table stored in a dedicated BRAM block. Assuming 16-bit price granularity (65,536 price levels per side), each entry stores a 32-bit aggregate quantity, giving 256 KB per book. With 4 stocks and 2 sides, total BRAM usage is approximately 2 MB—within the Cyclone V's 4 MB of embedded memory. For individual order tracking (needed for Cancel and Execute by order ID), we maintain a hash table in BRAM mapping order reference numbers to (stock, side, price, quantity) tuples. The hash table uses open addressing with linear probing, sized to hold 4,096 active orders.

VGA Display. A VGA controller displays the live order book state.

4 Software Design

Market Data Simulator (external host). A Python script reads a binary ITCH 5.0 sample file (available from NASDAQ's historical data FTP server), and transmits over a USB-to-serial adapter to the FPGA.

Software Reference Implementation (HPS ARM). A C program on the ARM core implements identical ITCH parsing and order book logic in software for analyzing performance of the FPGA to produce a latency distributiom.

Device Driver and Control Software. A Linux kernel module exposes the FPGA's status registers (message counts, latency counters, best bid/ask, error flags) to a program that can produce benchmark reports and can configure parameters (active stock filter, playback controls) via Avalon-mapped control registers.

The critical design principle is that the Avalon interface is used only for monitoring and benchmarking so market data flows from GPIO to the parser to the order book entirely within the FPGA fabric.

5 Milestones

- **Milestone 1 (25%):** Serial receiver and ITCH parser working in ModelSim simulation. Testbench sends a sequence of known ITCH messages; parser correctly extracts all fields for each supported message type. Framing logic validated.
- **Milestone 2 (50%):** End-to-end data path working on hardware. Market simulator on laptop transmits ITCH messages over serial to DE1-SoC GPIO. Parser receives and decodes messages correctly. Order book module accepts Add Order messages and maintains correct price-level state. Avalon registers readable from software.
- **Milestone 3 (75%):** Full order book functionality: Add, Cancel, Execute, Delete, and Replace all working correctly. VGA display showing live book state. Software reference implementation complete. Latency benchmarking infrastructure in place.
- **Final (100%):** Polished demo replaying a full day of NASDAQ ITCH data. VGA dashboard showing real-time book depth, spread, and message rate. Comparative latency analysis: per-message latency histogram for hardware path vs. software path. Written report with complete architecture documentation, register specifications, and performance results.

6 References

1. NASDAQ. NASDAQ TotalView-ITCH 5.0 Specification. 2020.
2. Litz, H. et al. High Frequency Trading Acceleration using FPGAs. IEEE Intl. Conf. on Field Programmable Logic and Applications, 2012.
3. Thenappan, C. et al. HFT Book Builder Implemented on DE1-SoC FPGA Board. CSEE 4840 Final Report, Columbia University, Spring 2024.
4. Chhabra, A. et al. Hardware Acceleration of Market Order Decoding. CSEE 4840 Final Report, Columbia University, Spring 2012.