LightTrader: A Standalone High-Frequency Trading System with Deep Learning Inference Accelerators and Proactive Scheduler

Sungyeob Yoo*§
sungyeob.yoo@kaist.ac.kr
KAIST

Hyunsung Kim*
hyunsungkim@rebellions.ai
Rebellions Inc.

Jinseok Kim
jinseok@rebellions.ai
Rebellions Inc.

Sunghyun Park
sunghyun.park@rebellions.ai
Rebellions Inc.

Joo-Young Kim
jooyoung1203@kaist.ac.kr
KAIST

Jinwook Oh
j.oh@rebellions.ai
Rebellions Inc.

***Abstract*—Recent research shows that artificial intelligence (AI) algorithms can dramatically improve the profitability of high-frequency trading (HFT) with accurate market prediction, overcoming the limitation of conventional latency-oriented approaches. However, it is challenging to integrate the computationally intensive AI algorithm into the existing trading pipeline due to its excessively long latency and insufficient throughput, necessitating a breakthrough in hardware. Furthermore, harsh HFT environments such as bursty data traffic and stringent power constraint make it even more difficult to achieve system-level performance without missing crucial market signals.**

In this paper, we present LightTrader, the world's first AI-enabled HFT system that incorporates an FPGA and custom AI accelerators for short-latency-high-throughput trading systems. Leveraging the computing power of brand-new AI accelerators fabricated in TSMC's 7nm FinFET technology, LightTrader optimizes the tick-to-trade latency and response rate for stock market data. The AI accelerators, adopting Coarse-Grained Reconfigurable Array (CGRA) architecture, which maximizes the hardware utilization from the flexible dataflow architecture, achieve a throughput of 16 TFLOPS and 64 TOPS. In addition, we propose both workload scheduling and dynamic voltage and frequency scaling (DVFS) scheduling algorithms to find an optimal offloading strategy under bursty market data traffic and limited power condition. Finally, we build a reliable and re-runnable simulation framework that can back-test the historical market data, such as Chicago Mercantile Exchange (CME), to evaluate the LightTrader system. We thoroughly explore the performance of LightTrader when the number of AI accelerators, power conditions, and complexity of deep neural network models change. As a result, LightTrader achieves $13.92\times$ and $7.28\times$ speed-up of AI algorithm processing compared to existing GPU-based, FPGA-based systems, respectively. LightTrader with multiple AI accelerators achieves up to 99.5% response rates, while LightTrader with the proposed workload scheduling and DVFS scheduling algorithm relieves the miss rate from 17.1% to 23.1%.

*These authors contributed equally to this work.

§Sungyeob Yoo performed this work as an intern at Rebellions Inc.

This research was supported in part by the Information Technology Research Center (ITRC) support program (IITP-2020-0-01847) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP) and in part by the National IT Industry Promotion Agency (NIPA) (R-20210319-010567, Development of AI accelerator and its software full stack for edge server system), both under the Ministry of Science and ICT (MSIT), Korea.

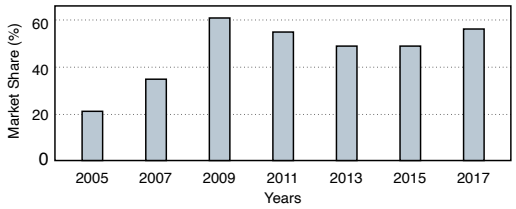

Fig. 1. HFT's share of US equity trading volumes [20]

I. Introduction

High-frequency trading (HFT) is an algorithmic trading method that uses high-performance computers to transact a large number of orders in fractions of a second [1]. HFT firms exploit the speed advantage over other traders to capture profit opportunities by predicting the market's short-term movements, thereby frequently making beneficial orders [2]. The activities of HFTs primarily improve the market quality by providing market fluidity, discovering the fair price, and resolving market inefficiencies [3]–[5]. Due to the great success of HFT firms over the last few decades, HFT has become dominant in modern exchanges, taking more than half of the total trading volume in the US equity market, as shown in Figure 1. As the competition has intensified, however, the market's short-term dynamics are becoming more and more chaotic, increasing the noise level and non-linearity of the high-frequency market data [6]–[8]. Accordingly, the conventional HFT strategies now show limitations in capturing profit opportunities, earning only marginal profit from the competition [9]–[11]. With the advent of artificial intelligence (AI) and machine learning (ML) technology, recent works [12]–[16] have proposed to utilize deep neural networks (DNNs) to capture hidden profit opportunities from the high-frequency market data, which was not possible by the conventional methods [17]–[19].

Despite the promising performance in predicting the market's movements, AI algorithms are difficult to deploy in an actual HFT system because the AI algorithms require excessive computation workload combined with the system's requirements for rapid response to irregular input. As the

market condition changes rapidly and frequently, the HFT system should make prompt actions by ingesting a vast amount of market data, which requires both low latency and high throughput processing [21], [22]. In the HFT process, *tick-to-trade*, defined as the process between receiving the market data and transferring the order, is comprised of multiple pipeline stages such as network packet processing, market data processing, and order transmission [21], [23]. The tick-to-trade process should be extremely short because the excessive latency reduces the probability of getting profit from short-living opportunities [8], [11], [24]. Meanwhile, the HFT system's limited throughput causes drops in the input data traffic [25] because the few input data may not meet the proposed deadline, thereby having a chance of missing crucial information in the market data. Although the AI-enabled HFT process can increase the prediction accuracy, its burdensome DNN computation deteriorates both latency and throughput, making it hard to be deployed [26], [27]. In this regard, the conventional hardware accelerators such as graphics processing units (GPUs) and field-programmable gate arrays (FPGAs) are not suitable for the AI-enabled HFT process due to either excessive computing latency and limited throughput, or both. Therefore, there is a growing demand to harness domain-specific AI accelerators into the existing HFT systems to overcome the latency and throughput limitations caused by the AI algorithm computation [26]. Although many AI accelerators have been proposed [28]–[30], none of them is developed for the HFT system. The existing systems integrating AI accelerators cannot resolve the system-level challenges associated with the HFT environment since it requires features that should be supported by the hardware architecture design.

In this paper, we propose LightTrader, the product-level AI-based HFT system that utilizes latency-optimized AI accelerators with a delicately tuned system configuration. Based on the AI accelerator architecture and design choice integrating multiple accelerators under power-limited conditions, the DNN processing latency and throughput are drastically improved compared to the existing FPGA-based or GPU-based systems while allowing the seamless integration to the conventional trading pipeline via the dedicated offloading engine. We also explore the best system configuration that optimizes the system-level performance for various HFT environments. LightTrader adopts a new workload scheduling and dynamic voltage frequency scaling (DVFS) scheduling for DNN processing to address the system-level requirements such as handling bursty input traffic with the conservative risk management policy and stringent power constraints of the co-location server.

Our main contributions are summarized as follows.

- We propose LightTrader, the world's first product-level AI-enabled HFT system integrating an FPGA and custom AI accelerators for a short-latency-high-throughput trading system.
- The custom AI accelerator, adopting Coarse-Grained Reconfigurable Array (CGRA), which maximizes the hardware utilization from the flexible dataflow architecture, achieves a throughput of 16 TFLOPS and 64 TOPS.
- We propose a workload scheduling and dynamic power distribution algorithm to find an optimal offloading strategy for AI algorithm computation under bursty market data traffic and limited power condition.
- We build a simulation framework that conducts a back-test using actual historical market data and thoroughly explores the performance of LightTrader for different system configurations.
- The proposed LightTrader system achieves $13.92\times$ and $7.28\times$ speed-up of AI algorithm processing compared to existing GPU-based, FPGA-based systems, respectively. LightTrader with multiple AI accelerators achieves up to 99.5% response rates, while LightTrader with the proposed workload scheduling and DVFS scheduling algorithm relieves the miss rate from 17.1% to 23.1%.

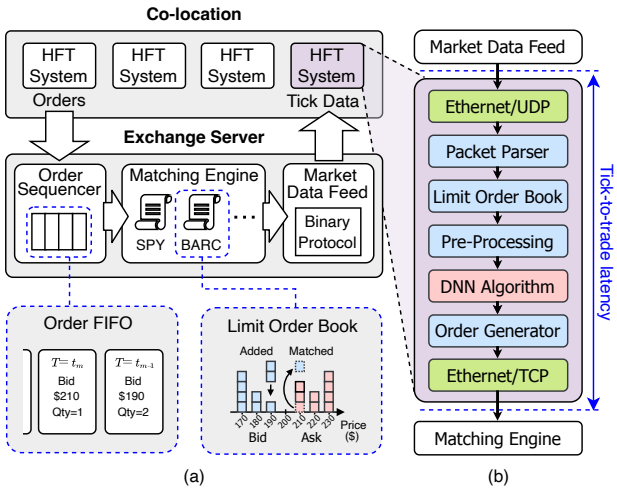

Fig. 2. (a) HFT infrastructure where the HFT systems are installed at the co-location server close to the exchange server for the direct access. (b) An end-to-end AI-enabled HFT process that receives tick data from the market feed, predicts market movement with a DNN algorithm, and transmits orders to the matching engine.

II. Background

A. HFT Process

Figure 2(a) shows the overall infrastructure of HFT, which involves the exchange server and the co-location warehouse where all the HFT systems are located. While various entities are involved in the trading process, the end-point of orders is the matching engine in an exchange server, where bid (buy) side and ask (sell) side orders for each security symbol are registered and matched within limit order books (LOBs). The LOB is the canonical representation of the market condition from which the HFT strategies extract features and determine their actions [31]. The LOB contains the price, quantity, and side information of registered orders for a given security symbol, where the orders are removed whenever a pair of bid and ask orders that match each other. The orders with the same price are aggregated to each price level, where the cheaper asks and more expensive bids are located in the lower

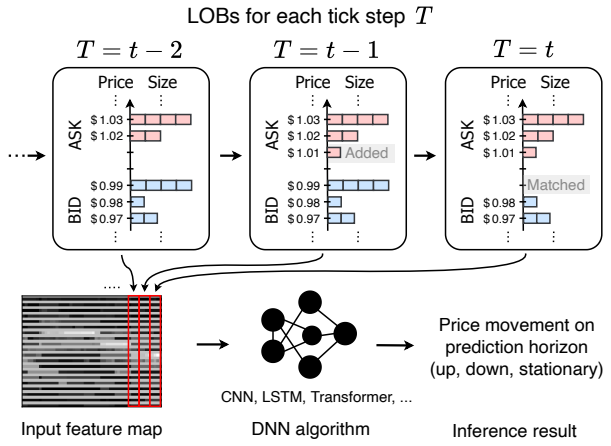

Fig. 3. DNN-based price movement prediction from the input feature map of which the columns consist of parameters of limit order books at each tick step.

level. HFT companies install their system on a co-location venue close to the exchange servers in order to reduce the physical networking latency between the servers [10]. The overall HFT operation flow is as follows. Once HFT systems transmit order packets to an exchange server that specifies the security symbol, price, volume, and side of the order, the orders are sequenced in chronological order and delivered to the matching engine. The matching engine looks up the corresponding LOB for a given symbol, matches the matching pairs, and registers the unmatched incoming orders to the LOB. More precisely, the orders in the LOB are filled with the price/time priority, where orders at the lower price level are filled first, and the earlier orders are filled first for orders at the same level. Note that HFT systems often make 'cancel orders' or 'replace orders' for the existing order in LOB if the market condition has changed and the order is thought stale, which also updates the LOB and generates tick data. HFT systems should track LOB movements and make proper responses, i.e., adding/replacing/canceling orders or doing nothing, from the tick data.

As shown in Figure 2(b), the HFT systems basically perform the trading pipeline from receiving tick data to making an order, called the tick-to-trade. Once the tick data is received from the market data feed, the Ethernet/UDP module first handles the network packet to obtain a payload containing the tick data in a hardware-friendly binary format. Then the packet parser decodes the tick data, and the obtained tick data is passed to the next module to update the local LOB stored in the trading pipeline. The HFT maintains a local LOB which represents a few lowest levels of the global LOB to relieve the storage and management overhead. Next, the HFT system analyzes the LOB data and generates orders at the following modules based on proprietary trading algorithms. In an AI-based trading process, the AI algorithm analyzes the input feature map of the LOB constructed by the pre-processing stage, while the order generator stage utilizes the analysis results to make advantageous orders. Finally, the generated orders are transmitted to the matching engine via TCP/Ethernet protocol.

Generally speaking, a trading pipeline earns more profit if it implements lower latency and more precise trading algorithms. Note that the exchange's co-location server guarantees identical and lowest network latency for all HFT systems; therefore, the tick-to-trade latency is the only factor that an HFT firm can squeeze out from the overall latency. Whereas the conventional tick-to-trade process without the AI algorithm processing takes about one microsecond when implemented on an FPGA [22], [23], [32], the AI algorithm processing can solely take up to several milliseconds without the dedicated accelerator depending on the system design and algorithmic complexity. Also, note that the exchange's co-location server charges an exceptionally high fee for using space, power units, and high-speed connections, so much so that it has become the major inhibitor to realizing profit using an HFT system.

B. AI Algorithms for HFT strategy

Many research works have studied how to use AI algorithms to predict the future movement of financial time series. Most of them focused on forecasting the exact price value or direction of the price movement [33]. In particular, various DNN models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers have been proved to be effective in forecasting short-term price changes. For example, several previous works about high-frequency finance time series forecasting [13], [14], [34] showed even a simple DNN model such as a vanilla CNN and long short-term memory (LSTM) could outperform the conventional forecasting methods. In addition, the advanced model structures that consider detailed characteristics of high-frequency data further improved the forecasting precision compared to the simple DNN models. Zhang et al. [12] presented a DNN model named DeepLOB that combines CNN layers and LSTM layers to handle both the ultra-short-term and short-term features, substantially improving the accuracy of predicting the direction of price change up to 85%. Likewise, Wallbridge [15] combined convolutional layers and transformer blocks to build TransLOB architecture, where it achieved the prediction precision up to 90% owing to the self-attention mechanism that overcomes the noisy high-frequency finance time series.

Note that the tick data is an event-based time-series [35], of which the time interval between tick data samples is highly variable since the LOB is updated irregularly whenever an exchange server receives an order from HFT systems. In addition, the inter-tick time gap is highly variable, from sub-microseconds to a few seconds even for the highly volatile securities, and the event is concentrated within a small fraction of the whole time series [6], [36], [37]. Thus, the above algorithms commonly adopt the tick step-based scheme, where the time step is progressed by the tick event, not by the wall-clock time. Figure 3 shows the process of predicting the price movement adopted by various financial DNN models including DeepLOB [12], TransLOB [15], and a simple CNN [14]. In this process, the model is trained to forecast the price movement for a given tick time, i.e., for the prediction

horizon, using an input feature map constructed from LOBs. More precisely, the LOB snapshots are converted into a two-dimensional input feature map of which each column is derived from the price and volume of the LOB at each tick step. From the input feature map, the DNN model then predicts the direction of price, i.e., up, down, or stationary, at a prediction horizon compared to the current price. As a result, an HFT system can make a beneficial order before the market actually moves in the expected direction.

C. Requirements of AI-enabled HFT system

Note that the above researches on AI algorithms for HFT strategy only focus on the algorithm-level performance, disregarding the challenges of deploying them in an actual environment. This section elaborates on the system-level requirements that must be taken into account to implement a practical AI-enabled HFT system.

Latency, throughput, and response rate The latency and throughput of the AI algorithm computation in Figure 3 are the most significant factors for the profitability of the AI-enabled HFT system. Even if the AI algorithms showcase a high prediction precision for the future market condition, the predicted information is only valid within the prediction horizon, meaning that only limited computation time is allowed for the tick-to-trade process, and it is extremely short in most cases. Also, it is well known that there is a probability that the profit opportunity vanishes even before the prediction horizon ends [10], [26], [38]. These prove that minimizing the tick-to-trade latency by accelerating the time-consuming AI algorithm computation is always preferred in the HFT system.

Throughput is also an important factor in the profit of an HFT strategy. Since HFT firms generally expect only marginal profit from a single opportunity, they are necessitated to find profit opportunities as much as possible and make a large number of orders for the given input data streams [1], [38]. In other words, the more input queries are investigated, the more profits are expected from an HFT strategy, which means the response rate for the input query should be maximized.

Power efficiency Due to the limited space and power supply constraints of the co-location server, HFT firms are eager to increase their performance-per-watt system efficiency of the servers alongside the compact fan-less card design and its low power operation. For the typical co-location-server-target PCIe card, only 75 Watts of the power budget is allocated for the entire card, so the new accelerators (GPUs, FPGAs, or AI accelerators) are designed to consume much less power than the maximum budget in consideration of other system components such as memory modules, high-speed interfaces, etc. This power constraint becomes a major bottleneck in HFT servers and introduces the hazards of power failures in the system, especially when the new AI acceleration feature needs to be deployed for HFT. Therefore, an intelligent power utilization strategy is required to optimize the system performance for different power budgets and workloads.

Bursty tick data traffic and miss rate The tick data shows an extremely bursty traffic pattern because of the chaotic interactions between traders [39], [40], of which the time interval between ticks dynamically varies from a few microseconds to a few seconds even if only a single symbol is subscribed. In addition, it is not uncommon that the burst of data accompanies the abrupt market condition changes, which can be a great change or critical risk for HFT strategies [41]–[46]. For instance, even a small number of orders can trigger a massive number of orders, which again triggers other orders [47]. Note that it was reported that this kind of market disruption occurred more than once a day [41], [43], being more frequent as the trading volume is sharply increasing every year [9].

In this case, the HFT system should allow maximum throughput to manage the risk of missing crucial micro-movements from *missed* tick signals, processing as many input data as possible within a deadline, rather than optimizing the latency of each query [38]. Under the bursty input condition, processing with a small batch size is undesirable to effectively utilize the available resources to process as many as possible. For example, an HFT system may temporally increase the batch size in case of a data burst to process multiple queries in parallel to minimize the number of missed queries. However, if the batch size is too large, the latency becomes excessively long, increasing the probability of violating the deadline. Therefore, the batch size must be actively scaled with a fine granularity.

D. AI-enabled HFT systems

GPU and FPGA have been used to build the GPU-based and FPGA-based systems to accelerate AI-enabled HFT, respectively. The GPU-based system consists of a central processing unit (CPU), network interface controller (NIC) card, and GPUs since the GPUs have no QSFP port for communication with the exchange servers. In the system, the CPU and NIC card process the network packet processing, input pre-processing, and output post-processing, while the GPUs accelerate the core DNN algorithm. However, as GPU's architecture is throughput-oriented, it is not suitable for latency-sensitive HFT operations. Furthermore, most job batch sizes in AI-enabled HFT are set to single due to the short-term market dynamics, so it is hard for GPU to achieve the best throughput performance. Different from the GPU-based systems, the FPGA-based system consists of FPGA boards with on-board QSFP ports. Leveraging the design flexibility, the FPGA boards process the entire pipeline, including the computationally intensive DNN algorithm. The FPGA-based system uses dedicated logic circuits to accelerate the entire pipeline with lower latency than the GPU-based system. However, FPGAs have limited computing resources compared to the demanding computational requirements of AI-enabled HFT, resulting in insufficient throughput.

To overcome the limitations of the existing AI-enabled HFT systems, a dedicated standalone system is necessitated with the consideration of the requirements described in the previous section. Furthermore, since the active research and development of AI-based HFT solutions in the finance industry

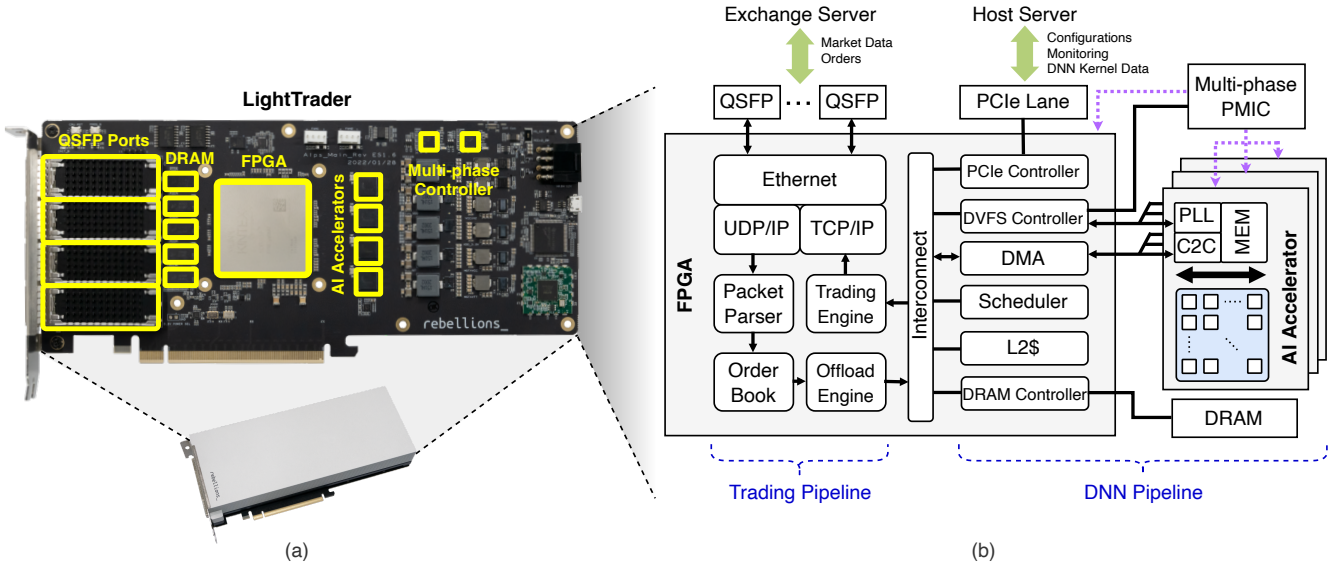

Fig. 4. (a) Photograph of LightTrader prototype (b) Overview of the proposed LightTrader system integrating accelerated DNN pipeline and conventional trading pipeline

have been dedicated to solving practical problems in different markets, customized neural networks are rapidly evolving to aim for different goals. This technical trend opens up a chance to adopt the specialized AI accelerators in HFT that can provide accurate inference and short processing latency via floating point support and flexible dataflow structure, respectively.

III. LightTrader System

We propose LightTrader, the world's first AI-enabled HFT solution that incorporates novel AI accelerator ASICs in an FPGA-based system, to satisfy the system-level requirements of AI-enabled HFT described in Section II-C. Figure 4(b) shows the LightTrader system's overall block diagram consisting of an FPGA and AI accelerators. The AI accelerator is designed to accelerate the DNN operations of HFT because attaining a low latency and high throughput even after incorporating the DNN algorithm in the processing pipeline is the key to successful AI-enabled HFT. Functionally, LightTrader performs the end-to-end AI-enabled HFT that involves both trading and DNN pipeline. We define the trading pipeline as the conventional processing stages of HFT, such as market data acquisition, packet processing, LOB look-up, and order generation, whereas the DNN pipeline refers to the DNN algorithm and its related operations within the AI-enabled HFT. The FPGA plays a role of a central hub of the LightTrader system, being responsible for all the communications around it, including the exchange servers, host server, and AI accelerators. It also performs the entire trading pipeline as well as the data preparation for offloading to the AI accelerators. However, it should be noted that the DNN algorithm executed by the accelerators is the most computationally challenging and thus the bottleneck of the system. In addition, we develop an end-to-end software stack for LightTrader. The in-house deep learning compiler generates instructions optimized for LightTrader with a DNN algorithm targeting HFT, and the runtime driver monitors the LOB and order history of LightTrader.

A. Trading Pipeline

The trading pipeline primarily receives the market data from the direct data feed and pre-processes them to transfer over to the DNN pipeline. Once it receives the processing results from the DNN pipeline, it performs the post-processing to stream the generated orders up to the exchange server.

More specifically, the market data is received through the Ethernet and UDP/IP connection and transferred to the packet parser. The packet parser filters messages of interest and decodes the packet data coded by the market data protocol, such as simple binary encoding (SBE) used in Chicago Mercantile Exchange (CME). The decoded market data is used to update the corresponding LOB and is passed to the offload engine to build the input tensor for the DNN pipeline. The direct memory access (DMA) module is responsible for transferring an input tensor from the offload engine to the AI accelerators. Once the inference is finished from the DNN pipeline, the DMA module transfers the inference result back to the trading engine to make orders. Finally, the trading engine generates the orders and transmits them to the exchange server through the Ethernet and TCP/IP connection.

Offload Engine The offload engine is in charge of processing the LOB data and converting them to normal tensors in order to offload them to the AI accelerators, as depicted in Figure 5. First, the offload engine converts the LOB data formatting with the data type defined in the market data protocol into brain floating-point format (BF16). The LOB data contains ten levels of bid and ask side orders, in which each level contains the price and quantity for the security symbol. Next, it normalizes the LOB data according to the Z-score, which divides the distance from the mean by the standard deviation, in which the mean and standard deviation

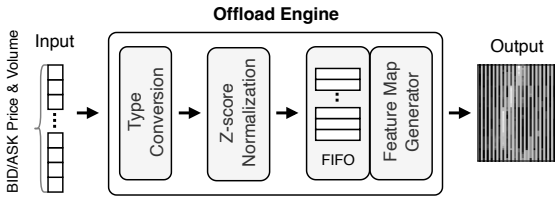

Fig. 5. The micro-architecture of the offload engine

values are obtained from historical market data. After Z-score normalization, one feature vector is generated for the LOB data of a tick. As the DNN models use a two-dimensional (2-D) input feature map made of feature vectors from recent ticks, the offload engine stacks a feature vector into FIFO and generates an input tensor for the DNN pipeline. Once an AI accelerator is ready for inference, the DMA module transfers the input tensor from the offload engine to the AI accelerator. The offload engine manages the stale feature vectors and input tensors, enabling input feature map generation with less storage.

Trading Engine The trading engine conducts the post-processing on the inference output and generates orders from the inference result of the AI accelerator. It allows HFT firms to combine the AI algorithm with the conventional trading algorithms or risk check logics, which are essential for managing the risk of black-box properties of AI algorithms. For example, the AI algorithm predicts the price will go down, and the trading engine immediately makes an ask order to sell the security holdings. The order information is encoded into the order message format as specified by exchange servers. For example, LightTrader supports the FIX message protocol [48] and CME iLink 3 order entry message format [49] by storing the message templates at the on-chip SRAM.

B. DNN Pipeline

The DNN pipeline receives the input tensor from the trading pipeline and processes the tensor with the DNN algorithm to predict the short-term market dynamics by leveraging the computing power of AI accelerators. Although the core computation of the DNN algorithm is performed on the AI accelerators, various interface and control functions are handled on the FPGA side, which includes the host interface, external DRAM interface, accelerator interface, and multi-phase PMIC interface for dynamic voltage and frequency scaling (DVFS). The scheduler is the central module for the control tasks of the DNN pipeline, performing both workload scheduling and DVFS scheduling. Whenever the offload engine notifies that an input tensor for the DNN pipeline is ready by generating an interrupt signal, the scheduler determines the best batch size that can yield the most profit under the given power options. At the same time, it scales the voltage and clock frequency of the AI accelerators to distribute the power efficiently. Meanwhile, the input tensors are loaded to the L2 cache, and the DMA module transfers them to the on-chip memory of each AI accelerator when the batch size is determined by the scheduler. The DNN kernel data are loaded into the L2 cache upon the user request via PCIe, and they are transferred when the AI accelerator is able to receive a new DNN kernel. Once the AI accelerator finishes the inference, the scheduler receives the inference result back and sends it to the trading engine through the DMA module.

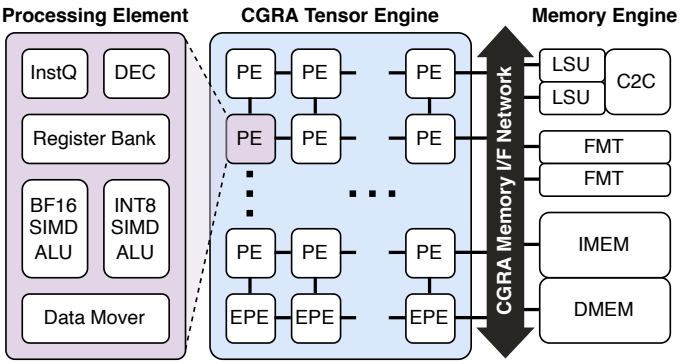

Fig. 6. Illustration of CGRA-based AI accelerator and its PE design

C. AI Accelerator

This section describes the architecture of the proposed AI accelerator featuring latency-aware architectures and logic designs. The ASIC-based AI acceleration is the essential part of the proposed DNN pipeline, enabling latency-optimized execution of different target networks in conjunction with the conventional trading pipelines. The AI accelerator utilizes the Brain floating-point 16 (BF16) execution units as the main computational precision to maintain the original network accuracy across different networks, whereas the lower INT precision, INT8 and INT4, are still supported for the acceleration of the quantized networks for the case that the processing latency is prioritized over the accuracy due to the equations of the profit and loss in the target exchange servers. The BF16 and INT8 configurations achieve 16 TFLOPS and 64 TOPS computation capacity, respectively. Note that the BF16 is especially useful for the DNNs targeting HFT since the networks frequently utilize irregular network structures and non-linear functions that necessitate precision in computations and network parameters.

Overall Architecture The proposed accelerator adopts a Coarse-Grained Reconfigurable Array (CGRA), which benefits from its flexible dataflow architecture, accelerating different types of neural networks available in the HFT system. The accelerator utilizes the hyperblock-level flexibility, similar to classical CGRA cores [50] [51] [52] which can deliver balanced performance (i.e., latency) and Power-Performance-Area (PPA) efficiency in the target applications. On the other hand, the large-scale CGRAs [53] [54] designed to map the entire end-to-end networks on the hardware fail to achieve high energy efficiency. Figure 6 illustrates the high-level diagram of LightTrader's AI accelerator consisting of the two primary components: tensor engine and memory engine.

Tensor Engine The main computational tasks of the AI accelerator are processed in the tensor engine that comprises a 2-D grid of the two types of processing elements (PEs),

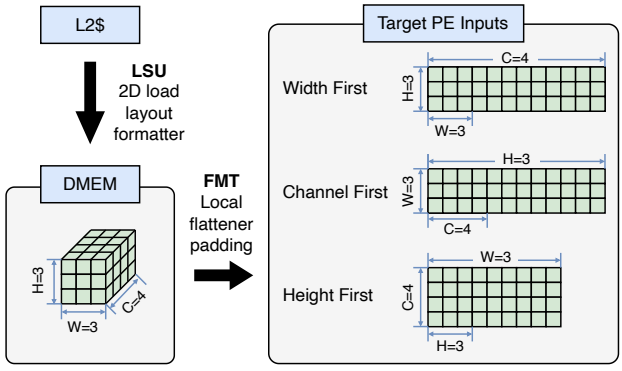

Fig. 7. Flexible data loading to PEs via LSU and FMT

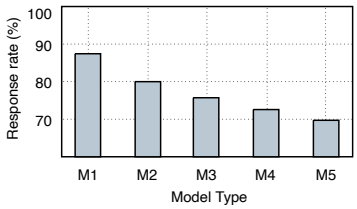

Fig. 8. Response rate for five different models (M1: simplest, M5: most complex)

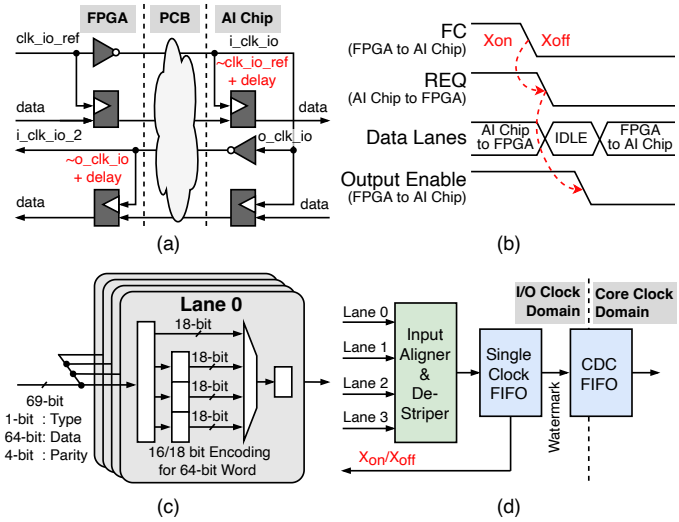

Fig. 9. Latency and bandwidth optimization for C2C interface (a) Source synchronous clocking (b) Out-of-band flow control (c) Data striping (d) Watermark-based flow control logic

the regular PE and the extended PE (EPE). The regular PE contains BF16 SIMD Arithmetic Logic Units (ALUs) and low-precision INT8/4 based SIMD ALUs focusing on low-cost support of Wide Multiplication-And-Accumulation (WMAC), whereas the EPE has extended BF16 functions that supports wider sets of complex arithmetic/logical operations including exponential, logarithmic, and shift. Therefore, EPE can be selectively activated for non-linear functions in inference. Each PE and EPE can run different instruction streams stored in the compact and dedicated instruction queue for the forwarded input data from the neighboring elements and push the computational results to the next target processing elements based on the compiled sequence.

Memory Engine The memory engine consists of two Load Store Units (LSUs), offering latency-hiding off-chip communication via our customized chip-to-chip (C2C) interface, and the data memory (DMEM) and the instruction memory (IMEM) that store the data and program code to allow double buffering between the computation and data transaction. DMEM primarily stores the pre-fetched weight parameters before the inference along with the activation data during the runtime, where the L2 cache can be additionally utilized through the C2C interface in case the data size exceeds the DMEM's capacity.

The data formatter (FMT) is proposed to support prompt data transformation of the streaming data as in lowering, shuffling, and transposing with the programmable controller with specialized RISC-type instructions, the dedicated logic, and the buffers. These layout formattings are implemented with a series of instructions executed over multiple cycles while their partial results are sent to PEs at their earliest possible moment to minimize the latency impact of non-computational operations for inference.

While the data transaction in the tensor engine is limited to the neighbor PEs, the transaction to the memory engine utilizes the memory interface network that provides high bandwidth for multiple traffic sources. Figure 7 illustrates the high-level illustration of transformation operations using LSU and FMT. The multi-level memory modules can operate DNN-optimized layout transformation without incurring any latency overhead to the data loading by employing fine-grained double buffering. At the first loading step, LSU fetches the input tensor block into the DMEM, and then FMT flattens 2-D tensors with respect to the height (H), width (W), or channel (C) dimensions for target kernels. Depending on the data parallelism exploited across the compute models, the output of FMT loads and stores the different data structures to the target PEs, and the instruction streams loaded in the tensor engine run different instruction streams.

Latency Optimization The overriding design goal of the AI accelerator for the HFT system is to minimize the latency of DNN inference operation for all hardware components. Figure 8 shows the response rate of the AI accelerator for five different models (M1: simplest, M5: most complex). As shown in the graph, the inference latency directly impacts the response rate of the input queries. It implies that the network accuracy, which usually increases with the inference latency, should be compromised to avoid the response rate drop that causes the profit decrease. In order to achieve the maximum PE utilization while considering the latency or to increase effective TFLOPS/TOPS for single inference, the AI accelerator utilizes the spatio-temporal parallelism in the hyperblocks identified by the data flow graph (DFG) of the target operations. For the DNNs that HFT maps onto the acceleration hardware, spanning from the small-size CNN networks to the large-size LSTM models, the computational capacity (tensor engine

dimension) and instructions are carefully selected to deliver the optimized latency performance and hardware efficiency. First and foremost, the AI compiler chases sufficient instruction-level parallelism in one hyperblock in the 2-D grid. The thread-level parallelism is often tackled for the fused operations and tensor-level parallel executions as the second target. The nested loops in convolution and matrix multiplication are mapped onto the 2-D grid with minimum consideration of batch-level parallelism to acquire batch-insensitive inference performance.

As depicted in Figure 9, the AI accelerator communicates with the host FPGA via the customized chip-to-chip interface, which utilizes the customized protocol and the DMA controller. For latency optimization, the source synchronous clocking and out-of-band flow control (FC) are adopted to relax the complexity of timing closure while increasing system stability (Figure 9(a) and (b)). The lane stripping is also applied to distribute the internal data to 16-bit width lanes for effective bandwidth scalability (Figure 9(c)). The bi-directional clocks are sent along each set of data lanes so the lane-wise timing matching can ease the PCB-level data communication with higher operating frequencies. For flow control, the additional two bits are directly generated from the interface FIFO watermarks to reduce the processing delay and overhead (Figure 9(d)). This eventually adds up to the off-chip bandwidth increase by $2.4\times$ compared to the Interlaken [55] implementation.

D. Adaptive Workload and DVFS Scheduling

In this section, we describe the proposed performance-per-watt (PPW) based workload and DVFS scheduling algorithms that cope with the dynamically-changing market data under a power budget.

Batching and DVFS In order to develop our own resource control scheduler, we adopt two well-known workloads and power control mechanism: input batching and DVFS. Input batching is a widely used technique in the DNN domain to increase resource utilization by batching the inputs. As the batch processing combines multiple inferences into a single inference, it improves the system throughput while the latency of each request becomes worse. DVFS is an active power management technique that adjusts the voltage and clock frequency at runtime. This technique can be either used to scale up the voltage and clock frequency to improve the performance or used to scale down the voltage and clock frequency to save power. Note that frequent changing in DVFS policy within a short time interval increases the risk of a power failure as well as the overall latency due to the power switching delay. These control hazards make the HFT system carefully uses DVFS for maximization of the scaling benefits.

PPW Metric We observe that both the batch size and DVFS policy affect the AI accelerator's performance and power consumption simultaneously. As the batch size increases, the utilization of the tensor engine increases, which improves the throughput with increased power consumption. Meanwhile, scaling up the voltage and clock frequency reduces the inference latency and also increases the power consumption.

Algorithm 1: Workload Scheduling

```
[Whenever scheduler issues new batch]
if unscheduled input tensor exists in offload engine then
    for dvfs in dvfs_options do
        for bs in batch_options do
            t_total ← t_infer[dvfs][bs] + t_trans[bs]
            if t_total < t_avail and
              power[dvfs][bs] < power_avail then
                Push (dvfs, bs) to candidate_queue
            end
        end
    end
    if candidate_queue is not empty then
        Select candidate with highest PPW metric
    end
    else
        Remove oldest input tensor in offload engine
    end
end
```

Putting the above observations together, we define the performance per watt (PPW) metric calculated as follows.

$$PPW = \frac{batch\_size}{latency * consumed\_power}$$

PPW becomes higher as the batch size increases, the latency per batch decreases, and the power consumption per batch decreases. As the high value of the PPW factor means that the AI accelerator operates computational-/energy-efficiently, we can profile and use this factor when scheduling to distribute the power among the AI accelerators.

PPW on workload scheduling Algorithm 1 shows the workload scheduling algorithm that controls issuing process of a new batch to an available AI accelerator, deciding an optimal DVFS policy of AI accelerator and batch size for the input tensors queued in the offload engine. $dvfs\_options$ is determined by the pre-defined DVFS policy table, and $batch\_options$ is the number of available batch sizes that can be offloaded from the offload engine, respectively. Note that batch and DVFS options affect system throughput, latency, consumed power, and energy efficiency. Both batch options with larger batch sizes and DVFS options with higher clock frequencies and voltages consume more power while improving system throughput. On the other hand, the batch options with larger batch sizes improve energy efficiency and deteriorate the latency of each request, whereas the DVFS options with high clock frequencies and voltages improve the latency of each request and worsen energy efficiency. To find the optimal DVFS policy and batch size with the trade-off between latency and energy efficiency, the scheduler explores all the possible combinations of DVFS policy and batch size and estimates a tick-to-trade of each case. The tick-to-trade, denoted as t_total, is the total latency of the entire DNN pipeline, which is calculated as the summation of the DNN inference latency, t_infer, and DNN result transfer latency, t_trans. If the estimated tick-to-trade, t_total, does not exceed the available time, t_avail, and the estimated power of the selected DVFS policy does not exceed the available power budget, $power\_avail$, the scheduler enqueues the pair of DVFS policy and batch size to $candidate\_queue$. The pair that generates the highest PPW metric score will be selected

Algorithm 2: Power Distribution in DVFS Scheduling

```
[Whenever scheduler distributes available power]
for AI_accelerator not in idle AI_accelerators do
    power_inc ← calc_power_diff(new_dvfs, dvfs, bs)
    if power_inc < power_avail then
        ppw_inc ← calc_ppw_diff(new_dvfs, dvfs, bs)
        Push (new_dvfs, id, ppw_inc) to candidate_queue
    end
end
if candidate_queue is not empty then
    Select candidate with highest ppw_inc
end
```

and committed to the scheduler when there is more than one set of rules in $candidate\_queue$. If $candidate\_queue$ is empty, which means there is no available offloading option that meets the current performance and power constraint, the scheduler defers the workload to the conventional trading pipeline and removes it from the offload engine.

Once the scheduler determines the next target voltage and frequency based on the algorithm, it applies the DVFS policy to the DVFS controller, which physically configures the board's multi-phase PMICs and the AI accelerator's phase-locked loop. On the same note, once the scheduler determines the batch size, the scheduler assigns the DMA to transfer the batch size of the input tensors from the offload engine to the AI accelerator.

PPW on DVFS scheduling The DVFS scheduling consists of steps: saving power and redistributing power. Firstly, the DVFS scheduling algorithm saves power before the scheduler executes the workload scheduling to make room for a new batch issue. The scheduler scales down the voltage and frequency of all AI accelerators until each AI accelerator's inference time does not exceed the available time. Secondly, once the scheduler finishes the workload scheduling, it redistributes the available power budget to the AI accelerators as shown in Algorithm 2. The scheduler estimates the power increase, $power\_inc$, of non-idle AI accelerators for the target scaling-up voltage and frequency. If the $power\_inc$ does not exceed the available power budget $power\_avail$, the scheduler calculates the PPW increase, ppw_inc, with the current DVFS policy, $dvfs$, the new DVFS policy, new_dvfs, and batch size, bs. It enqueues the pair of the new policy, new_dvfs, the AI accelerator's ID, id, and the PPW increase, ppw_inc, to $candidate\_queue$. If one or more ppw_inc exists in $candidate\_queue$, the scheduler selects the candidate with the highest ppw_inc to maximize the sum of the PPW of AI accelerators. The scheduler iterates Algorithm 2 until it can not distribute the available power budget to the AI accelerators. Once the scheduler finishes the power distribution, it assigns the scheduled DVFS policies to the DVFS controller. Algorithm 2 aims to maximize the performance of AI accelerators while fully consuming the constrained power to improve the response rate under bursty market data traffic and limited power conditions.

E. LightTrader Software Stack

Figure 10 shows the overview of the end-to-end HFT software stack, which includes the in-house Deep Learning compiler stack in tandem for tick-level inference. The key components of the software stack include (1) end-to-end HFT software which enables latency-optimized execution of host-engaged function calls invoked from the trading application to the runtime HFT driver; and (2) ML compiler which generates the command streams for the latency-aware network execution of a given neural network graph, managing compute and data transaction tasks in the accelerators. For Runtime operation, the ML driver's commands that capture the position and sequence of the program stream and the weights are passed to the HFT driver, which controls the host-to-accelerator traffic via PCIe interface. The driver is currently implemented with the customized in-house driver and Xilinx PCIe DMA (XDMA).

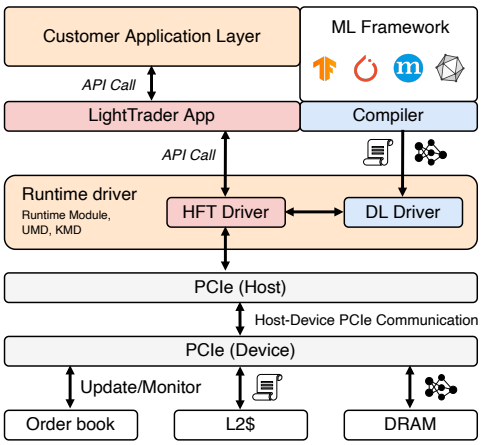

Fig. 10. End-to-End HFT software stack with AI-enabled inference

TABLE I
SINGLE AI ACCELERATOR SPECIFICATION

Process	Package Size	Voltage	Frequency	Power
7 nm	8.7 mm × 8.7 mm	*0.68-1.16 V	*Up to 2.2 GHz	*Up to 10.8 Watts

*These conditions do not imply the optimum operating conditions for energy efficiency (i.e., TFLOPS/Watt)

IV. EVALUATION

A. Evaluation Setup

LightTrader Prototype We build a LightTrader prototype version 1.0 that integrates a Xilinx Ultrascale+ XCKU15P FPGA [56] and AI accelerators for evaluation (see Figure 4(a)). This PCIe card includes a PCIe Gen3 x16 slot for host interface, four QSFP ports for networking, 2GB DDR4 RAM modules, and multi-phase PMICs. The AI accelerator is fabricated in TSMC 7nm FinFET technology to be operated in a clock frequency ranging from 0.8 GHz to 2.2 GHz and consumes up to 10.8 Watts as shown in Table I. This prototype card utilizes four AI accelerators to meet the latency and throughput requirements under the limited power budgets, in which each accelerator is connected to the FPGA via the chip-to-chip interconnect.

Baseline Systems For a fair comparison of LightTrader, we use two popular HFT systems as comparing baselines: a

TABLE II
HFT DNN MODELS FOR EVALUATION BENCHMARK

Baseline Model	Network	Total OPs
Vanilla CNN	CNN	93.0G
TransLOB [15]	CNN+Transformer	203.9G
DeepLOB [12]	CNN+LSTM	515.4G

GPU-based system and an FPGA-based system. As described in Section II-D, the GPU-based system consists of a CPU, NIC card, and GPU, and the FPGA-based system consists of a CPU and an FPGA board. We use the GPU-based system that harnesses an Intel i7-11700 CPU, Xilinx XtremeScale X2522 NIC card [57], and a NVIDIA Tesla V100 GPU, and the FPGA-based system that has an Intel i7-11700 CPU, and a Xilinx Alveo U250 FPGA board [58].

Simulation Framework Because evaluating the HFT systems under real-time stock traffic is difficult, it is imperative to set up a reliable and re-runnable simulation environment. Therefore, we build a simulation framework that can back-test the historical market data, including timestamp and LOB snapshot, which consists of the price and volume of each level on the ask and bid side at each tick. The latency of the HFT system is calculated as the difference between the query input time and the timestamp of the final order message, measuring the tick-to-trade. The simulation framework tracks each input query to see if its tick-to-trade meets the available time and stores the result for the record. We also make the simulation framework have an option of power constraint to emulate the realistic co-location server environment. For faster simulation, we profile the tick-to-trade and power consumption of each system for many LOB data and target DNN models and use them in the simulation framework.

We use the CME's E-mini S&P 500 Futures as market data to validate our system. For DNN algorithms, we choose vanilla CNN, DeepLOB [12], and TransLOB [15] and modify them to predict the future price movement in the market data. Table II summarizes the network type and the total number of required operations of the above DNN models.

B. Non-batching Performance

Methodology Since accelerating the DNN algorithm is the key for AI-enabled HFT, we measure the latency, response rate and effective TFLOPS/W of the baseline GPU-based system, FPGA-based system, and LightTrader when they process input queries immediately without batching (i.e., batch size = 1) to evaluate their intrinsic performance clearly. We set a single accelerator for LightTrader, giving each system a sufficient power condition.

Latency, Response Rate, and Effective TFLOPS/W Figure 11 shows the latency, response rate, and effective TFLOPS/W of the three HFT systems for the target DNN models. LightTrader shows 119 μs, 160 μs, and 296 μs latency for the vanilla CNN, TransLOB, and DeepLOB, respectively, outperforming the GPU-based and FPGA-based system by 13.92× and 7.28× on average, as shown in Figure 11(a).

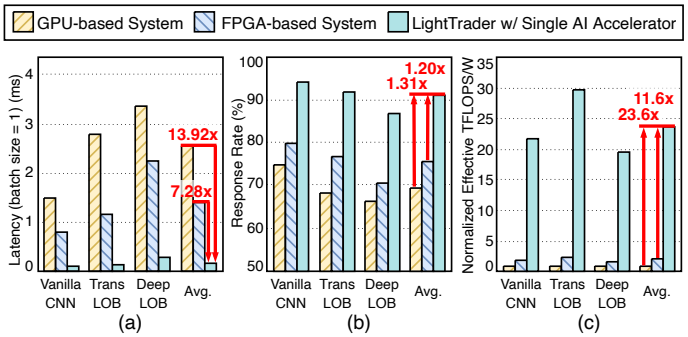

Fig. 11. Non-batching performance (a) Inference latency (b) Response rate (c) Normalized effective TFLOPS/W of LightTrader with single AI accelerator compared to GPU-based, FPGA-based system on various benchmarks (Vanilla CNN, TransLOB, and DeepLOB)

Likewise, the LightTrader system shows a much higher response rate than the other systems, proving that it can process market data in high throughput. In Figure 11(b), the graph shows that LightTrader achieves 94.2%, 91.9%, and 87.1% response rate for the vanilla CNN, TransLOB, and DeepLOB, respectively, outperforming the GPU-based system and FPGA-based system by 1.31× and 1.20× on average. Moreover, the LightTrader system shows much more effective TFLOPS/W than the other systems, even though it consists of the FPGA, peripherals, and only a single AI accelerator. Figure 11(c) depicts that LightTrader achieves 23.6× and 11.6× higher energy efficiency compared to the GPU-based system and FPGA-based system on average, respectively.

Figure 11 shows that the FPGA-based and GPU-based systems are not suitable for AI-enabled HFT processes due to the excessive latency and limited throughput; the FPGA-based system has limited computing resources, and the GPU-based system suffers from low utilization as HFT DNN algorithms are designed as small network considering the trade-off between network accuracy and inference latency. The LightTrader system with a single AI accelerator already outperforms other systems (GPU-based, FPGA-based system), but the LightTrader with a single AI accelerator fails to handle few market data due to its limited throughput; hence the LightTrader system with multiple AI accelerators is required.

C. Performance Scaling with Multiple AI Accelerators

Methodology LightTrader supports multiple AI accelerators with the direct chip-to-chip interface to address the limited throughput issue with a single AI accelerator. We measure the response rate of LightTrader, which is the system-level metric for throughput performance when the number of attached AI accelerators varies. We change the number of AI accelerators from 1 to 16 for the sufficient power condition of 75 Watts and the limited power condition of 40 Watts. Note that the AI accelerators receive the power, except the FPGA and peripherals consume on the LightTrader system.

Without DVFS scheduling, we set the clock frequency and voltage of the AI accelerator conservatively, considering all AI accelerators operate simultaneously. As the number of AI accelerators increases, the available power per AI accelerator

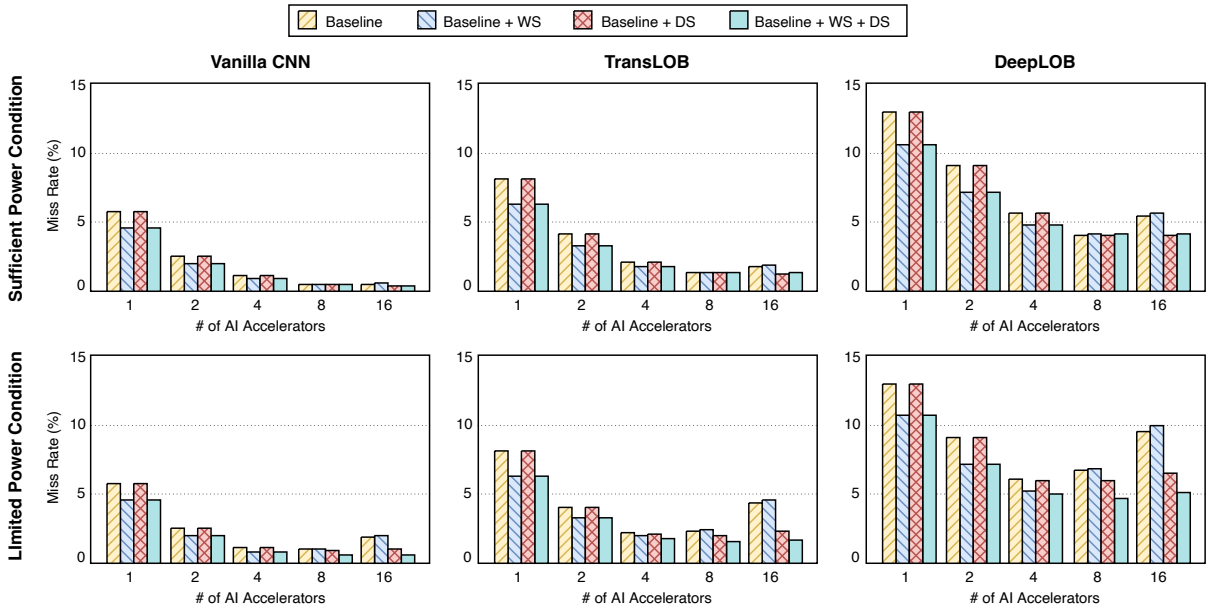

Fig. 13. Miss rate of LightTrader adopting workload scheduling (WS), DVFS scheduling (DS), and both. They are compared to the baseline LightTrader with no resource scheduling schemes when the number of AI accelerators vary from 1 to 16 under the sufficient and limited power condition for various benchmarks (Vanilla CNN, TransLOB, and DeepLOB)

TABLE III
CLOCK FREQUENCY & AVAILABLE POWER CONFIGURATION

Sufficient Power Condition

# of AI Accelerators		1	2	4	8	16
Available Power (W)		55.0	27.5	13.8	6.9	3.4
Frequency (GHz)	Vanilla CNN	2.0	2.0	2.0	2.0	1.9
	TransLOB	2.0	2.0	2.0	2.0	1.7
	DeepLOB	2.0	2.0	2.0	2.0	1.6

Limited Power Condition

# of AI Accelerators		1	2	4	8	16
Available Power (W)		20.0	10.0	5.0	2.5	1.3
Frequency (GHz)	Vanilla CNN	2.0	2.0	2.0	1.6	1.2
	TransLOB	2.0	2.0	1.9	1.5	1.0
	DeepLOB	2.0	2.0	1.9	1.4	1.0

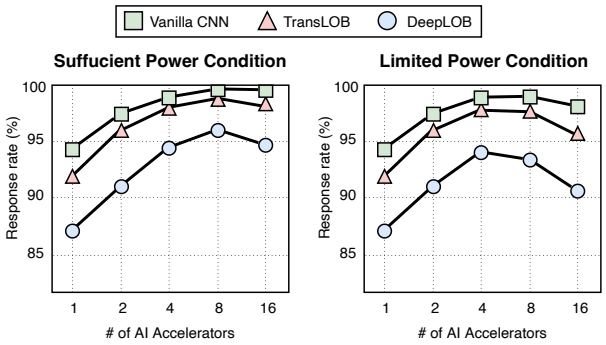

Fig. 12. Response rate of LightTrader when the number of AI accelerators varies from 1 to 16 under the sufficient power condition and limited power condition for various benchmarks (Vanilla CNN, TransLOB, and DeepLOB)

decreases, reducing its clock frequency and voltage accordingly. The clock frequency and available power budget per AI accelerator under the sufficient and limited power conditions are listed in Table III.

Trade-off between Throughput and Power Figure 12 shows the response rate of LightTrader with the various numbers of AI accelerators for the DNN benchmarks under the two power conditions. Under sufficient power conditions, the LightTrader systems with eight AI accelerators achieve 99.5%, 98.7%, and 95.9% response rates for vanilla CNN, TransLOB, and DeepLOB, respectively. Under limited power conditions, the LightTrader system with eight AI accelerators achieves a 98.9% response rate for vanilla CNN, while the LightTrader systems with four AI accelerators achieve 97.8% and 94.0% response rates for TransLOB, and DeepLOB, respectively. The response rate tends to increase with the number of accelerators as the overall system's throughput improves, but the throughput saturates with the increasing number of accelerators in both conditions because the performance of each AI accelerator decreases as the number of AI accelerators increases. After the saturation point, the loss due to the degradation of individual AI accelerators outweighs the performance gains obtained by increasing the number of AI accelerators. It is necessary to improve the throughput of an individual AI accelerator and intelligently distribute power to improve system performance without harming the performance of a single AI accelerator. The trade-off emphasizes the need to improve the throughput of an individual AI accelerator while intelligently distributing the power to improve the overall system performance without harming the performance of a single AI accelerator, which is satisfied by the proposed workload and DVFS scheduling.

D. Putting Everything Together with Workload and DVFS Scheduling

Methodology To ensure that the proposed scheduling algorithms improve the LightTrader system, we evaluate the end-to-end performance of LightTrader when it adopts nothing (baseline), only workload scheduling (WS), only DVFS scheduling (DS) scheduling, and both. We also explore the power conditions (sufficient power and limited power condition), benchmarks (Vanilla CNN, TransLOB, and DeepLOB), and the number of AI accelerators (1, 2, 4, 8, and 16).

Miss Rate Figure 13 shows various miss rate results of LightTrader with four scheduling schemes under the sufficient and limited power condition for three benchmarks. In this comprehensive experiment, we have the following three observations. First, the result shows that workload scheduling is effective in reducing the miss rate especially when the number of AI accelerators is small. This is because the LightTrader system with a small number of AI accelerators tend to fail to handle the bursty tick data traffic due to the limited throughput, and the workload scheduling with batching helps in this situation. However, when the number of AI accelerators of LightTrader is large, the workload scheduling rarely improves the system's miss rate because the throughput of the system is already sufficient. LightTrader shows 21.4%, 18.4%, and 17.6% miss rate reductions, on average, with a small number of AI accelerators (1, 2, and 4) for the vanilla CNN, TransLOB, and DeepLOB, respectively.

Second, we also found that the DVFS scheduling is more effective in reducing the miss rate when the number of AI accelerators of LightTrader is large. As the DVFS scheduling distributes the available power budget to the AI accelerators based on the PPW metric, this scheduling algorithm effectively reduces the miss rate of LightTraders with a large number of AI accelerators. LightTrader shows 19.6%, 23.1%, and 17.1% miss rate reductions, on average, with a large number of AI accelerators (8 and 16) for the vanilla CNN, TransLOB, and DeepLOB, respectively.

Lastly, the experimental results show that the LightTrader systems with both workload and DVFS scheduling algorithms meaningfully reduce the miss rate regardless of the changes in the number of AI accelerators, power conditions, and benchmarks. As a result, LightTrader shows 25.1%, 23.7%, and 20.7% miss rate reductions on average with all the number of AI accelerators (1, 2, 4, 8, and 16) for the vanilla CNN, TransLOB, and DeepLOB, respectively.

V. Related works

HFT Pipeline Accelerators As the tick-to-trade latency of typical HFT systems has shrunk to the microseconds-scale [59], the hardware accelerator has become mainstream due to the order-of-magnitude acceleration performance. In particular, FPGAs have been widely used to accelerate the trading pipeline since it offers the line-rate processing capacity for the tick-to-trade process while retaining flexibility [21]–[23], [32], [60]. They mainly focus on the rapid processing of the network layer and market data, but integrating AI processing with them is almost impossible. LightTrader not only implements the existing trading pipeline but also proposes a solution for AI-based trading logic.

AI for High Frequency Market Data While various research has been made to adopt AI algorithms to predict the market's mid- or long-term movement [61]–[64], the high-frequency market data demonstrate different characteristics from the longer-term market data [36], [37], [41], [44]. For example, the short-term price movement is known to be driven by endogenous factors such as volatility and liquidity rather than exogenous factors such as macroeconomic events or trends, which are utilized by the longer-term prediction methods [59], [65]. Hence, numerous research has focused on financial applications requiring short-term market prediction, such as risk assessment [66], portfolio optimization [67], and order book reconstruction [68]. The LightTrader can be generally utilized for these financial applications, which require prompt AI algorithm processing.

Off-the-shelf AI Accelerators The AI accelerators have been actively researched and commercialized over the past decade in various form-factors for their own target applications; servers [69]–[71]; cards [72]–[74]; modules [28], [30], [75]; and chipsets [76], [77]. However, the existing AI accelerators are not suitable for the HFT since they are not sufficient in handling the following technical challenges of AI integration to HFT simultaneously; (a) small-batch-centric low-latency operation, (b) fine-grained resource utilization control and its hardware support, and (c) high effective (sustained) T(FL)OPS and/or T(FL)OPS/W performance for the customized and irregular neural networks in HFT domain.

VI. Conclusion

The expanding demands for introducing AI algorithms to HFT strategies have necessitated a new HFT system that enables the AI algorithm to meet the latency and throughput requirements. In this paper, we propose LightTrader, which enables the AI-based HFT process by incorporating an FPGA and four CGRA-motivated custom AI accelerators into a board-level system, together with the advanced workload scheduling and power utilization algorithms. Leveraging the high-performance AI processor, the trading pipeline of the proposed system achieves up to $13.92\times$ faster latency, achieving a response rate of 99.5% compared to conventional HFT systems. Targeting the power-limited environment, LightTrader further. The LightTrader further demonstrates advanced features, i.e., workload scaling and PPW-based DVFS, to minimize the risk related to the miss rate under the bursty data traffic, reducing the miss rate by 17.1% and 23.1%.

Acknowledgements

The authors would like to express our sincere appreciation to the contributors of this project in Rebellions; Jaewan Bae, Kyeongryeol Bong, Yoonho Boo, Karim Charfi, Hyo-Eun Kim, Hyun Suk Kim, Byungjae Lee, Jaehwan Lee, Myeongbo Shim, Sungho Shin, and Jeong Seok Woo.

REFERENCES

[1] Securities and Exchange Commission. Concept release on equity market structure, 2010.

[2] Maureen O'hara. High frequency market microstructure. *Journal of financial economics*, 116(2):257–270, 2015.

[3] Jonathan Brogaard, Terrence Hendershott, and Ryan Riordan. High-frequency trading and price discovery. *The Review of Financial Studies*, 27(8):2267–2306, 2014.

[4] Terrence Hendershott, Charles M Jones, and Albert J Menkveld. Does algorithmic trading improve liquidity? *The Journal of finance*, 66(1):1–33, 2011.

[5] Jonathan Brogaard et al. High frequency trading and its impact on market quality. *Northwestern University Kellogg School of Management Working Paper*, 66, 2010.

[6] Julio E Sandubete and Lorenzo Escot. Chaotic signals inside some tick-by-tick financial time series. *Chaos, Solitons & Fractals*, 137:109852, 2020.

[7] Sandrine Jacob Leal and Mauro Napoletano. Market stability vs. market resilience: Regulatory policies experiments in an agent-based model with low-and high-frequency trading. *Journal of Economic Behavior & Organization*, 157:15–41, 2019.

[8] Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.

[9] Jean-Philippe Serbera and Pascal Paumard. The fall of high-frequency trading: A survey of competition and profits. *Research in International Business and Finance*, 36:271–287, 2016.

[10] Matteo Aquilina, Eric Budish, and Peter O'neill. Quantifying the high-frequency trading "arms race". *The Quarterly Journal of Economics*, 137(1):493–564, 2022.

[11] Matthew Baron, Jonathan Brogaard, Björn Hagströmer, and Andrei Kirilenko. Risk and return in high-frequency trading. *Journal of Financial and Quantitative Analysis*, 54(3):993–1024, 2019.

[12] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.

[13] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2511–2515. IEEE, 2017.

[14] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, volume 01, pages 7–12, 2017.

[15] James Wallbridge. Transformers for limit order books. *arXiv preprint arXiv:2003.00130*, 2020.

[16] Andrés Arévalo, Jaime Niño, German Hernández, and Javier Sandoval. High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing*, pages 424–436. Springer, 2016.

[17] Fabien Guilbaud and Huyen Pham. Optimal high-frequency trading with limit and market orders. *Quantitative Finance*, 13(1):79–94, 2013.

[18] Álvaro Cartea, Sebastian Jaimungal, and Jason Ricci. Buy low, sell high: A high frequency trading perspective. *SIAM Journal on Financial Mathematics*, 5(1):415–444, 2014.

[19] Álvaro Cartea and Sebastian Jaimungal. Risk metrics and fine tuning of high-frequency trading strategies. *Mathematical Finance*, 25(3):576–611, 2015.

[20] Thomas Grindsted. Algorithms and the antropocene: Finance, sustainability and the promise and hazards of new financial technologies. 2018.

[21] Christian Leber, Benjamin Geib, and Heiner Litz. High frequency trading acceleration using fpgas. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 317–322. IEEE, 2011.

[22] Andrew Boutros, Brett Grady, Mustafa Abbas, and Paul Chow. Build fast, trade fast: Fpga-based high-frequency trading using high-level synthesis. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–6, 2017.

[23] John W. Lockwood, Adwait Gupte, Nishit Mehta, Michaela Blott, Tom English, and Kees Vissers. A low-latency library in fpga hardware for high-frequency trading (hft). In *2012 IEEE 20th Annual Symposium on High-Performance Interconnects*, pages 9–16, 2012.

[24] Chen Yao and Mao Ye. Why trading speed matters: A tale of queue rationing under price controls. *The Review of Financial Studies*, 31(6):2157–2183, 2018.

[25] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May. Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic. In *Proceedings. Sixth IEEE Symposium on Computers and Communications*, pages 122–129, 2001.

[26] Irene Aldridge and Steven Krawciw. *Real-time risk: What investors should know about FinTech, high-frequency trading, and flash crashes*. John Wiley & Sons, 2017.

[27] Boming Huang, Yuxiang Huan, Li Da Xu, Lirong Zheng, and Zhuo Zou. Automated trading systems statistical and machine learning methods and hardware implementation: a survey. *Enterprise Information Systems*, 13(1):132–144, 2019.

[28] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Ai accelerator survey and trends. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, 2021.

[29] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016.

[30] EETimes and S. Ward-Foxton, "Details of Hailo ai edge accelerator emerge," EETimes, 29-Aug-2019. [Online]. Available: https://www.eetimes.com/details-of-hailo-ai-edge-accelerator-emerge/. [Accessed: 23-Sep-2022].

[31] Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

[32] Heiner Litz, Christian Leber, and Benjamin Geib. Dsl programmable engine for high frequency trading acceleration. In *Proceedings of the Fourth Workshop on High Performance Computational Finance*, WHPCF '11, page 31–38, New York, NY, USA, 2011. Association for Computing Machinery.

[33] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[34] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

[35] Mohammad Sadoghi, Martin Labrecque, Harsh Singh, Warren Shum, and Hans-Arno Jacobsen. Efficient event processing through reconfigurable hardware for algorithmic trading. *Proc. VLDB Endow.*, 3(1–2):1525–1528, sep 2010.

[36] RS Tsay. High-frequency data analysis and market microstructure. *Analysis of Financial Time Series*, pages 231–285, 2010.

[37] Bruce Vanstone and Tobias Hahn. Data characteristics for high-frequency trading systems. In *Handbook of High Frequency Trading*, pages 47–57. Elsevier, 2015.

[38] Credit Suisse. U.s. market structure hft 101 with tradeworx, 2014.

[39] Christian T Brownlees and Giampiero M Gallo. Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational statistics & data analysis*, 51(4):2232–2245, 2006.

[40] Jeffrey R Russell and Robert F Engle. Analysis of high-frequency data. In *Handbook of financial econometrics: tools and techniques*, pages 383–426. Elsevier, 2010.

[41] Neil Johnson and Guannan Zhao. Brave new world: quantifying the new instabilities and risks arising in subsecond algorithmic trading. *Foresight Driver Review DR27. UK Government Office for Science*, 2012.

[42] Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, 72(3):967–998, 2017.

[43] Neil Johnson, Guannan Zhao, Eric Hunsader, Jing Meng, Amith Ravindar, Spencer Carran, and Brian Tivnan. Financial black swans driven by ultrafast machine ecology. *arXiv preprint arXiv:1202.1448*, 2012.

[44] John Cartlidge and Dave Cliff. *Exploring the "robot phase transition" in experimental human-algorithmic markets*. Number DR25 in Foresight Report - The Future of Computer Trading in Financial Markets. UK Government Office for Science, April 2012. Foresight, The Future of Computer Trading in Financial Markets, Driver Review DR25, Crown Copyright 2012.

[45] Jonathan Brogaard, Allen Carrion, Thibaut Moyaert, Ryan Riordan, Andriy Shkilko, and Konstantin Sokolov. High frequency trading and

extreme price movements. *Journal of Financial Economics*, 128(2):253–265, 2018.

[46] Anton Golub, John Keane, and Ser-Huang Poon. High frequency trading and mini flash crashes. *arXiv preprint arXiv:1211.6667*, 2012.

[47] Samir Abrol, Benjamin Chesir, Nikhil Mehta, and Ron Ziegler. High frequency trading and us stock market microstructure: a study of interactions between complexities, risks and strategies residing in us equity market microstructure. *Financial Markets, Institutions & Instruments*, 25(2):107–165, 2016.

[48] "Fix standards," FIX Trading Community, 18-Jun-2021. [Online]. Available: https://www.fixtrading.org/standards/. [Accessed: 23-Sep-2022].

[49] "Ilink 3 binary order entry," iLink 3 Binary Order Entry. [Online]. Available: https://www.cmegroup.com/confluence/display/EPICSANDBOX/iLink+3+Binary+Order+Entry. [Accessed: 23-Sep-2022].

[50] Venkatraman Govindaraju, Chen-Han Ho, and Karthikeyan Sankaralingam. Dynamically specialized datapaths for energy efficient computing. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 503–514, 2011.

[51] Madhu Saravana Sibi Govindan, Doug Burger, and Steve Keckler. Trips: A distributed explicit data graph execution (edge) microprocessor. In *2007 IEEE Hot Chips 19 Symposium (HCS)*, pages 1–13, 2007.

[52] Tony Nowatzki, Vinay Gangadhar, Newsha Ardalani, and Karthikeyan Sankaralingam. Stream-dataflow acceleration. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 416–429, 2017.

[53] Yaqi Zhang, Nathan Zhang, Tian Zhao, Matt Vilim, Muhammad Shahbaz, and Kunle Olukotun. Sara: Scaling a reconfigurable dataflow accelerator. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 1041–1054, 2021.

[54] Seth Copen Goldstein, Herman Schmit, Mihai Budiu, Srihari Cadambi, Matt Moe, and R. Reed Taylor. Piperench: A reconfigurable architecture and compiler. *Computer*, 33(4):70–77, apr 2000.

[55] "Interlaken 150G v1.6 product guide", 04-Oct-2017. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg212-interlaken-150g. [Accessed: 23-Sep-2022].

[56] "Kintex UltraScale+ fpgas," Xilinx. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/kintex-ultrascale-plus.html. [Accessed: 23-Sep-2022].

[57] "X2 series ethernet adapters," Xilinx. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/x2-series.html. [Accessed: 23-Sep-2022].

[58] "Alveo U250 Data Center Accelerator Card," Xilinx. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/alveo/u250.html. [Accessed: 23-Sep-2022].

[59] Albert J Menkveld. The economics of high-frequency trading. *Annual Review of Financial Economics*, 8:1–24, 2016.

[60] Qiu Tang, Majing Su, Lei Jiang, Jiajia Yang, and Xu Bai. A scalable architecture for low-latency market-data processing on fpga. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 597–603, 2016.

[61] Erkam Guresen, Gulgun Kayakutlu, and Tugrul U. Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

[62] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2823–2824, 2015.

[63] Mahdi Pakdaman Naeini, Hamidreza Taremian, and Homa Baradaran Hashemi. Stock market value prediction using neural networks. In *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pages 132–136, 2010.

[64] David M. Q. Nelson, Adriano C. M. Pereira, and Renato A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426, 2017.

[65] Gianluca PM Virgilio. A theory of very short-time price change: security price drivers in times of high-frequency trading. *Financial Innovation*, 8(1):1–34, 2022.

[66] Ye-Sheen Lim and Denise Gorse. Deep probabilistic modelling of price movements for high-frequency trading. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[67] Aiusha Sangadiev, Rodrigo Rivera-Castro, Kirill Stepanov, Andrey Poddubny, Kirill Bubenchikov, Nikita Bekezin, Polina Pilyugina, and Evgeny Burnaev. Deepfolio: Convolutional neural networks for portfolios with limit order book data. *arXiv preprint arXiv:2008.12152*, 2020.

[68] Zijian Shi, Yu Chen, and John Cartlidge. The lob recreation model: Predicting the limit order book from taq history using an ordinary differential equation recurrent neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 548–556, 2021.

[69] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[70] Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, Max Baker, Tom Hawkins, Andrew Bell, John Thompson, Temesghen Kahsai, Garrin Kimmell, Jennifer Hwang, Rebekah Leslie-Hurd, Michael Bye, E. R. Creswick, Matthew Boyd, Mahitha Venigalla, Evan Laforge, Jon Purdy, Purushotham Kamath, Dinesh Maheshwari, Michael Beidler, Geert Rosseel, Omar Ahmad, Gleb Gagarin, Richard Czekalski, Ashay Rane, Sahil Parmar, Jeff Werner, Jim Sproch, Adrian Macias, and Brian Kurtz. Think fast: A tensor streaming processor (tsp) for accelerating deep learning workloads. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ISCA '20, page 145–158. IEEE Press, 2020.

[71] Murali Emani, Venkatram Vishwanath, Corey Adams, Michael E. Papka, Rick Stevens, Laura Florescu, Sumti Jairath, William Liu, Tejas Nama, and ∧Arvind Sujeeth. Accelerating scientific applications with sambanova reconfigurable dataflow architecture. *Computing in Science & Engineering*, 23(2):114–119, 2021.

[72] Linley Gwennap, "Untether Delivers At-Memory AI," The Linley Group Microprocessor Report, 2-Nov-2020. [Online]. Available: https://www.linleygroup.com/mpr/article.php?id=12385. [Accessed: 25-Sep-2022].

[73] Jasmina Vasiljevic, Ljubisa Bajic, Davor Capalija, Stanislav Sokorac, Dragoljub Ignjatovic, Lejla Bajic, Milos Trajkovic, Ivan Hamer, Ivan Matosevic, Aleksandar Cejkov, Utku Aydonat, Tony Zhou, Syed Zohaib Gilani, Armond Paiva, Joseph Chu, Djordje Maksimovic, Stephen Alexander Chin, Zahi Moudallal, Akhmed Rakhmati, Sean Nijjar, Almeet Bhullar, Boris Drazic, Charles Lee, James Sun, Kei-Ming Kwong, James Connolly, Miles Dooley, Hassan Farooq, Joy Yu Ting Chen, Matthew Walker, Keivan Dabiri, Kyle Mabee, Rakesh Shaji Lal, Namal Rajatheva, Renjith Retnamma, Shripad Karodi, Daniel Rosen, Emilio Munoz, Andrew Lewycky, Aleksandar Knezevic, Raymond Kim, Allan Rui, Alexander Drouillard, and David Thompson. Compute substrate for software 2.0. *IEEE Micro*, 41(2):50–55, 2021.

[74] Eitan Medina and Eran Dagan. Habana labs purpose-built ai inference and training processor architectures: Scaling ai training systems using standard ethernet with gaudi processor. *IEEE Micro*, 40(2):17–24, 2020.

[75] "ARA-1 Edge Ai Processor Breakthrough AI performance." [Online]. Available: https://kinara.ai/wp-content/uploads/2022/08/Ara-1_ProductBrief.pdf. [Accessed: 23-Sep-2022].

[76] L Gwennap. Released its first ai processor rk3399pro npu performance up to 2.4tops. *Linley Group, Tech. Rep*, 2020.

[77] Sally Ward-Foxton. Kneron's next-gen edge ai chip gets $40m boost. *eetimes asia*, 2020.