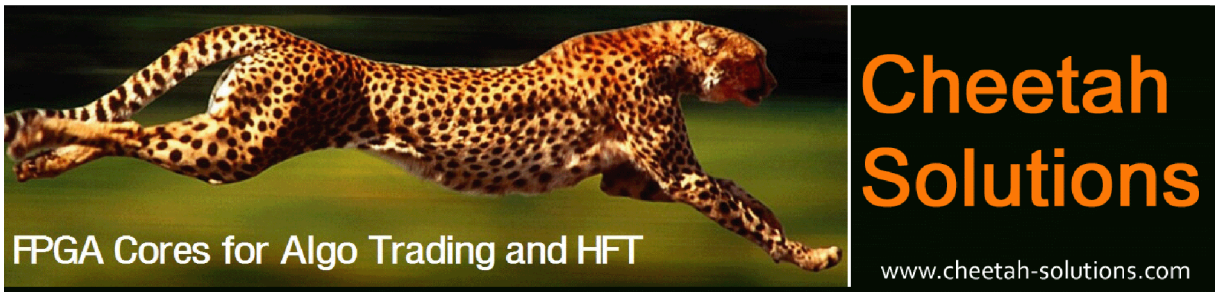


Ultra Low Latency Arbitrage

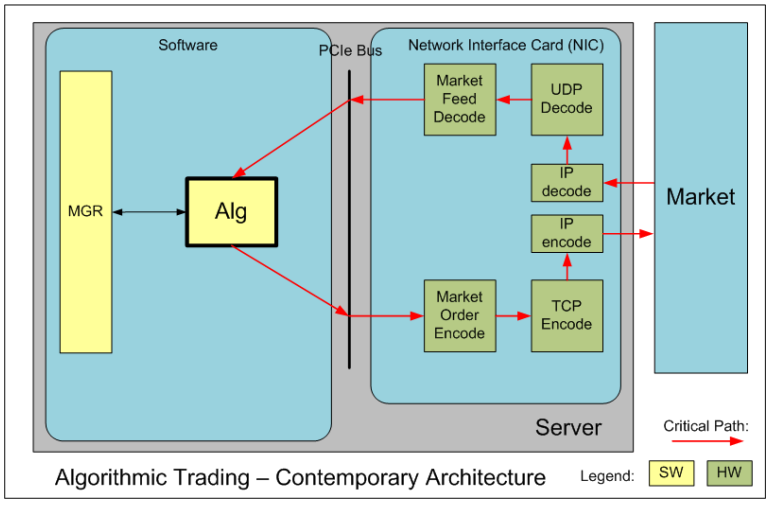
Whitepaper

Introduction

Most algorithmic trading systems based on statistical arbitrage rely on executing the algorithm in a server system usually specially written low latency code so that the algorithm can touch the market in front of other traders.

There have been significant advances, in software architectures for minimising latency caused by CPU cache misses, thread wakeups, and data copying. In addition, there has been a growing trend to push functionality down into the hardware. Typically, the TCP/IP network protocol decoding has been moved into the Network Interface Card (NIC) so that the software doesn't need to spend time decoding the network protocols. Now we are beginning to see full Market Feed protocol decoding in the hardware as shown in the figure below.

However, it is still necessary to feed the market data into the server, have it processed by the software algorithm and then have the software place a market order. Despite all of the CPU and software architecture advances, the server side processing is still the slowest part of the system:



Algorithmic Trading – Contemporary Architecture

The data must be marshaled across the PCIe server bus to the NIC device driver, the application thread has to be ready and waiting, instructions will be executed sequentially by the CPU (or if multithreaded, some delays will occur in synchronising the result set). The resultant market order must be marshaled back over the server bus to the NIC where it can be encoded and transmitted to the exchange. With all this marshaling, threading, drivers, and instruction execution, it is no wonder that the tick-to-trade latency can easily be in the order of 100s of microseconds.

What if you could reduce that tick-to-trade latency to less than one microsecond?

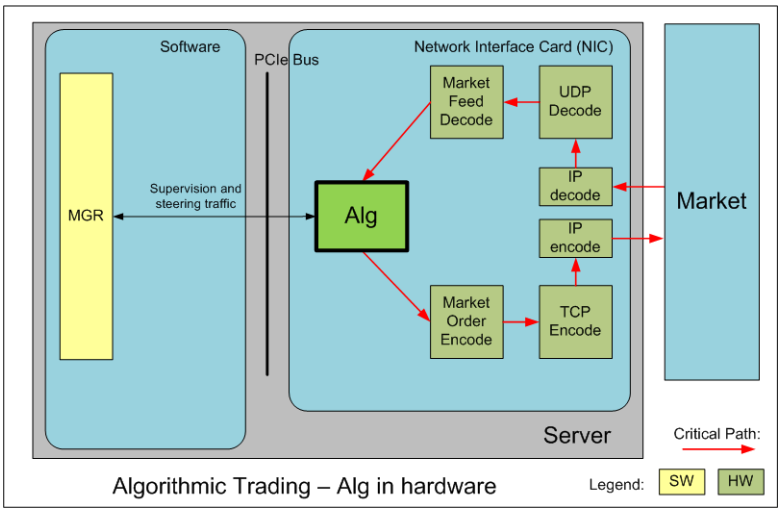
FPGA Solutions

FPGAs are integrated circuits (chips) that provide configurable hardware. Now that hardware vendors have started to offer NICs which contain both network components and user-configurable FPGAs, the door is open to implementing the Alg directly in the hardware, an architecture that offers significant advantages:

- No longer bound to the sequential execution of instructions by a CPU, a hardware designer can design parallel circuitry to implement the Alg to execute much more quickly – if the Alg requires two multiplies, then instead of executing them one at a time, the designer just creates two completely separate multiplier blocks and runs them simultaneously.
- The critical execution path for tick-to-trade no longer needs to cross over the PCIe bus into the server and back out again.

FPGA enhanced NICs make an ideal platform to develop ultra-fast solutions for trading, opening up the possibility to receive market ticks, evaluate an algo computation, and place an outgoing order entirely from within the hardware.

This approach completely eliminates the latency introduced by the server and software and allows tick-to-trade latencies of under one microsecond.



Algorithmic Trading – Alg in hardware

In the diagram to the left, the Alg has been redesigned as an FPGA HW solution and deployed within an FPGA on the NIC. The flow now goes direct from the Market Feed decoder through the Alg and on to the Market Order Encoder (if the Alg decides a trade is required).

The critical flow (red arrows) stays entirely within the HW on the network interface card.

The server software now participates only in the supervision of the Alg's behavior and perhaps to provide some level of control over the alg.

By keeping the tick to trade flow within the hardware, we completely eliminate the latencies introduced by the round trip to the server, including:

- Transmission across the PCIe bus.
- CPU cache invalidation and refresh.
- Waking up of the server thread.
- Sequential execution of the algorithm's instructions.

In addition to this, we can also exploit the parallelism inherent in hardware design such as multiple concurrent floating point operations and parallel data movements.

However, the downside is this:

FPGAs are difficult to program, requiring expert and experienced developers. Even then, making a small change can result in a design that no longer fits or meets timing constraints within the FPGA.

The result is that traditional FPGA solutions take significantly longer to develop than the equivalent algorithm in software and typically require several days if not weeks to make even a small tweak to the algorithm.

The Cheetah-Solution

Providing the best of both worlds …

Cheetah-Solutions brings reconfigurability into the equation providing both high performance ***and*** the ability to change your Alg on the fly.

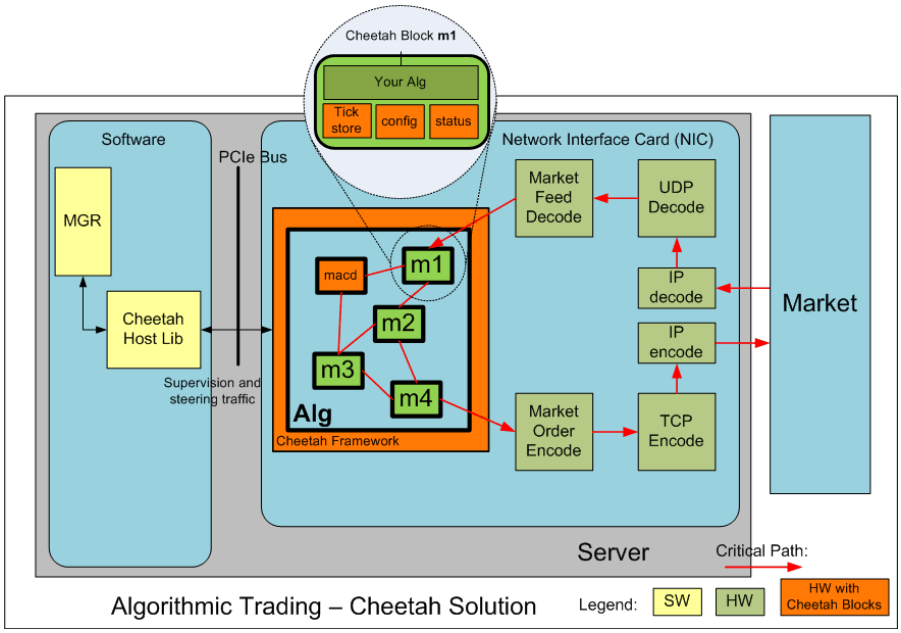
We do this by providing a library of hardware components (Intellectual Property Cores) for you to use in your FPGA design, and corresponding server-side software libraries for managing your Alg.

By modularizing your Alg and using the Cheetah Framework you can:

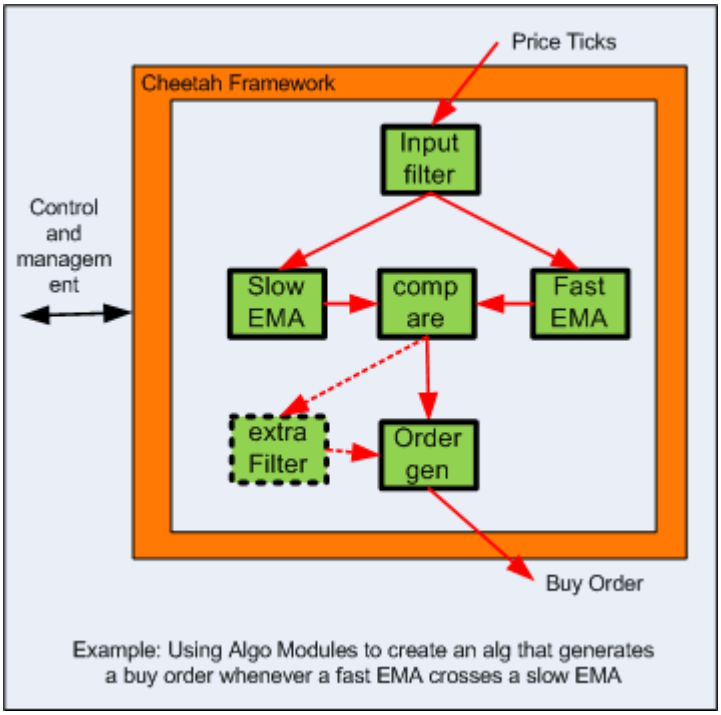
- Change the algorithm by configuring connectivity between the Alg's modules
- Tweak the algorithm by changing parameters inside the Alg's modules
- Manage your Alg by collecting audit data

… without recoding your FPGA.

In this diagram, the algorithm has been divided into a number of modules (m1..m4, and macd). Each of these modules builds upon Cheetah Cub Intellectual Property Cores (shown orange in the **m1** inset) to provide run-time configurability, status, and auditing. The interconnections between the modules is configurable, allowing you to include or exclude various processing steps.



As an example, the diagram on the right shows how some simple modules might be interconnected to produce a trading algorithm that issues a buy order whenever a fast moving average crosses a slow moving average.



Example: Using Algo Modules to create an alg that generates a buy order whenever a fast EMA crosses a slow EMA

Each of these modules can be configured on the fly. For example:

- Configure the Input Filter to respond only to certain instruments.
- Configure the length of the fast and slow EMAs.
- Configure the tolerance in the Comparator.
- Configure the position limit in the Order Gen.
- Introduce or remove a module (such as “extra filter”) from the path.

Cheetah Solutions makes it fast and easy

Cheetah offers framework solutions for support of pipelined processing, giving you support for data and decision rates up to 60 million events per second over up to 255 instruments. Latencies as low as 100-200ns are possible.

Data sharing between modules can be either event driven (eg: price ticks, volume ticks), or by asynchronous lookup which allows one module to publish instrument statistical data and others to access this data.

Next Steps

Visit www.cheetah.solutions.com to read more. Request detailed specification sheets or contact us to discuss project solutions and licensing models.

info@cheetah-solutions.com

The Cheetah Offering

In summary, Cheetah provides the following libraries:

Cheetah Cubs	A library of FPGA Intellectual Property Cores to help you build your Alg modules.	• Tick stores • Moving averages, • Instrument filter	• Hashmap • Period filter • Config & status
Cheetah Framework	A library of Intellectual Property Cores to glue your Alg modules together in reconfigurable ways.	• Remote configuration of your Alg modules • Get status information from your Alg modules • Data audit capability • Runtime configurable interconnect routing • Interconnect for statistical data	
Cheetah Host	Server side library to interface your software components.	• Send configuration information to your Alg • Get status information from your Alg • Get audit data	
Cheetah Blocks	A library of standard ready to go Alg modules.	• EMA • MACD • ATR	• ADX • Arithmetic and Logic

Glossary

EMA	Exponential Moving Average	A trend indicator that smooths and delays a signal such as a stream of price ticks.
FPGA	Field Programmable Gate Array	An electronic chip that can be configured to create arbitrary digital electronic circuits.
MACD	Moving Average Convergence Divergence	A particular type of technical indicator that compares moving averages of different lengths and can be used to predict rising or falling trends.
NIC	Network Interface Controller	An electronic circuit board that plugs into your server and connects the network (wired or fibre) to the server.
PCIe	Peripheral Component Interconnect Express	A standard server bus for interface cards such as a NIC to plug into.
Pipelining	-	The technique of cascading a number of processing stages sequentially (like a manufacturing assembly line) so that each state executes concurrently. This increases throughput.
TCP/ UDP/ IP	Transport Control Protocol / User Datagram Protocol / Internet Protocol	Network protocols commonly used for communications over networks.