UC San Diego

# Low-latency Ethernet Communications on FPGA SoC for High Frequency Trading

Rudy Osuna, Brandon Reponte & Leeza Gutierrez-Ramirez

CSE 145 | June 3, 2025

# Overview

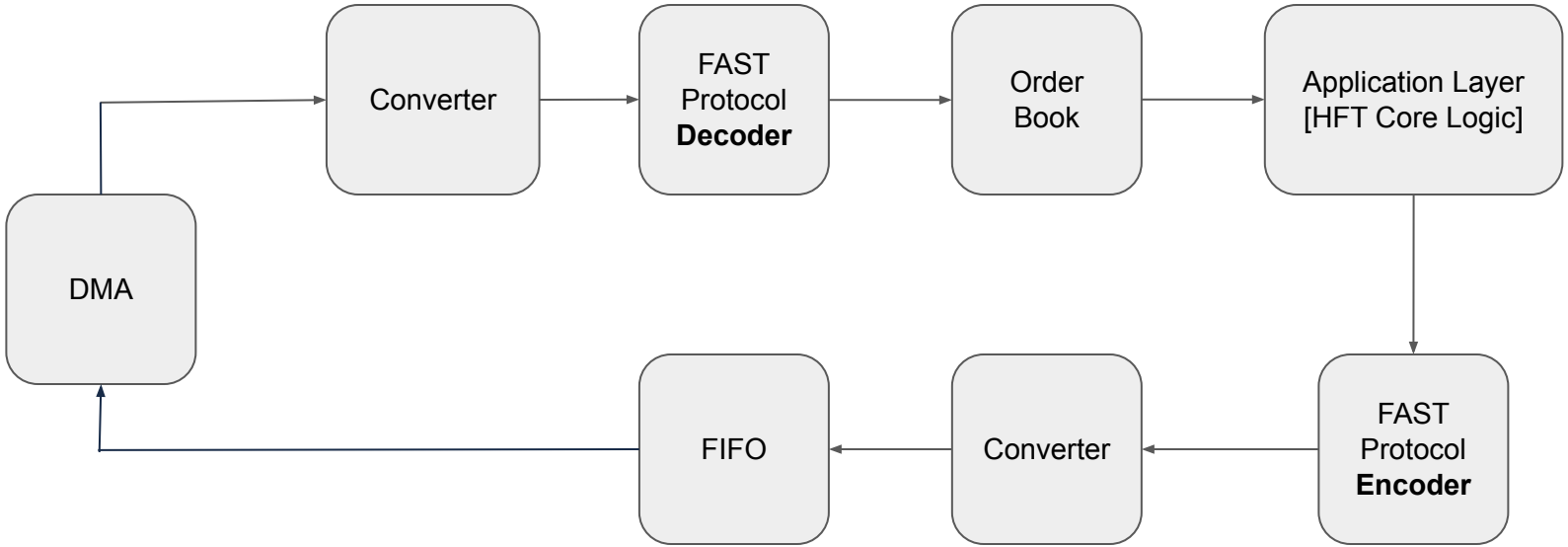
- 90% of trades are made algorithmically (HFT)

**We Created:**

- Baseline solution so people can improve and develop their own strategies
- What we need: ethernet, udp parser, fast encoder/decoder, order book manager, trading logic
- Very niche problem that requires high speed, important to show the architecture
- Tradeoff: smart or fast

References: [1] A. Boutros, B. Grady, and M. Abbas, “A Low-Latency FPGA-based Infrastructure for High Frequency Trading Systems,” Dept. of Electrical and Computer Engineering, University of Toronto, 8 pages. [Online]. Available: https://github.com/mustafabbas/ECE1373_2016_hft_on_fpga

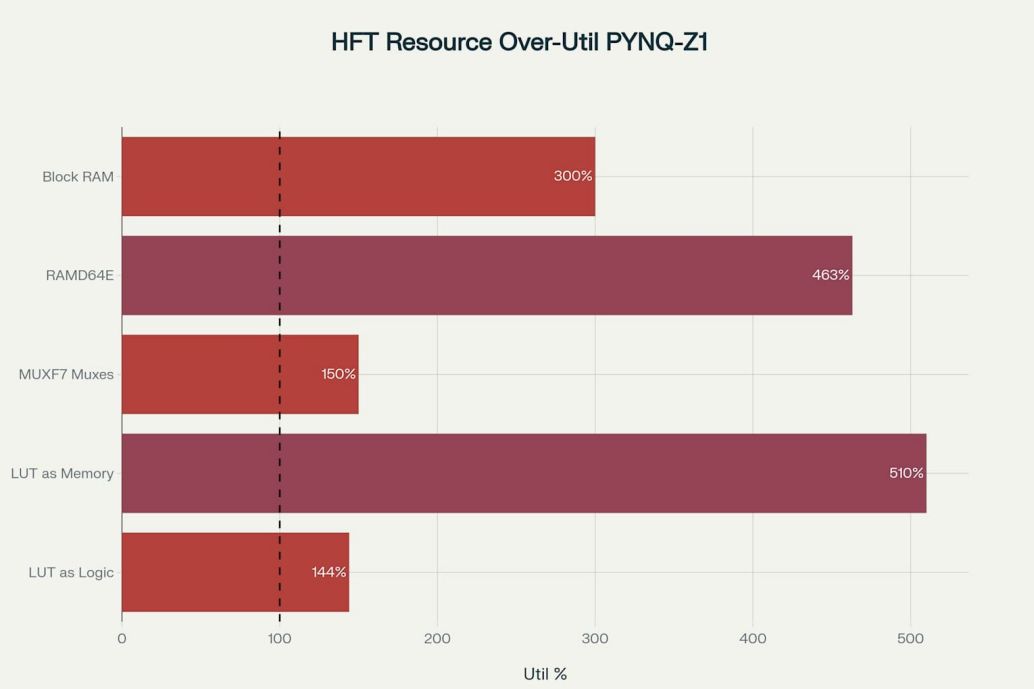
# Solution




References: [1] A. Boutros, B. Grady, and M. Abbas, “A Low-Latency FPGA-based Infrastructure for High Frequency Trading Systems,” Dept. of Electrical and Computer Engineering, University of Toronto, 8 pages. [Online]. Available: https://github.com/mustafabbas/ECE1373_2016_hft_on_fpga

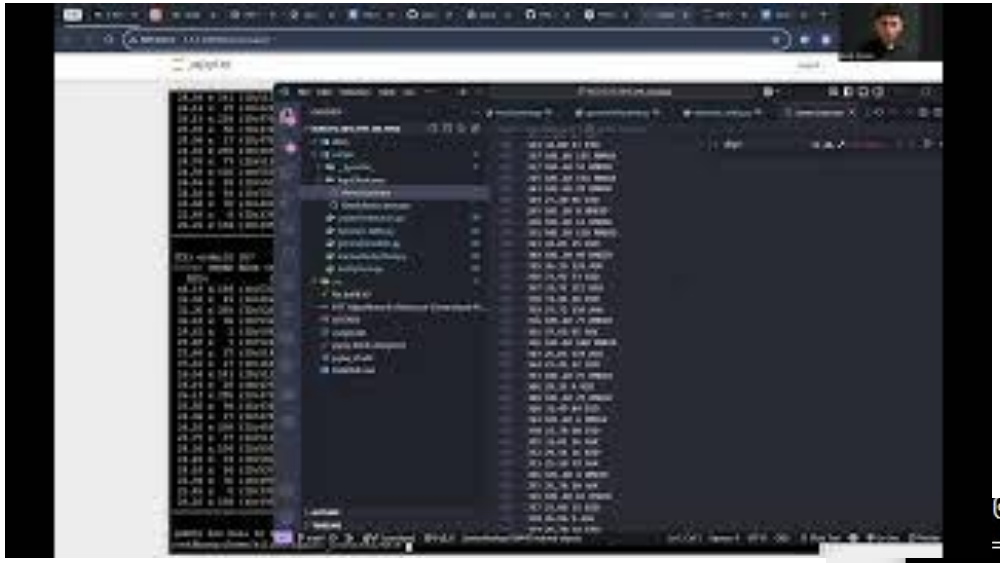
# Optimizing + Downscaling

- **Order Book**:
  - 4096 BID + 4096 ASK -> 16 BID + 16 ASK
- **order_data_converter**:
  - 8→12 TO 4→6 bytes
- **top_bid_converter (+ask):**
  - 12→8 to 6→4 bytes
- **meta_converter**:
  - 12→8 bytes (was 16→12)
- NEW **time_converter_in + time_converter_out:**
  - 32-bit Time signals




References: [1] A. Boutros, B. Grady, and M. Abbas, “A Low-Latency FPGA-based Infrastructure for High Frequency Trading Systems,” Dept. of Electrical and Computer Engineering, University of Toronto, 8 pages. [Online]. Available: https://github.com/mustafabbas/ECE1373_2016_hft_on_fpga

# Solution



```
@pynq:/home/xilinx/jupyter_notebooks/HFT# py
================================================
Total trades matched    : 63
Realized P&L            : $340.20
Note: unmatched buys or sells are ignored in thi
================================================
root@pynq:/home/xilinx/jupyter_notebooks/HFT#
```

References: [1] A. Boutros, B. Grady, and M. Abbas, “A Low-Latency FPGA-based Infrastructure for High Frequency Trading Systems,” Dept. of Electrical and Computer Engineering, University of Toronto, 8 pages. [Online]. Available: https://github.com/mustafabbas/ECE1373_2016_hft_on_fpga

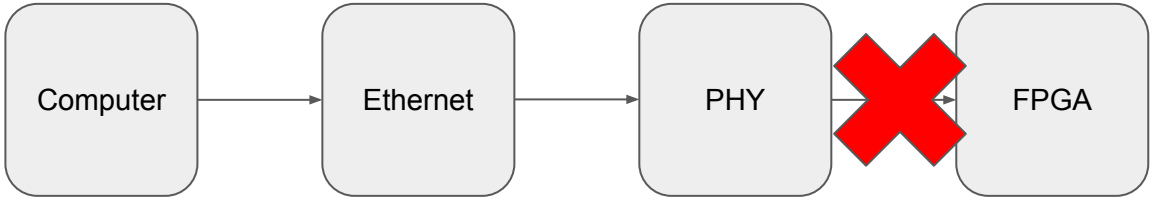
# Outcome?

- **Full complete bitstream programmed onto the PYNQ-Z1**
- 292 lines are new orders (i.e. type BID or ASK), and 252 lines are cancels/removes

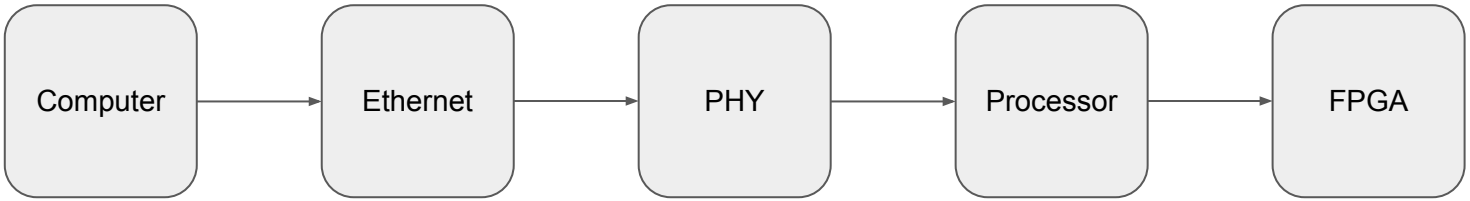
| IP Core | $F_{max}$ (MHz) | Clock Period (ns) |
|---|---|---|
| fast_protocol (fast_hls) | 57.16 | 17.50 |
| order_book (order_book_hls) | 137.19 | 7.29 |
| simple_threshold (threshold_hls) | 137.81 | 7.26 |
| microblaze_to_switch (microblaze_to_switch_hls) | 285.76 | 3.50 |

References: [1] A. Boutros, B. Grady, and M. Abbas, “A Low-Latency FPGA-based Infrastructure for High Frequency Trading Systems,” Dept. of Electrical and Computer Engineering, University of Toronto, 8 pages. [Online]. Available: https://github.com/mustafabbas/ECE1373_2016_hft_on_fpga

# Challenges with Ethernet




PYNQ-Z2 board wires the Ethernet directly to the processing system!

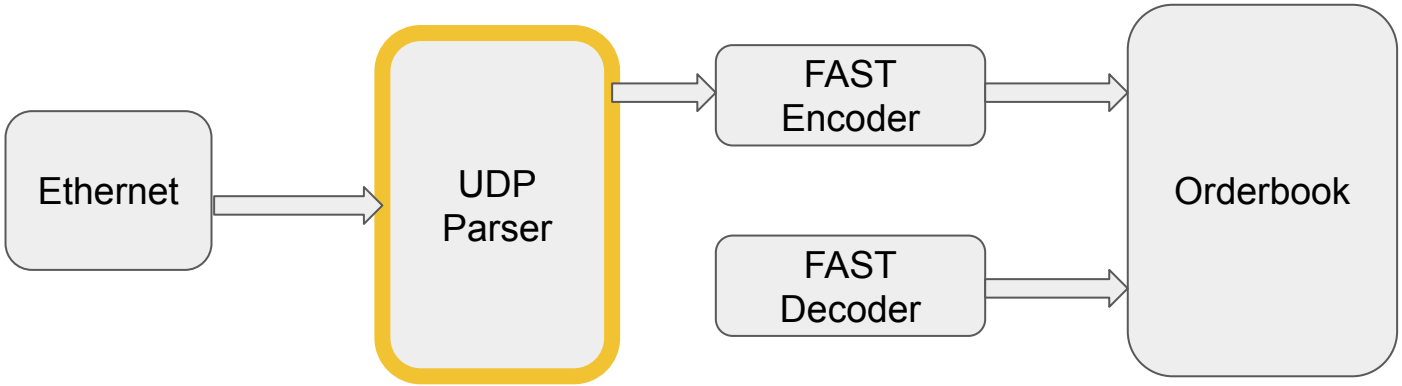



Use direct memory access module to move data quickly between PS and PL

[2] FPGA Ninja, “Taxi: Modular AXI, Ethernet, and UDP hardware stack for FPGAs,” GitHub repository, [Online]. Available: https://github.com/fpganinja/taxi. [Accessed: Jun. 3, 2025].

# Challenges with UDP Parsing

- No existing implementation to reference
- Integrating the UDP packets to FAST decoding pipeline
- Having to get familiar with the quickfast codebase




References: [3] Object Computing, Inc., “QuickFAST: A high performance implementation of the FAST protocol,” GitHub repository, [Online]. Available: https://github.com/objectcomputing/quickfast. [Accessed: Jun. 3, 2025].

# Unfinished Goals

- Complete the Step-by-Step Setup & Use guide
- Do not have a definite way to log round trip latency (without using the PS via Jupyter)
- Backtester to show PnL (profits and losses)

## What's Next?:

1. Optimization Through SystemVerilog Code Instead of Vitis HLS
2. Implement Different Trading Strategy Logic
3. Experiment Upscaling Boards: ZCU104 -> Xilinx Ultrascale+
4. Experiment With Real Data