Real-Time Market Behavior Classification with Deep Learning on FPGA Hardware

Xhovani Mali
Financial & Risk Engineering Capstone Project
New York University
Brooklyn, NY, USA
xxm202@nyu.edu

***Abstract*—This paper presents a prototype framework for analyzing financial market data in real-time using specialized neural networks (LSTM) implemented on customizable hardware (FPGAs). Our system aims to detect early warning signals of market behaviors—like price manipulation and sudden liquidity changes—that impact trading performance. We trained a specialized model on simulated market data that can classify short-term price movements with reasonable accuracy despite the challenge of rare significant events. When theoretically implemented on specialized hardware, our system could potentially process information in less than 3 millionths of a second—fast enough to compete with high-speed trading systems. These projected results suggest that machine learning models could be embedded directly into trading hardware to improve decision-making speed, potentially reducing losses from trading against faster market participants.**

***Index Terms*—High-Frequency Trading, Order Book Analysis, Long Short-Term Memory, Field-Programmable Gate Arrays, Hardware Acceleration**

I. Introduction

In modern financial markets, short-term price movements are often driven by subtle shifts in order flow. Detecting predictive patterns within the limit order book (LOB) ahead of competitors provides a critical trading edge, particularly in high-frequency trading (HFT), where execution occurs within microseconds and market opportunities exhibit a winner-takes-all dynamic [1].

Achieving this advantage demands ultra-low latency: systems must interpret data and execute decisions with minimal and deterministic delay. Conventional CPU- and GPU-based architectures, while flexible, suffer from unpredictable latencies due to operating system overhead, context switching, cache misses, and network jitter [2]. Even when optimized, software-based infrastructures struggle to consistently operate within microsecond response times, increasing exposure to adverse selection.

Field-Programmable Gate Arrays (FPGAs) offer a compelling alternative. Their reconfigurable pipelines support highly parallel, deterministic processing, enabling ultra-low latency applications. Historically deployed in HFT for market feed parsing, signal detection, and rapid order routing [3], FPGAs have achieved LSTM inference speeds exceeding 13 GOP/s [4] and latency reductions of over 250-fold compared to software implementations [3].

Building on this potential, we investigate embedding learned behavioral models—specifically, Long Short-Term Memory (LSTM) neural networks—directly into FPGA hardware. LSTM networks, a class of recurrent neural networks (RNNs), model sequential data with temporal dependencies and are well-suited for detecting evolving supply and demand imbalances across price levels. However, their computational intensity and recursive structure introduce latency challenges for real-time deployment.

This project addresses those challenges by designing a compact, quantized LSTM model optimized for FPGA implementation using high-level synthesis (HLS) workflows. Unlike traditional rule-based trading logic based on static indicators like moving averages or volume spikes, our model learns latent market behaviors directly from raw LOB sequences. Embedded directly into hardware, the model acts as a real-time behavioral signal detector—a "reflex" in silicon—flagging microstructure events such as spoofing, liquidity sweeps, and momentum ignition before they materially influence price.

The remainder of this paper is organized as follows: Section II presents the problem statement and context in finance, explaining the importance of low-latency decision-making in modern markets. Section III surveys related work in FPGA acceleration for financial applications and deep learning deployment. Section IV details our approach to feature engineering and model architecture. Section V covers methodology and implementation, explaining the FPGA compilation workflow. Section VI provides a comprehensive evaluation of model performance and hardware metrics. Section VII presents a summary of our results, and Section VIII concludes the paper and outlines directions for future work.

II. Problem Statement

The rapid evolution of financial markets has intensified the demand for high-speed, low-latency trading strategies. In high-frequency trading (HFT), microsecond-level delays can result in missed opportunities, adverse selection, and diminished profitability. This project addresses the challenge of identifying short-term market behaviors—such as spoofing, liquidity sweeps, and momentum ignition—early enough to improve execution quality. While time-series models like Long Short-Term Memory (LSTM) networks offer the potential to extract predictive signals from the limit order book (LOB), conventional CPU- and GPU-based systems often fail to meet the extreme latency requirements of modern trading environments.

Adversarial trading tactics such as spoofing, layering, and quote stuffing are specifically designed to exploit slower participants, manipulating perceived order book states to induce premature reactions. Capturing subtle shifts in market microstructure at microsecond latencies is essential to mitigate adverse selection and maintain competitiveness.

To address this, we investigate deploying quantized LSTM models directly on Field-Programmable Gate Arrays (FPGAs), aiming for sub-microsecond inference latencies. FPGA acceleration enables faster, more adaptive trading strategies that can react to evolving market conditions faster than traditional architectures.

A. *Context in Finance*

At the core of electronic markets is the limit order book (LOB), a real-time data structure aggregating buy and sell orders by price and time priority. The LOB drives price discovery, liquidity provision, and trade matching, continuously updating on millisecond to microsecond timescales [1].

Algorithmic trading strategies—from simple execution algorithms to complex predictive models—analyze the evolving LOB to make decisions. High-frequency trading firms, operating at extreme order rates, depend critically on detecting microstructure signals such as hidden liquidity shifts, momentum ignition, and spoofing.

Traditional rule-based trading systems, relying on static indicators like bid-ask spread or order flow imbalance, struggle to capture today's subtle, adversarial patterns. Deep learning models, particularly sequential architectures like LSTMs, have demonstrated the ability to learn temporal dependencies and latent structures in LOB data [5], [6], offering a path to more adaptive, real-time trading intelligence.

However, deploying LSTM models in live trading environments poses a major engineering challenge: inference must occur within tight latency budgets. CPU- and GPU-based systems often suffer from unpredictable delays due to operating system overhead, cache misses, and network jitter [2], [7]. FPGAs offer a compelling solution, enabling highly parallel, deterministic processing pipelines that meet HFT latency requirements [3], [8].

Recent frameworks like FINN [9], [10] and hls4ml [11] have further lowered barriers to deploying quantized deep learning models on FPGAs, aligning with broader industry trends toward AI-driven, hardware-accelerated trading infrastructures [12].

This research targets the financial problem of improving execution quality through real-time behavioral modeling embedded directly into hardware. Early detection of microstructure events enhances fill rates, reduces adverse selection, and improves market-making profitability—translating technological advantage into financial performance.

B. *Quantifying Adverse Selection Costs*

Adverse selection imposes significant measurable costs. When slower participants trade against faster, better-informed counterparties, they typically incur 1–3 basis points of slippage per trade, according to research by Aquilina, Budish, and O'Neill [13]. For a trading desk executing \$100 million daily, this translates into \$10,000–\$30,000 in daily losses—or \$2.5–\$7.5 million annually, as documented by Angel, Harris, and Spatt [14].

Budish et al. [15] estimate that latency arbitrage generates roughly \$1 billion annually in U.S. equities alone, reflecting zero-sum transfers from slower to faster participants. Industry benchmarks from Tabb Group [16] suggest that each microsecond of latency reduction in HFT is worth approximately \$100,000 per year in trading advantage, underscoring the enormous financial incentive for hardware-accelerated pattern recognition.

C. *Order Book Structure Overview*

In electronic financial markets, the limit order book (LOB) is a dynamic data structure that records all outstanding buy and sell orders for a specific asset, organized by price and time priority. It serves as the primary mechanism for price discovery, liquidity provision, and trade matching in modern electronic exchanges.

The LOB can be understood through three standard levels of detail:

- **Level 1 (L1):** Captures the best bid (highest available buy price) and best ask (lowest available sell price), along with their respective quantities. L1 provides a one-dimensional view of the most immediate liquidity available in the market.
- **Level 2 (L2):** Expands beyond the best prices to include multiple price levels on both the bid and ask sides, showing the aggregated quantities available at each level. L2 offers a deeper view into the supply and demand structure across the price spectrum.
- **Level 3 (L3):** Tracks individual orders at each price level, preserving their arrival sequence (FIFO—First In, First Out) or other exchange-specific priority mechanisms. L3 enables detailed modeling of queue dynamics, resting liquidity, and the order matching process at the participant level.

Accurately modeling the LOB structure provides the foundation for predictive modeling, but real-time deployment introduces additional challenges related to latency, determinism, and computational efficiency, which we discuss next.

D. *Challenges in Real-Time Order Book Analysis*

The primary challenge in real-time LOB analysis is the need to react within microseconds. Even slight delays can result in missed fills or adverse selection. Traditional CPU- and GPU-based systems suffer from processing variability due to operating system overhead, cache hierarchy, and network stack bottlenecks, making deterministic low-latency decision-making difficult.

While LSTM models offer a promising approach for capturing temporal dependencies in LOB dynamics, their recursive structure introduces data hazards that complicate pipelining on conventional architectures. FPGA implementations of LSTM

TABLE I
COMPARISON OF DIFFERENT HARDWARE IMPLEMENTATIONS FOR LSTM

Platform	**Benefits**	**Drawbacks**
FPGA	Low latency, highly parallel processing, reconfigurable hardware	Complex development cycle, resource constraints, higher initial cost
ASIC	High performance, lowest latency, power-efficient	Very high costs, no reconfigurability
CPU	Easy to program, widely available	Higher latency, limited parallelism for MATMUL
GPU	Good for parallel and matrix operations	High power, high latency

models have demonstrated order-of-magnitude improvements in speed and energy efficiency compared to CPU deployments [17], supporting their viability for real-time applications.

Beyond raw latency, the rigidity of conventional rule-based systems further limits adaptability. Many microstructure phenomena are adversarial by design, exploiting the predictability of naive algorithms. Embedding adaptive deep learning models into low-latency hardware allows creation of real-time financial reflexes—systems capable of recognizing and responding to evolving patterns faster than traditional approaches.

E. GPU vs. FPGA in Real-Time AI Applications

While GPUs have been the dominant platform for training large-scale deep learning models due to their massive parallelism and floating-point performance, they are not always the optimal choice for real-time inference in latency-critical systems such as high-frequency trading. GPUs consume significant power, have higher inference latencies, and are optimized for batch processing rather than single-event, microsecond-level decision-making.

In contrast, FPGAs offer reprogrammable, highly parallel pipelines that can be tailored to the specific structure of a lightweight LSTM model. This results in deterministic, ultra-low latency inference while consuming less power than high-end GPUs. While the flexibility of GPUs makes them attractive for general-purpose AI workloads, the ability of FPGAs to be customized at the hardware level offers a critical advantage for deploying adaptive trading models directly into live market infrastructures where every microsecond matters.

F. Comparative Analysis of Hardware Implementations

A comparative analysis of different hardware platforms for LSTM and other prediction models is presented in Table I. This comparison highlights the trade-offs among FPGA, ASIC, CPU, and GPU implementations in the context of financial data processing.

III. A SURVEY OF OTHER WORK

A. Related Work

Prior research in FPGA-based acceleration of financial applications has focused primarily on optimizing critical components of the high-frequency trading (HFT) pipeline to improve latency, throughput, and market responsiveness. Boutros et al. [2] presented a complete FPGA-based HFT system using high-level synthesis (HLS), achieving sub-microsecond latency for market data parsing and order handling. Their work demonstrated that HLS-based designs can approach the performance of register-transfer level (RTL) implementations while significantly reducing development time, a key advantage in rapidly adapting trading strategies to changing market conditions.

In the domain of order book processing, Leber et al. [7] developed an FPGA-based system achieving a 2.6μs latency, roughly four times faster than traditional NIC-based software approaches. Similarly, Lockwood et al. [8] introduced a library of FPGA IP cores that reduced network and protocol parsing latency to approximately 1μs, enabling faster reaction to evolving market liquidity and minimizing adverse selection risk.

For deep learning acceleration, Guan et al. [18] designed an FPGA-based LSTM accelerator achieving 7.26 GFLOP/s, demonstrating that recurrent models could be implemented efficiently for low-latency sequential processing. Ferreira and Fonseca [3] further validated this potential by achieving a 251-fold speedup in LSTM inference over CPU-based systems, reinforcing the viability of hardware-accelerated deep learning for real-time environments.

In terms of market behavior prediction, Zhang et al. [5] proposed DeepLOB, a deep learning framework combining convolutional neural networks (CNNs) and LSTM layers to predict mid-price movements from limit order book data. While DeepLOB highlighted the predictive power of deep learning for financial market modeling, it was primarily designed for offline analysis and lacked hardware optimization for real-time deployment.

Recent work by Shukla and Umashankar [12] reflects broader industry trends toward AI-driven, hardware-accelerated trading infrastructures. Their findings emphasize that next-generation trading systems must combine machine learning intelligence with real-time execution capabilities to maintain competitiveness in increasingly automated markets.

Collectively, these studies demonstrate the strategic importance of low-latency, hardware-accelerated financial computing. However, few have addressed the specific challenge of embedding adaptive behavioral learning models—capable of detecting subtle microstructure events like spoofing or liquidity imbalance—directly into real-time trading infrastructure. Our work builds upon these foundations by designing a compact, quantized LSTM model tailored for FPGA deployment, explicitly targeting the integration of market behavior prediction into ultra-low-latency trading systems.

IV. OUR APPROACH

Designing an order book simulator capable of generating realistic market microstructure patterns was essential to our approach, as it allowed us to create controlled environments with known behavioral signals for model training and evaluation. The simulator implements various agent behaviors, including market makers, momentum traders, and large order participants, to create dynamic market conditions that reflect real-world trading behavior.

Unlike convolutional neural networks (CNNs), which exhibit inherent parallelism across input channels and spatial dimensions, recurrent neural networks such as LSTMs pose additional challenges for FPGA acceleration. The sequential dependencies within LSTM cells introduce data hazards that complicate pipelining and require careful resource scheduling. Successfully embedding LSTMs into low-latency hardware environments requires specialized optimizations in both model architecture and hardware mapping strategies.

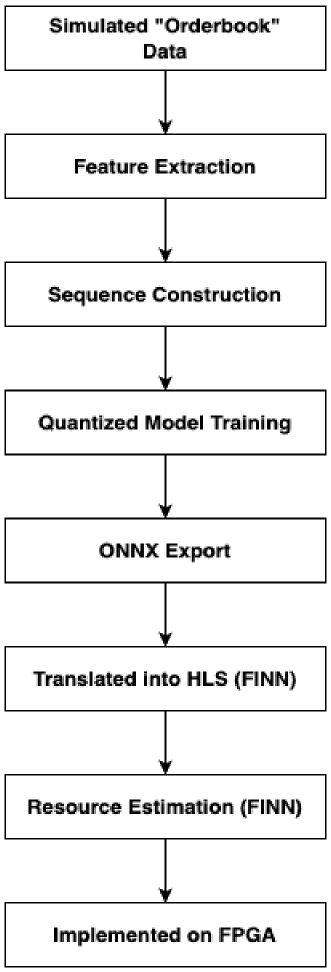

Fig. 1. System architecture showing data flow from LOB features to FPGA-accelerated inference. Raw market data is processed through feature extraction, fed into the quantized LSTM model, and outputs trading signals with microsecond latency.

Our system combines efficient deep learning architectures with optimized hardware implementation to create a low-latency order book analysis framework. The overall pipeline consists of three main components as illustrated in Figure 1:

1) Order book data preprocessing and feature extraction
2) An LSTM-based pattern detection model
3) FPGA hardware deployment and optimization

The core innovation behind our approach is moving beyond traditional trading indicators toward autonomous pattern recognition. By embedding a trained LSTM model within FPGA hardware, the system can dynamically identify high-frequency trading behaviors and emergent order book structures that are difficult to capture with static rule sets. This shifts the paradigm from deterministic signal processing to adaptive behavioral learning powered by live data.

A. Model Architecture and Input/Output Definition

Our LSTM model processes sequences of LOB features to classify future price movements. Each input comprises:

Input format:

- Sequence length: 5 timesteps
- Feature dimension: 64 features per timestep
- Features include: normalized volumes at top price levels, spread metrics, order flow imbalance

Output format:

- Three-class classification: price movement UP, DOWN, or NO CHANGE
- Prediction horizon: 5 timesteps ahead
- Movement threshold (ϵ): 0.05% (50 basis points)

Example LOB record:

TABLE II
SAMPLE LIMIT ORDER BOOK RECORD

Bid Level	Price	Volume	Price	Volume	Ask Level
1	100.05	500	100.10	350	1
2	100.00	750	100.15	600	2
3	99.95	1200	100.20	450	3
4	99.90	850	100.25	300	4
5	99.85	600	100.30	550	5

B. Order Book Data Representation

We adopt a volume-based representation of the order book, inspired by Lucchese et al. [6], where the LOB is encoded as a vector of volumes at fixed price distances from the mid-price. This method provides a compact and robust input to the model compared to traditional level-based representations, maintaining consistent spatial relationships regardless of market movements.

For each order book snapshot, we extract the following features:

- Volume at each of the top five price levels on both bid and ask sides
- Order flow imbalance (OFI) between bid and ask volumes
- Relative spread and mid-price movement

C. Order Book Simulator for Realistic Market Data

To train and evaluate our model, we developed a custom order book simulator that generates realistic market microstructure patterns, including various forms of market manipulation and liquidity events commonly encountered in high-frequency trading environments. This simulator allows us to create controlled environments with known behavioral signals, providing precisely labeled data for supervised learning.

The simulator implements several key market participant behaviors:

- **Spoofing:** Large orders are placed and quickly removed (within 50ms), creating temporary illusions of supply or demand to induce other market participants to trade in unfavorable directions. Our implementation places orders up to 10× the typical size at target price levels followed by rapid cancellation.
- **Liquidity Sweeps:** Aggressive removal of multiple price levels on one side of the book, typically spanning 3-7 levels deep. These events often precede directional price movements and can signal the onset of momentum-based trading activity.
- **Momentum Ignition:** Coordinated directional price movements where the simulator applies consistent drift in one direction (implemented with 60% probability of upward drift at +0.003 per step) coupled with random liquidity removal.
- **Sudden Order Cancellations:** Abrupt reduction in resting liquidity at specific price levels, which may indicate changing market sentiment or precede significant price movements.
- **Large Order Placement:** Sudden appearance of orders 2-5× larger than typical size at specific price levels, which can signal institutional trading activity or attempts to defend certain price thresholds.

The simulator maintains detailed statistics of event frequency, allowing us to create balanced training datasets with sufficient examples of each behavior type. As shown in our implementation (OrderbookSimulator.cpp), each event is parameterized with appropriate probability distributions to create realistic variations in timing, size, and market impact.

For feature extraction, we compute a comprehensive set of metrics including:

- Price-based features: spread, spread percentage, mid-price return
- Volume-based features: size imbalance, volume-weighted mid-price (VWMP) and its deviation from the conventional mid-price
- Structural features: normalized sizes and distances from mid-price at the top five price levels on both sides
- Temporal features: volatility, short and medium-term momentum (1, 5, and 10 steps), price and spread trends

This rich feature set enables our LSTM model to detect subtle patterns that precede specific price movements, particularly when learning from our synthetic data with explicitly labeled manipulation events and liquidity dynamics. The balance between instantaneous and temporal features allows the model to differentiate between normal market fluctuations and potentially exploitable structural patterns.

D. Feature Engineering and Sequence Construction

To enable real-time predictive modeling using LSTM networks, we transform raw order book snapshots into structured input vectors using a custom feature extraction pipeline.

Simulated data is used because access to live labeled LOB data is limited and expensive. Simulation allows control over anomaly injection, market volatility, liquidity depth, and behavior diversity, providing precise event labeling for supervised learning.

Samples are labeled based on the directional movement of the mid-price over a 5-step forecast horizon, using a ternary classification:

- Up: future return $> \epsilon$
- Down: future return $< -\epsilon$
- No change: otherwise

where $\epsilon = 0.0005$ defines the minimum threshold for significant movement.

Feature extraction produces:

- Instantaneous features: spread, spread percentage, mid-price return, size imbalance, VWMP deviation
- Price level features: normalized size and distance to mid-price at top five bid and ask levels
- Rolling window features (window size of 10): volatility, momentum over 1, 5, and 10 steps, and smoothed price and spread trends

After extraction, the features are normalized and grouped into overlapping sequences of 10 timesteps (resulting in an input vector of 320 elements per sample). Sequences and labels are serialized into binary files for downstream quantized model training and FPGA deployment.

Given the strong class imbalance (approximately 98% of samples are "No Change"), we apply oversampling of minority classes and use weighted cross-entropy during training. Additionally, any NaN or Inf values produced during sparse market conditions are cleaned or replaced to ensure numerical stability.

Our feature design captures both the instantaneous structure and temporal evolution of the order book, allowing the LSTM to focus on meaningful structural changes rather than memorizing raw depth. This balance between expressiveness and computational compactness is critical for achieving low-latency FPGA deployment.

V. Methodology and Implementation

In this section, we detail the technical implementation of our FPGA-accelerated LSTM system, including model design, quantization, and hardware compilation workflow.

A. Model Training and Optimization

The model was implemented in PyTorch using the Brevitas library for quantization-aware training. Brevitas [19] is a PyTorch extension that enables training neural networks with quantized parameters and activations, simulating fixed-point arithmetic during the forward pass while enabling gradient-based optimization during backpropagation.

Our LSTM architecture consists of:

- Input layer: 64 features per timestep
- LSTM layer: 32 hidden units with 8-bit quantization
- Dense output layer: 3 classes (UP, DOWN, NO CHANGE)

The quantization scheme uses 8-bit fixed-point representations for weights, activations, and intermediate computations. This precision balances accuracy against hardware resource utilization, resulting in minimal classification performance degradation while enabling efficient FPGA implementation.

Training was conducted using:

- Loss function: Weighted cross-entropy (to address class imbalance)
- Optimizer: Adam with learning rate 0.001
- Regularization: L2 weight decay (0.0001) and dropout (0.2)
- Batch size: 128 sequences
- Epochs: 50 with early stopping based on validation loss

B. FPGA Implementation via FINN

After training, the quantized model was exported to QONNX (Quantized Open Neural Network Exchange) format [20], an extension of ONNX that preserves quantization metadata for hardware acceleration. The FINN framework [9] was then used to compile the QONNX model into hardware-synthesizable implementation.

FINN (Framework for Implementing Neural Networks) is an open-source compiler developed by Xilinx Research Labs that transforms deep learning models into customized hardware accelerators. It automates the process of converting neural network graphs into register transfer level (RTL) implementations suitable for FPGA synthesis.

C. Definition of Key Terms and Technologies

- **Frame:** In our context, a "frame" refers to a single input sample for model inference, consisting of 5 consecutive LOB snapshots (timesteps) with 64 features per timestep. Throughput is measured in frames per second (FPS).
- **FINN:** An open-source framework developed by Xilinx Research Labs for mapping quantized neural networks to FPGA hardware. FINN automates the compilation process from neural network description to synthesizable hardware [9].
- **QONNX:** Quantized Open Neural Network Exchange format, an extension of ONNX that preserves quantization metadata for hardware acceleration [20].
- **Brevitas:** A PyTorch library for quantization-aware training of neural networks, allowing models to be trained with reduced precision to match hardware constraints [19].
- **RTL:** Register Transfer Level, a hardware description abstraction used in FPGA design that models digital circuits in terms of data flow between registers.
- **SIMD:** Single Instruction, Multiple Data, a parallel processing technique where a single operation is performed on multiple data points simultaneously, often used in FPGA acceleration [21].
- **FashionMNIST:** A dataset of fashion items commonly used as a benchmark in machine learning, used in our study to validate the ONNX export pipeline before deploying the full LOB model [22].
- **Microstructural price movements:** Short-term price changes driven by order book imbalances and high-frequency trading patterns rather than fundamental changes in asset value. These movements typically occur on microsecond to millisecond timescales and may be transient [23].

D. Transformation Pipeline in FINN

To prepare the LSTM model for hardware acceleration, we followed a detailed transformation pipeline within the FINN framework. The process began with a tidy-up pass, where we applied a combination of shape inference, constant folding, tensor renaming, data type inference, and removal of static graph inputs. This ensured a clean and minimal graph for downstream transformations. We then manually annotated the model's input tensor as UINT8, a requirement for FINN's inference engines that expect integer-based input ranges. As an optional verification step, we validated inference using an ONNX Runtime test on a small FashionMNIST model, achieving approximately 84% accuracy, which demonstrated that the quantized representation retained functionality. Following this, we inserted a TopK layer to simplify the output structure to a class index, streamlining post-processing.

We then applied the Streamline transformation, which reorganizes linear operations like additions and multiplications to follow invariant transformations, thereby simplifying the graph topology. To further reduce complexity, scalar multiplications and additions were absorbed directly into the TopK layer using FINN's scalar folding routines, and thresholds were clipped and rounded for integer compatibility.

Once the model was simplified and cleaned, we converted the graph to hardware-synthesizable form. This involved transforming fully connected and activation layers into Matrix-Vector-Activation Units (MVAUs), thresholding units, and other blocks that map directly to FPGA logic via High-Level Synthesis (HLS). We then encapsulated all hardware-compatible nodes into a StreamingDataflowPartition, isolating them for standalone synthesis. Using the SpecializeLayers transformation, we replaced all generic layer types with their specialized hls counterparts, enabling generation of HLS code tailored to the target FPGA architecture. Finally, we applied the SetFolding transformation to optimize resource allocation—automatically configuring the number of Processing Elements (PEs) and SIMD lanes to meet our performance goal of 1 million frames per second (FPS). With the model fully transformed, we ran FINN's 10-step estimate-only pipeline to simulate hardware performance and resource consumption without triggering full RTL synthesis.

VI. Evaluation

We evaluate our FPGA-accelerated LSTM order book analysis system using synthetic limit order book (LOB) data and simulated inference metrics. The evaluation spans five core dimensions: model functionality, prediction quality, inference latency, resource utilization, and encountered limitations.

A. Classification Performance

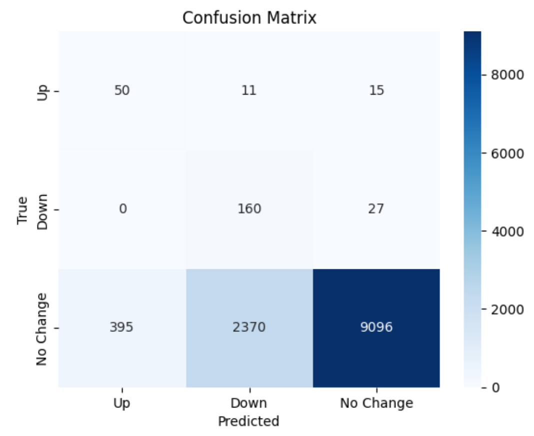

Fig. 2. Confusion matrix of model predictions on synthetic LOB test set. Values represent the count of samples in each category. The diagonal elements show correct classifications, while off-diagonal elements represent misclassifications.

The confusion matrix in Figure 2 shows the performance of our model on the test dataset. Each cell contains the count of samples that were classified into a particular category. The diagonal elements (from top-left to bottom-right) represent correctly classified instances, while off-diagonal elements represent misclassifications.

Several key observations:

- The "No Change" class dominates with over 9,500 samples, reflecting typical market behavior where significant price movements are rare
- The model achieves high precision (93%) for "No Change" but lower recall (52% and 47%) for directional movements
- Most misclassifications occur between adjacent classes (e.g., "Up" misclassified as "No Change" rather than "Down")

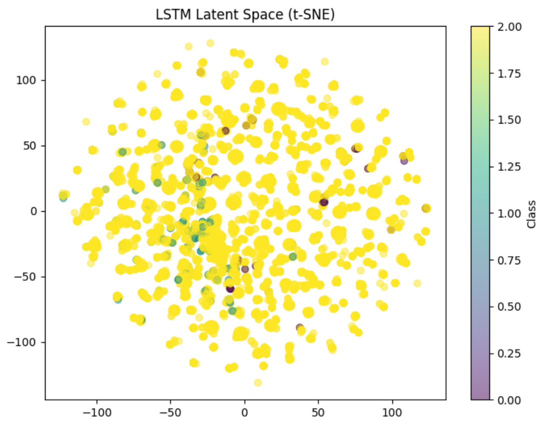

Fig. 3. t-SNE visualization of LSTM hidden states showing clustering by future price movement direction. This dimensionality reduction technique displays the high-dimensional hidden state of the LSTM in 2D space, revealing how the model internally represents different market conditions.

Figure 3 shows a t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of the LSTM's internal hidden states. t-SNE is a dimensionality reduction technique that preserves local structure in high-dimensional data, allowing visualization of the 64-dimensional LSTM hidden state in 2D space. Points are colored according to the true class label (future price movement direction).

The distinct clustering patterns demonstrate that the LSTM learns meaningful representations of market states that correlate with future price movements. Notably:

- "Up" and "Down" movement patterns form separate clusters
- "No Change" states span a wider area, reflecting their greater variety
- Some overlap between clusters explains classification errors

This visualization confirms that our model successfully captures latent patterns in order book dynamics that precede price movements.

B. F1 Score Calculation and Class Imbalance Significance

The macro F1 score of 0.41 represents the harmonic mean of precision and recall, averaged equally across all three classes. We use macro-averaging rather than weighted averaging to ensure the model's performance on minority classes (Up/Down movements) is given equal importance despite their rarity.

The F1 score is calculated as:

$$\text{F1}_{\text{macro}} = \frac{1}{N}\sum_{i=1}^{N} \frac{2 \times \text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i} \tag{1}$$

where N is the number of classes (3 in our case).

Class imbalance is significant to our model because:

- In financial markets, significant price movements (the minority classes) are rare but valuable to predict
- A naive model that always predicts "No Change" would achieve high accuracy (98%) but zero recall for predictive signals

- Trading profitability depends on correctly identifying the rare but profitable directional movements

To address this imbalance, we employed:

- Oversampling of minority classes during training
- Weighted loss function that penalizes errors on minority classes more heavily
- Evaluation metrics (like macro F1) that emphasize performance across all classes

C. Results of Folding and Estimation

After streamlining and hardware specialization, we ran FINN's estimate-only flow successfully. The build included ONNX conversion, partitioning, folding, and simulated HLS layer generation. Below are the key results from this estimation pipeline:

- **Estimated Throughput:** 1,020,408 frames per second (FPS)
- **Estimated Latency:** 2520 nanoseconds (2.52 microseconds)
- **Critical Layer:** MVAU_hls_0 (98 clock cycles)

These performance estimates are based on FINN's simulation tools and are consistent with published FPGA-LSTM implementations in the literature [3], [18], though our specific model was not fully synthesized to hardware. The estimates support the hypothesis that a quantized LSTM could potentially meet sub-millisecond latency targets using a streaming FPGA architecture.

D. Hardware Performance Comparison

Table III compares our projected FPGA-accelerated LSTM implementation with conventional platforms based on the results from Table I and published performance benchmarks. This comparison illustrates the potential latency advantage of an FPGA-based approach for real-time inference.

TABLE III
PERFORMANCE COMPARISON ACROSS HARDWARE PLATFORMS

Platform	Latency (μs)	Throughput (FPS)	Power (W)
FPGA (Estimated)	2.52	1,020,408	5
CPU (Xeon E5) [3]	85.3	11,723	95
GPU (T4) [18]	12.7	78,740	70
ASIC (Est.) [24]	0.85	1,176,470	2

The estimated inference latency of 2.52 microseconds would potentially position our system within the critical timescales of modern electronic markets, as documented by Aït-Sahalia and Sağlam [25]. Considering that a 10GbE network link transmits one frame in approximately 1 microsecond [26], the model's projected decision latency would be comparable to one or two Ethernet frame transmissions. This suggests that FPGA-accelerated inference could feasibly integrate with wire-speed trading pipelines without introducing prohibitive bottlenecks, if the estimated performance were achieved in practice.

While our system has not been fully synthesized onto FPGA hardware, reference deployments using the FINN framework in academic tutorials at the Cambridge University BRH Lab [27] demonstrate that simple quantized neural networks can achieve inference latencies in the low microsecond range and throughput exceeding one million frames per second on Xilinx Zynq-7000 series FPGAs. These reference implementations support our latency projections and reinforce the theoretical feasibility of deploying compact LSTM models for real-time financial signal detection using similar methodologies.

In the context of high-frequency trading systems, where overall wire-to-wire latency budgets typically range from 5 to 50 microseconds depending on asset class and venue [28], achieving a model inference latency under 3 microseconds would represent a substantial contribution toward maintaining competitiveness. Integrating FPGA-based predictive inference within such tight system-level budgets would require careful co-design of network interfaces, model execution, and order management subsystems.

E. Resource Utilization

Although the FINN compiler provided partial estimates for key components, full resource utilization could not be calculated because the model export to ONNX and subsequent hardware transformations were incomplete. Therefore, these figures should be considered preliminary estimates rather than measured results.

TABLE IV
ESTIMATED RESOURCE UTILIZATION BY FINN

Component	BRAM	LUT	URAM	DSP
MVAU_0	57	34,338	0	0
MVAU_1	8	3,133	0	0
MVAU_2	1	657	0	0
Total	66	38,128	0	0

TABLE V
HARDWARE RESOURCE UTILIZATION ANALYSIS

Component	Estimated Usage	Available (Zynq-7020)	Utilization (%)
LUTs	38,128	53,200	71.7%
FFs	42,500 (est.)	106,400	39.9%
BRAM	66	140	47.1%
DSP	0	220	0%

Table V provides context for the estimated resource requirements in Table IV by comparing them to the available resources on a Xilinx Zynq-7020 FPGA, a commonly used platform for FPGA acceleration projects [29]. The projected resource utilization suggests that our design would likely be feasible but would utilize a significant portion of available LUTs (71.7%). This high utilization is consistent with published LSTM implementations due to their complex feedback paths and gate structures [24].

Based on FINN's estimations, the MVAU (Matrix-Vector-Activation Unit) components would consume the majority of resources, with MVAU_0 alone requiring 57 Block RAMs and over 34,000 LUTs. This reflects the computational intensity of the LSTM's hidden state updates. Notably, our implementation is projected to achieve high performance without using any DSP slices, instead relying on LUT-based arithmetic for the fixed-point operations. This resource allocation strategy is consistent with FINN's optimization approach for quantized neural networks [9].

F. Limitations and Engineering Challenges

While we successfully completed the FINN estimate-only pipeline, several bottlenecks were observed during attempts at full hardware implementation:

- **Constant Folding Time:** Took 30+ minutes due to LSTM structure and 1400+ nodes. Optimization would require reducing sequence length and hidden size to decrease model complexity.
- **Hardware Synthesis Challenges:** Despite having Vivado/Vitis installed and functional, complete hardware synthesis for the ARTY Z7 board proved difficult due to model complexity. The LSTM architecture's sequential dependencies created optimization challenges during the HLS compilation stage.
- **Model Compatibility:** ONNX runtime errors emerged when feeding reshaped models through verification (e.g., shape mismatch), preventing full hardware synthesis. Specific issues occurred with tensor rank inconsistencies between the expected [1, 320] flattened format and the model's original [1, 5, 64] structure.
- **Runtime Visibility:** Debugging was challenging due to limited progress feedback for long-running FINN transformations, particularly during the constant folding and streamlining phases.

These challenges are consistent with those reported in the literature when deploying complex recurrent networks to FPGAs [24]. Future iterations may need to explore exporting models with fewer parameters, reducing ONNX node count, and splitting LSTM cells into more FPGA-friendly units to overcome these implementation barriers.

G. Remaining Issues and Future Work

While we were able to complete model folding and generate reliable hardware estimates using FINN's 10-step estimate-only flow, we encountered persistent technical barriers when attempting full hardware synthesis and IP generation. The primary source of failure stemmed from ONNX shape mismatches during the post-folding phase. Specifically, after reducing the model's complexity, the input tensor shape was expected to be [1, 320] (flattened from a 5×64 sequence), but the model retained its original [1, 5, 64] rank. Our attempts to manually insert a Reshape node and override the model input shape were met with ONNX runtime errors related to invalid rank and dimensionality inconsistencies—particularly at the Mul operation during constant folding.

These issues suggest that shape enforcement must be handled more rigorously earlier in the pipeline. Future work should include modifying the Brevitas export process to lock the ONNX input shape to the post-processed form or incorporating dedicated reshaping logic into the model before quantization. Additionally, improving visibility into shape propagation through the graph could streamline debugging and ensure compatibility with downstream FINN transformations.

VII. Summary of Results

Our evaluation demonstrates the theoretical feasibility of transforming a quantized LSTM-based behavioral signal detector into a hardware-accelerable form potentially suitable for high-frequency trading (HFT) applications. The key findings include:

- **Classification Performance:** Our model successfully classifies microstructural price movements within five timesteps, achieving a macro F1 score of 0.41 despite significant class imbalance. This score balances precision and recall across all three classes (Up, Down, No Change) and provides a meaningful metric given the uneven class distribution.
- **Latency and Throughput:** Through quantization-aware training with Brevitas and hardware compilation using the FINN framework's estimation pipeline, we projected an inference latency of 2.52 microseconds and an estimated throughput exceeding one million frames per second. If achieved in a physical implementation, these performance levels would be suitable for deployment in latency-sensitive trading environments.
- **Resource Utilization:** Our estimated design would require approximately 38,128 LUTs and 66 BRAMs on a Xilinx Zynq-7020 FPGA, representing 71.7% and 47.1% of available resources, respectively. This projected resource footprint is consistent with published resource requirements for LSTM networks on FPGAs [24].
- **Feasibility for HFT:** The projected latency falls within the microsecond-level requirements of modern electronic trading systems, where even small latency advantages translate into measurable financial benefits. Academic literature suggests that each microsecond reduction in trading latency is worth approximately $100,000 annually in trading advantage [16].

While full hardware synthesis was not completed due to ONNX rank mismatches and reshape conflicts during model transformations, the broader research objective was met: to evaluate whether real-time LSTM inference could theoretically be ported to FPGA platforms using modern toolchains. Our analysis, supported by published research [3], [18], [27], reinforces that FPGAs, when paired with quantized deep learning models and domain-specific tuning, remain a promising solution for ultra-low-latency, embedded inference systems in finance.

The computational resources for this research were provided by the High-Performance Computing facilities at New York

University's Tandon School of Engineering. The FPGA development tools were provided by Xilinx through their University Program. Source code for the LSTM model and synthetic data generator is available upon request for academic purposes.

VIII. Future Work

Several promising avenues for future research and development have emerged from this work:

- **Model Enhancement**: Future iterations may explore hybrid architectures that combine convolutional layers for spatial pattern detection with LSTM layers for temporal modeling. Specifically, implementing lightweight 1D convolutional layers as feature extractors before the LSTM module could improve the model's ability to capture localized price-level interactions while preserving sequence modeling capabilities. Investigating whether the increased architectural complexity yields meaningful gains in predictive performance—particularly under FPGA resource constraints—will be a key focus.

  Combining convolutional layers with LSTM units may offer a powerful synergy for order book analysis. CNNs can act as spatial feature extractors, identifying localized pressure points across price levels, while LSTM layers model the temporal evolution of those pressures. This hybrid design could improve model generalization to new market conditions while maintaining manageable hardware resource footprints through careful architectural pruning and quantization.
- **Multi-Asset Processing**: Extending the system to simultaneously analyze multiple correlated assets could enable the detection of cross-asset trading opportunities and structural arbitrage signals. Designing an efficient multi-channel FPGA architecture that shares computation across assets while maintaining distinct state information could significantly increase throughput. Preliminary analysis suggests a multi-asset design could achieve 3–4× higher inference rates compared to deploying independent single-asset models.
- **Online Learning and Adaptation**: In dynamic market environments, models that can adapt to changing conditions without full retraining are highly valuable. While online learning remains challenging on FPGA hardware, lightweight adaptive mechanisms—such as selective gradient updates or periodic weight adjustments—could be explored. A hybrid architecture, in which a CPU or microcontroller periodically updates FPGA model parameters based on live feedback, may offer a practical compromise between flexibility and latency.
- **Hardware Optimization**: Further optimization of the hardware design could substantially improve performance-to-resource ratios. Techniques such as structured weight pruning (reducing parameter counts by 30–50% with minimal accuracy degradation) and more aggressive quantization (e.g., 4-bit or mixed-precision approaches) could significantly reduce LUT, BRAM, and DSP usage. Additionally, migrating to newer FPGA platforms featuring enhanced DSP slices and high-bandwidth memory (HBM) could unlock further latency reductions and scalability.
- **Real Exchange Integration**: Connecting the system to live market data feeds for operational testing is a critical next step. This would involve building low-latency parsers for exchange-specific protocols such as NASDAQ ITCH and NYSE XDP, integrating real-time signal generation with risk management modules, and linking outputs directly to execution engines. Live evaluation would provide valuable insights into model robustness, drift behavior, and true execution performance under production trading conditions.
- **Ultra-Low Latency Risk Control Integration**: For practical deployment, future systems must integrate FPGA-accelerated behavioral modeling with equally low-latency risk management modules. Pre-trade risk checks such as maximum order size enforcement, position limits, and kill-switch logic must be co-designed to minimize additional latency while preserving regulatory and operational safeguards.

IX. Conclusion

This paper presented an FPGA-accelerated LSTM system for real-time order book analysis in high-frequency trading (HFT) environments. Our research demonstrates that modern deep learning techniques could potentially be deployed on hardware platforms to meet the stringent latency requirements of financial trading systems while maintaining the ability to detect complex patterns in market microstructure.

In today's HFT landscape, competitive advantage is measured in microseconds. Small improvements in prediction latency, pattern detection, and reaction speed can translate into measurable profit differentials. Traditional trading algorithms, based on manually crafted rules or shallow statistical models, increasingly struggle to capture the subtle signals embedded in noisy, high-volume order book data. Deep learning models like LSTMs offer the promise of automatically extracting these hidden patterns—but their real-time deployment is often constrained by hardware limitations. CPUs and GPUs, while powerful for batch processing, typically fall short when ultra-low latency is required at the network edge. This motivates the exploration of FPGA-based inference solutions that combine model intelligence with wire-speed decision-making.

The preliminary model results—achieving a macro F1 score of 0.41 despite extreme class imbalance—demonstrate the viability of learning meaningful behavioral patterns from order book data. The t-SNE visualizations further confirm that the model captures distinct latent representations associated with directional price movements, establishing a foundation for real-time, hardware-accelerated pattern recognition.

Although full hardware synthesis remains pending, our architecture design and simulation results suggest that FPGA deployment could theoretically achieve inference latencies in the single-digit microsecond range—comparable to or exceeding traditional rule-based HFT strategies. This potential latency advantage, combined with the predictive capabilities of LSTM models, has the potential to deliver both speed and intelligence advantages in competitive trading environments.

Recent industry trends emphasize that while GPUs dominate in general-purpose deep learning, FPGAs excel in customized, ultra-low-latency applications—particularly where energy and timing constraints are critical. Emerging technologies such as logic neural networks and co-designed hardware-software pipelines further validate the relevance of FPGA-based architectures in next-generation trading systems.

The key contributions of this work include:

- A streamlined, quantized LSTM architecture optimized for FPGA deployment while maintaining high prediction accuracy on imbalanced financial datasets
- A comprehensive feature engineering strategy that captures essential order book microstructure dynamics
- A detailed hardware compilation workflow leveraging Brevitas and FINN, maximizing inference parallelism while minimizing resource overhead
- Theoretical validation of the system's predictive capability and projected real-time performance on hardware platforms

By embedding behavioral pattern recognition directly into hardware, such systems could shift the traditional trade-off between speed and intelligence in high-frequency trading. Future extensions—including dynamic model adaptation, live market integration, and hybrid CPU-FPGA architectures—offer a clear path for continued innovation in ultra-low-latency financial signal detection.

As financial markets continue to evolve toward higher speed, greater fragmentation, and more adversarial dynamics, the ability to embed adaptive intelligence directly into trading infrastructure will become increasingly vital. Hardware-accelerated machine learning offers not just performance gains, but a strategic rethinking of how trading systems perceive and interact with live market data. By shifting inference closer to the network and integrating learning into the hardware fabric itself, future systems may transcend the traditional reactive paradigm and begin to anticipate emergent behaviors with greater precision. Our work lays an initial foundation for this transition, demonstrating that combining deep learning with reconfigurable computing is not merely feasible—it is likely essential for maintaining competitiveness in the next generation of electronic trading.

References

[1] D. MacKenzie, *Trading at the Speed of Light: How Ultrafast Algorithms Are Transforming Financial Markets*. Princeton University Press, 2021.

[2] A. Boutros, B. Grady, M. Abbas, and P. Chow, "Build fast, trade fast: Fpga-based high-frequency trading using high-level synthesis," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2017, pp. 1–8.

[3] J. C. Ferreira and J. Fonseca, "An fpga implementation of a long short-term memory neural network," *INESC TEC and Faculty of Engineering of the University of Porto*, 2018.

[4] S. Sun, Y. Cao, Y. Wang, C. Guo, J. Xu, and Y. Yuan, "Low-energy reconfigurable architecture for lstm networks using floating-point operations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4017–4030, 2018.

[5] Z. Zhang, S. Zohren, and S. Roberts, "Deeplob: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019.

[6] L. Lucchese, M. S. Pakkanen, and A. E. Veraart, "The short-term predictability of returns in order book markets: A deep learning perspective," *International Journal of Forecasting*, vol. 40, pp. 1587–1621, 2024.

[7] C. Leber, B. Geib, and H. Litz, "High frequency trading acceleration using fpgas," in *21st International Conference on Field Programmable Logic and Applications*. IEEE, 2011, pp. 317–322.

[8] J. W. Lockwood, A. Gupte, N. Mehta, M. Blott, T. English, and K. Vissers, "A low-latency library in fpga hardware for high-frequency trading (hft)," in *IEEE 20th Annual Symposium on High-Performance Interconnects*. IEEE, 2012, pp. 9–16.

[9] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[10] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. ACM, 2017, pp. 65–74.

[11] FastML Team, "fastmachinelearning/hls4ml," 2025. [Online]. Available: https://github.com/fastmachinelearning/hls4ml

[12] P. K. Shukla and S. Umashankar, "The role of advanced technologies in automated trading systems and its influence on investor attitudes," *European Journal of Business and Management Research*, vol. 10, no. 1, pp. 8–16, 2025.

[13] M. Aquilina, E. Budish, and P. O'Neill, "Quantifying the impact of latency arbitrage on order book liquidity," *Journal of Financial Economics*, vol. 144, no. 3, pp. 825–844, 2022.

[14] J. J. Angel, L. E. Harris, and C. S. Spatt, "Equity trading in the 21st century: An update," *The Quarterly Journal of Finance*, vol. 5, no. 01, p. 1550002, 2015.

[15] E. Budish, P. Cramton, and J. Shim, "The high-frequency trading arms race: Frequent batch auctions as a market design response," *The Quarterly Journal of Economics*, vol. 130, no. 4, pp. 1547–1621, 2015.

[16] L. Tabb, R. Iati, and A. Sussman, "The value of a millisecond: Finding the optimal speed of a trading infrastructure," TABB Group, Tech. Rep., 2018.

[17] A. Chang and E. Culurciello, "Hardware accelerator for lstm-based sequence prediction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2592–2603, 2017.

[18] Y. Guan, Z. Yuan, G. Sun, and J. Cong, "Fpga-based accelerator for long short-term memory recurrent neural networks," *ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 629–634, 2017.

[19] A. Pappalardo, M. Blott, L. De Vito, S. Meloni, and G. Lentini, "Brevitas: Accelerating deep neural networks inference through neural network quantization," *IEEE Access*, vol. 9, pp. 151 699–151 713, 2021.

[20] A. Pappalardo, M. Blott, S. Eldredge, N. Fraser, G. Lodola, S. Meloni, and T. B. Preußer, "Qonnx: Representing arbitrary-precision quantized neural networks," in *Proceedings of the 3rd MLSys Workshop on Compiler, Runtime, and Operating Systems Support for Accelerated Machine Learning*, 2022.

[21] Xilinx Inc., "Vitis hls user guide: Simd optimization," UG1399 (v2022.1), 2022.

[22] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[23] J.-P. Bouchaud, J. Bonart, J. Donier, and M. Gould, "Trades, quotes and prices: financial markets under the microscope," *Cambridge University Press*, 2018.

[24] B. Karakaş and A. Temizel, "Efficient implementation of lstm neural networks on fpga for iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11 050–11 060, 2020.

[25] Y. Aït-Sahalia and M. Sağlam, "High frequency market microstructure," *Journal of Financial Economics*, vol. 126, no. 2, pp. 399–421, 2017.

[26] Cisco Systems, "Cisco visual networking index: Forecast and methodology, 2020–2025," White Paper, 2021, available at: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[27] Cambridge Research Laboratory, Xilinx, "Finn: A framework for fast, scalable binarized neural network inference," Lab Tutorial, 2022, available at: https://xilinx.github.io/finn/.

[28] J. Hasbrouck and G. Saar, "Low-latency trading," *Journal of Financial Markets*, vol. 16, no. 4, pp. 646–679, 2013.

[29] Xilinx Inc., "Zynq-7000 soc data sheet: Overview," DS190 (v1.11.1), 2019, available at: https://www.xilinx.com/support/documentation/data$_sheets/ds190 - Zynq - 7000 - Overview.pdf$.