Спонсорский доступ с малой задержкой посредством FPGA

Валерий Флоров, Павел Гарин, Павел Смирнов,
Максим Метельков
EXACTPRO
{valery.florov, pavel.garin, pavel.smirnov,
maxim.metelkov}@exactprosystems.com

***Abstract* — В статье рассматривается реализация спонсорского доступа к бирже с помощью FPGA платы с несколькими сетевыми разъемами. Рассмотрен способ снижения задержки прохождения пакетов через такое устройство за счет ранней передачи пакетов, еще не прошедших полную обработку (прием). Описываются использованные инструменты для тестирования разработанной системы, процедура проверки корректности ее функционирования и механизм измерения задержек.**

Keywords—sposored access; trading; fpga; low latency

I. Насущность проблемы

В мире биржевой торговли существует понятие “высокоскоростной трейдинг” (High-frequency trading, HFT). Стратегии, применяемые в HFT, используют преимущество в скорости при доступе к данным торгов (market data, MD) и при формировании заявок на покупку/продажу. Разница в ценах может быть незначительна, но поскольку сделок совершается очень большое количество (сотни сделок в секунду на одного клиента и больше), то в целом прибыль оказывается значительной. В целом позиции открываются на очень короткое время, среднее время удержания 22 секунды, и не переносятся на следующий день (внутридневная торговля). Согласно исследованиям Aite Group рынок HFT составлял 25% от общего объема торгов фьючерсами с ожиданием роста до 40% в 2015 году [2].

Актуальность самой задачи рассмотрена уже неоднократно, например [3] и [4]. Заметим, что наибольшее влияние на результативность сделок при высокоскоростной торговле оказывает значение задержки от момента публикации данных с биржевой площадки, до момента, когда сформированный ордер будет передан в ПО биржи (matching engine, ME). Естественно, большое значение играет и качество самого алгоритма, но оно не является предметом рассмотрения данной статьи.

II. Известные пути решения

Рассмотрим из чего состоят задержки при распространении ордера от клиента до биржи. Цепочка в общем случае может выглядеть так:

- ПО клиента (торговый алгоритм)
- Сетевая карта клиента, (Network interface card, NIC)
- Среда передачи, сюда может входить активное сетевое оборудование, а также физические линии передачи (провода или оптические кабели)
- ПО брокера
- Среда передачи
- Входной шлюз биржи

Кроме длины цепочки, на задержку влияют еще следующие факторы:

- скорость передачи данных. Используются скорости 10-40G
- размер сообщения (ордеров и ответов биржи)
- протокол передачи (обычно TCP или UDP, но могут быть и проприетарные протоколы)

Оценка сетевых задержек на стороне клиента достаточно подробно рассмотрена в [5].

Некоторые пункты в этих двух списках могут быть использованы клиентом для улучшения качества торговли, например: использование более скоростных подключений, написание алгоритма на языках/платформах, которые лучше управляют временем жизни объектов и, как следствие, уменьшают накладные расходы по управлению памятью, делают их более детерминированными и позволяют, таким образом, ускорить выполнение этих алгоритмов по сравнению, например, с языками с автоматической сборкой мусора; использование более производительных сетевых карт с прямым доступом к буферам пакетов, минуя ядро операционной системы (вариации таких сетевых карт можно найти на ресурсе NetFPGA [6] также множество готовых решений предлагает например компания Myricom [7] и другие поставщики); сокращение физической дистанции до брокера, в том числе сокращение количества “прыжков” по активному оборудованию; подключение к тому брокеру, кто вносит меньшие задержки.

Остальные пункты определяются биржей и брокером и не могут быть изменены так просто – протокол передачи, формат сообщения и т.п. Но есть еще одна возможность для существенного сокращения издержек. Некоторые биржи предоставляют непосредственное подключение к своим шлюзам, минуя ПО брокера (sponsored access, SA). Но при этом брокер все равно отвечает за своего клиента. Таким образом, есть некоторое противоречие – с одной стороны брокер и биржа заинтересованы в более активной торговле своих клиентов, но, с другой стороны, брокер сам может понести потери из-за их некорректного поведения.

Таким образом, стоит задача – с одной стороны снизить задержку до минимума, с другой стороны – позволить брокеру своевременно вмешаться и защитить себя (а возможно и клиента) от потерь при неправильном поведении торгового алгоритма клиента.

III. Экстремальный путь решения.

Сейчас можно найти много решений для HFT реализованных на программируемых логических матрицах (Field-Programmable Gate Array, FPGA), т.е. данный подход в настоящее время рассматривается как перспективный, но в основном решения фокусируются на стороне клиента, на таких аспектах, как анализ MD, ускорение принятия решений, ускорение обработки сетевого трафика. Хорошим примером такого обзора является [8].

Также на рынке уже присутствует несколько решений, нацеленных на решение задачи спонсорского доступа с микросекундными задержками: HFT Risk Gateway FPGA based solution [9] – 5 мкс; Ullink FPGA Risk Solution [10] – 2 мкс, но в [1] упоминается (но не рассматривается) подход при котором задержки можно существенно уменьшить. Автор предлагает начинать обрабатывать сообщения, еще до того, как они были приняты полностью.

Обычно в таких устройствах используются те или иные реализации аппаратного стека TCP/IP, например nxTCP for Xilinx [11]. Но это неминуемо вводит задержку на прием и передачу ввиду буферизации пакетов.

Эта задача также может быть решена с применением технологии FPGA. Ускорение достигается заменой последовательной логики, реализуемой процессором общего назначения (central processing unit, CPU) на схемную реализацию – логику, в виде параллельно работающей схемы везде, где это возможно. Такая схема обычно описывается на языке описания схем (hardware description language, HDL) VHDL, Verilog, SystemVerilog или подобном. Компилируется в специальный файл прошивки, который загружается в целевую микросхему FPGA и организует в ней связи для реализации задуманной схемы. Несмотря на то, что тактовая частота для такой схемы обычно меньше чем тактовая частота в 3-4 GHz для топовых моделей процессоров, выигрыш в производительности достигается за счет высокой параллельности (многостадийные конвейеры, при необходимости - множество экземпляров, работающих параллельно) и заточенности под конкретную задачу каждого из блоков. Последним шагом в ускорении таких алгоритмов может быть только реализация их в виде заказных микросхем (application specific integrated circuit, ASIC). Как правило, они имеют более высокую предельную частоту, по сравнению с той же схемой, реализованной в FPGA (за счет исключения всех накладных расходов на организацию таких схем, а также за счет лучшей низкоуровневой оптимизации). Оборотной стороной является высокая стоимость такого решения при малых производимых партиях, а также меньшая гибкость, по сравнению с FPGA.

IV. Цели проекта.

Разработать решение для предоставления спонсорского доступа клиенту со временем задержки менее 200 нс.

Доступ предоставить для Ethernet на скорости 10G по протоколу Native LSE [13], данные MD использовать MITCH LSE [27] Блокировать любое действие клиента, которое может нарушить условия брокера. Рассматривать нарушение следующих условий: Нулевой объем сделки; цена, выходящая за статический диапазон; цена, выходящая за динамический диапазон от цены последней сделки; превышение лимитов торговли как по отдельным инструментам, так и по сумме портфеля в целом. Предоставить возможность брокеру задавать границы проверок для статических и динамических цен, задавать пределы по торговле для каждого клиента,

V. Анализ целей.

Попробуем оценить минимально возможное время задержки. Минимальное время задержки может быть реализовано как быстродействующий “предохранитель”, встроенный непосредственно в линию, связывающую клиента и порт сетевого подключения на площадке биржи. Опираться будем на следующие предпосылки:

1) Клиент и биржа находятся на одной площадке, фактически клиент подключает свой сетевой порт непосредственно к порту входного коммутатора биржи.

2) Используется скоростной протокол нижнего уровня Ethernet 10G, оптика или медь.

3) Используется бинарный протокол Native LSE, а значит устанавливается tcp-сессия.

Оценим время передачи одного сообщения от клиента к бирже, нас интересует главным образом сообщение "New Order". Именно это сообщение дает распоряжение купить или продать определенный актив по указанной цене, основной тип сообщений при HFT. Оценку будем давать для указанной скорости передачи. Размер сообщения — 106 байт. Размер всех заголовков (TCP/IP, Ethernet frame) — 55 байт. Общий размер 161 байт. Итого, при идеальной среде передачи без помех, не учитывая задержку на распространение сигнала в медном/оптическом кабеле, время передачи такого пакета составит ~130 нс (10G) Таким образом, если устройство встраивается в разрыв между клиентом и биржей и использует буферизацию на прием всего пакета, то задержка увеличивается как минимум на двойное время передачи пакета, которое складывается из задержек PHY-MAC, MAC-TCPoffload,

что можно наблюдать как декларируемая задержка для решений [1]. Увеличение размера пакета при передаче сообщений большей длины соответственно увеличивает и задержку.

Возможно ли снизить это время и если возможно, то до каких величин? Да, возможно, если отказаться от буферизации, хотя бы частично. Можно начинать транзитную передачу пакета от клиента, не дожидаясь полного приема. Можно лишь незначительно задержать информацию внутри (на несколько тактов, необходимых на принятие решения — не нарушает ли пересылаемое сообщение некоторых правил торговли). Оценим необходимую задержку в тактах опорной частоты, пересчитав затем их в единицы времени.

Опорную частоту разумно выбрать 156 МГц, как необходимую для тактирования модулей приемо-передатчиков (Small Form-factor Pluggable, SFP). Период одного такта — 6.4 нс. Глубины конвейера в 30 тактов будет достаточно для анализа текущего пакета и принятия решения об разрешении или запрете. Таким образом можно уложиться в поставленный предел по задержке в 200 нс.

Для подключения через Ethernet на скорости 10G можно использовать плату расширения на несколько SFP+ слотов, с которой можно использовать несколько приемо-передатчиков для нужной скорости / среды передачи.

Чтобы не дать возможность клиенту послать некорректный ордер на биржу, нужно иметь возможность его анализировать в реальном времени в процессе передачи, для этого уметь извлекать из ордера такие поля как цена, объем, тип ордера, сторона (покупка/продажа), инструмент. Для бинарного протокола это не представляется сложным.

Основная идея состоит в том, что клиент не должен устанавливать соединение с устройством, оно должно быть абсолютно прозрачно для клиента до тех пор, пока он не нарушает определенных правил торговли. Отсутствие буферизации, связанная с контролем целостности протокола TCP, убирает двойную задержку на прием/отправку каждого транзитного пакета 400 нс + 400 нс [1]. Устройство лишь наблюдает за транзитным потоком, выстраивая одновременно анализ этого потока и принимая решения “на лету”. В случае, если анализ говорит, что передача такого пакета может нарушить правила торговли, а пакет уже начал передаваться на сторону биржи, этот пакет намеренно портится (искажается его контрольная сумма, которая передается последними байтами в ethernet frame, точнее — инвертируется один бит контрольной суммы). Такой пакет будет отвергнут входным коммутатором биржи и не будет анализироваться ME, как не существовавший. Со стороны клиента ситуация будет выглядеть, как внезапный обрыв кабеля. Естественно, необходимо будет предпринять некоторые меры, для минимизации потерь клиента. Для этого должен быть послан определенный сигнал брокеру, отвечающему за этого клиента. Поведение брокера в таком случае может быть оговорено предварительно. Разумным поведением видится отмена всех текущих открытых позиций данного клиента и попытка связаться с клиентом для разъяснения ситуации.

VI. Детали реализации.

Рассмотрим диаграмму, представленную на рис 1.

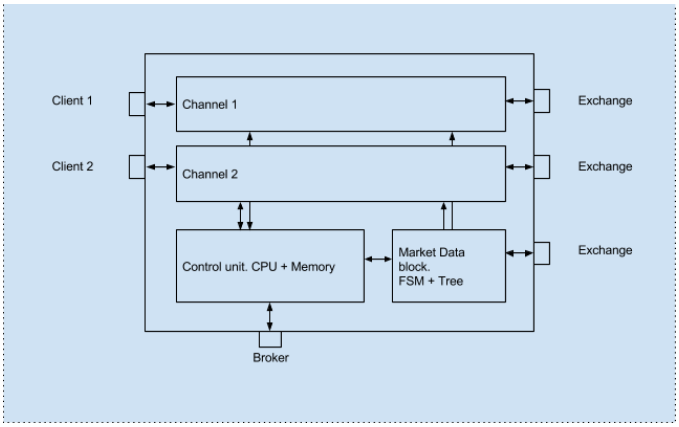

Рис 1. Устройство в целом.

На рисунке обозначены:

1. Блок управления, инициализации и мониторинга.

Представляет собой фирменную реализацию CPU от Xilinx — MicroBlaze [14]. Это 32-разрядный процессор с максимальной тактовой частотой около 200 МГц (для выбранной платы) и гибкой системой команд на основе IP core. Состоит из собственно процессора, блока памяти DDR3, коммутаторов, регистров. Процессор исполняет монолитную программу, которая обслуживает один Ethernet порт, через который представитель брокера может связываться с устройством для настройки и мониторинга. Блок управления реализован как процессор с программной логикой, потому что скорость его работы не критична (это не основной путь прохождения торговых приказов), а логика работы достаточно сложна, чтобы полностью ее реализовывать аппаратно. Он обеспечивает связь брокера с отдельными каналами клиентов, их настройку, а также выдачу сигналов об исключительных ситуациях (нарушениях правил торговли). Один блок управления может обслуживать несколько каналов клиентов, ограничение существует скорее только по производительности процессора. В том числе в задачу этого блока входит ведение журнала событий в системе с записью их на карту памяти (Secure Digital Memory Card, SD).

Блок управления связан со всеми каналами шиной управления, адреса и данных. Каждый канал может выставлять свой собственный сигнал прерывания. Блок управления обрабатывает эти сигналы прерывания и читает регистры соответствующего канала для детализации информации о прерывании.

2. Блок получения актуальных данных о ценах (market data MD).

Также имеет один порт Ethernet, но этот порт подключен уже к бирже для получения актуальных

котировок. Реализован как конечный автомат (finite state mashine, FSM), содержит также хранилище котировок по выбранным инструментам в виде красно-черного дерева, имеет свою собственную память, а также очередь запросов от каналов клиентов на обновление котировок. Посылку сообщений для подписки на прием MD осуществляется с помощью программного блока, поскольку сам процесс подписки не критичен по времени.

3. Несколько пользовательских каналов.

Каждый канал имеет два скоростных порта Ethernet в виде SFP+ модуля, к одному порту подключается клиент, второй порт подключается к коммутатору биржи. Каждый блок имеет связь с блоком управления через шину управления / обмена, а также связь с блоком MD. Подход к разбору сообщений MD рассмотрен в [15] и не представляет особого интереса в данном проекте.

Наиболее интересным является канал, именно он отвечает за обслуживание одного клиента см. рис 2.

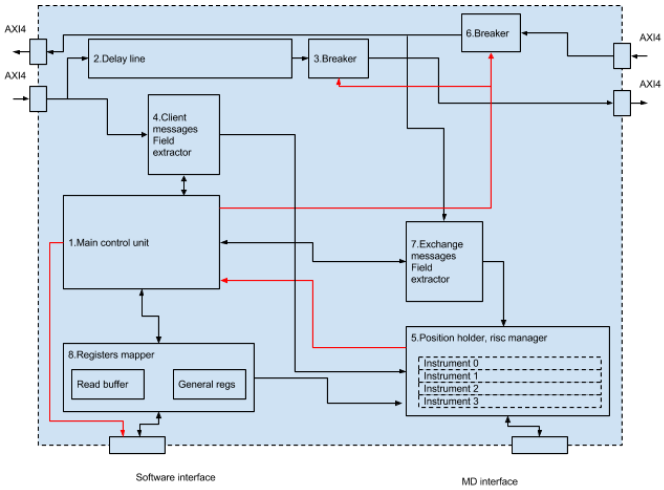

Рис 2. Один клиентский канал

Основные блоки:

1. Основной блок управления канала. (Main control unit) Реализован в виде конечного автомата. Выдает управляющие сигналы на все блоки, принимает решение о возбуждении прерываний, а также о прерывании сессии клиента.

2. Линия задержки (Delay line) 64xN, где N — подбираемый параметр. От него напрямую зависит задержка при передаче ордера. Как видно из диаграммы, задержка вносится только в канал передачи от клиента к бирже, обратный канал такой задержки не имеет. Это связано с тем, что прервать передачу мы можем только в ответ на неправильное поведение клиента. Поведение биржи по определению считается корректным.

3. Прерыватель (Breaker) именно этот модуль прекращает передачу по сигналу от центрального автомата.

4. Экстрактор полей клиентских сообщений (Client messages field extractor). Этот блок отвечает за разбор стека протоколов, выделение “чистого” потока данных. Он же определяет, что передача данных идет только в одной TCP сессии. Определяет, что данное сообщение должно быть разобрано и обработано дальнейшими блоками. Это легко, поскольку тип сообщений в протоколе Native LSE кодируется одним байтом в заголовке. В случае, если в конце приема данного пакета MAC выдаст сигнал, что пакет был принят с ошибкой контрольной суммы, пакет игнорируется, пропускается на сторону биржи, чтобы он также был отвергнут. Поскольку передается "сырой" поток и устройство не вмешивается в него, пока не нарушаются правила.

5. Держатель позиций (Position holder) По поступившей информации от экстрактора полей клиентских сообщений и от экстрактора полей биржевых сообщений подсчитывается размер доступных средств по каждому из настроенных инструментов. Этот блок также запрашивает цену последней сделки у блока обработки MD. Цены кэшируются, поскольку ответ может запаздывать. Если в установленное время блок MD не ответил, используется предыдущая (закэшированная цена по этому инструменту, если она не 0) Также держатель позиций подсчитывает общий доступный объем по всему портфелю и проверяет, что все условия соблюдаются и не выходят за указанные границы. При нарушении — выдает сигнал, по которому блок управления начинает разрыв соединения.

6. Прерыватель обратного потока (Breaker). Срабатывает одновременно с клиентским прерывателем.

7. Экстрактор полей сообщений биржи (Exchange messages field extractor). Необходим для извлечения ответных сообщений со стороны биржи.

8. Блок чтения-записи регистров (Register mapper). Позволяет записывать информацию с шины данных в адресуемые регистры, а также читать при необходимости состояние устройства.

Словари для сообщений от клиента и от биржи отличаются, поскольку в сторону биржи и в сторону клиента курсируют разные наборы сообщений.

На диаграмме обозначены связи между блоками — черным цветом — передача данных и сигналов управления, красным — сигнал прерывания (аппаратного на блоки прерывателей и программного — на интерфейс с программным блоком).

Поскольку канал не может инициировать передачу любых пакетов, как в сторону клиента, так и в сторону биржи, то есть необходимость решить проблему с потерянными пакетами и пакетами, пришедшими в неправильном порядке в пределах одной сессии TCP.

Рассмотрим места, где пакеты могли бы теряться (см. рис. 3).

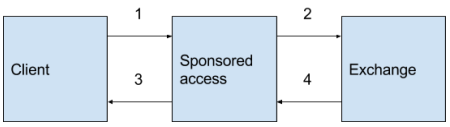

Рис 3. Варианты потерь пакетов при передаче.

1. С пакетом, который потерялся в этом месте все совершенно нормально. Плата этого пакета “не видела”, шлюз — тоже. Потеря будет обнаружена позже, стек TCP/IP со стороны биржи запросит повтор пропущенного пакета. Тогда его “увидит” и это устройство, а после передачи этого пакета дальше — также и биржа.
2. Пакет, потерянный в этом месте уже был обработан устройством, но после запроса со стороны биржи он будет повторен и пройдет через плату дважды. Для этого мы должны обнаруживать повтор такой передачи и игнорировать повторно передаваемый пакет.
3. Здесь ситуация симметрична 1, разве что инициатором передачи становится биржа. Обработка не требуется. Стек TCP/IP клиента должно обнаружить пропуск пакета и запросить повторную передачу, тогда устройство и “увидит” этот пакет.
4. Данная ситуация аналогична 2, действуют те же самые соображения.

В данной реализации устройства сделано одно упрощение — оно может работать только с корректной TCP сессией. Принадлежность Ethernet фрейма именно к текущей сессии определяется по комбинации всех значений: адреса-источника, адреса-приемника, порт-источника, порт-приемника. Эта четверка параметров захватывается при начале TCP сессии клиентом и в дальнейшем, от клиента требуется корректная поддержка установки и прерывания сессии. Фреймы, содержащие пакеты, относящиеся к другим протоколам UDP, ICMP и т.д. пропускаются беспрепятственно. Решение данной проблемы вполне возможно, но потребует расхода аппаратуры на дублирование некоторых блоков, которые обрабатывают входной / выходной поток.

Следующей проблемой, которую пришлось решать при опережающей обработке сырого потока данных это то, что сообщение, из которого извлекаются поля для обработки может оказаться в “битом” пакете. Это означает, что когда приходит сигнал о том, что текущий пакет принят полностью (а одновременно это означает, что его начало уже передается в сторону биржи) все извлеченные данные из этого пакета могут быть неверны. Значит, все поля, которые были извлечены конкретно из этого пакета должны быть сброшены и ожидать, когда данный пакет будет передан повторно.

Когда сообщение полностью разобрано и пришел сигнал подтверждения, что пакет был исправен, извлеченные поля передаются в держатель позиций. Он направляет внутри себя на один из равноправных блоков построителей книжек. Количество таких блоков параметризуется внутри проекта. Каждый из блоков представляет собой конечный автомат, диаграмма его состояний соответствует схеме обработки сообщений [13] (см. рис 4)

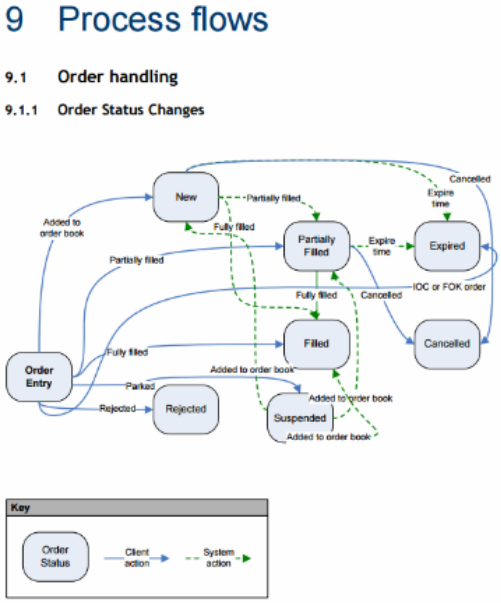

Рис 4. Схема обработки сообщений.

Все сигналы об всех событиях собираются со всех блоков в блок управления. Он, в зависимости от состояния регистра маскирования прерываний, решает — нужно ли запросить прерывание процессора для обслуживания, например для записи в журнал событий. Если же приходит сигнал от блока проверки правил торговли о нарушении какого-то правила, то блок управления вырабатывает управляющий сигнал на прерыватели потока, а также возбуждает прерывание.

Программный блок управления обслуживает сигналы прерываний от всех каналов, собирает по значения всех регистров на момент прерывания и в зависимости от типа прерывания либо просто записывает эту информацию на SD-карту, либо отправляет по резервному каналу сообщение, которое обрабатывается на стороне брокера. Для этого разработана библиотека на языке Java, которая предоставляет API для программной настройки устройства, мониторинга и сигнализации о событиях.

VII. Достигнутые результаты и их анализ.

Проект был реализован на языке SystemVerilog [16] для отладочной платы Xilinx KC-705 [12]. Для подключения четырех дополнительных SFP+ модулей была установлена плата QUAD SFP/SFP+ FMC105 [17], а также в целях отладки установлена Debug Mezzanine Card XM105 [18], на

которую выводились некоторые сигналы для внутрисхемной отладки и замеров задержки прохождения пакетов во время прогона тестов.

Использовались SFP+ модули FTLC8571D3BCL [19] с оптическими патч-кордами и связью с сетевой платой Intel X520-DA2 [20], также с оптическими SFP+ модулями.

Для подключения канала market-data использовался SFP+ слот, уже присутствующий на отладочной плате. Подключение со стороны брокера производилось через 1G ethernet разъем, также присутствующем на плате изначально.

Проект занял примерно 53% ресурсов при реализации каналов для двух клиентов. Утилизация DSP менее 3%. Была задействована внешняя память DDR3 для нужд программного ядра MicroBlaze размером 128 MB.

Для измерения получившейся задержки было выведено два сигнала “Last” от каждого из MAC — входного и выходного. Поскольку длина сообщения не изменяется, как и содержимое сообщения, то разница по времени между этими сигналами и будет искомая величина.

Задержка составила 194 нс с нулевым разбросом, как ожидалось по результатам предварительного моделирования.

Задержка не меняется для разных типов сообщений, поскольку она определяется только глубиной сдвигового регистра плюс небольшие задержки в модулях PHY & MAC. Мы получили стабильную работу при глубине сдвигового регистра N=30.

Окончательное тестирование производилось с помощь специального тестового ПО, разработанного в компании EXACTPRO — Sailfish [21]. Тестовые сценарии работали против тестового окружения площадки MIT [22] Это позволило протестировать корректную работу платы как в “прозрачном” режиме, когда клиент не нарушал контролируемых правил, а также в режиме срабатывания “предохранителя”, когда блок проверки правил обнаруживал такие нарушения.

Поскольку сервер с тестовым окружением недоступен физически, а также активно используется в других проектах, включение сетевых карт на него с оптическими коннекторами не представлялось возможным. Эта проблема была решена таким образом: компьютер, на котором было запущен Sailfish и к сетевой плате которого были подключены и вход и выход устройства SA посредством оптических кабелей, был настроен в режиме "бриджа". Пакеты с одного сетевого порта (к которому был подключен выход канала клиента SA) перенаправлялись на другой сетевой порт, уже подключенный в локальную сеть, соединенную с сервером тестового окружения. Sailfish подключался через тот сетевой порт к которому был подключен вход SA. В результате все пакеты проходили через устройство SA и в прямом и в обратном направлении.

Тестирование устройства производилось как "серого ящика". Т.е. тестовые данные подготавливались с использованием знаний о работе внутренних механизмов устройства. Это позволяет протестировать работу устройства с лучшим покрытием тестами, чем при тестировании его как "черного ящика". Подготавливая тесты для "серого ящика" можно заставить сработать все цепи во всех возможных режимах, обеспечивая максимальное покрытие тестами. Особое внимание уделяется данным, которые разбиты по соседним пакетам, а также обработке сбойных ситуаций (потери пакетов, нарушения очередности доставки, несколько TCP сессий для торговли, нарушений правил RM). Рассматривались ситуации, когда один клиент нарушает правила торговли и это не должно повредить другому клиенту, например. Отдельно рассматривается корректная работа после восстановления обрыва. Т.е. когда RM сработал и прервал соединение клиента, выдал оповещение на стороне брокера, то, после устранения причин нарушений и разрешения брокером соединения этого клиента, соединение может быть установлено снова.

В настоящее время тестирование продолжается. В планах - произвести тестирование устройства под нагрузкой, но для этого необходимо, чтобы соединение осуществлялось непосредственно с сервером, на котором запущена тестовая торговая платформа. Пока с этим есть некоторые проблемы.

VIII. Выводы.

Примененный подход позволяет действительно достигать очень низких значений задержек, значительно ниже, чем при использовании TCP/IP offload, выводя HFT на новый уровень. И, несмотря на то, что примененная стратегия управления рисками (Risk management, RM) выглядит слишком простой, этот проект позволил проверить применимость подхода и показал его расширяемость, в том числе в сторону усложнения RM. Ресурсы современных чипов FPGA позволяют это реализовать полностью. Особенно, если брать семейства, в которых уже включены ядра ARM-совместимых процессоров с повышенной (относительно MicroBlaze) производительностью, например Zinq-7000 [23]

Существенным подспорьем для решения таких задач является появление, особенно в последнее время, библиотек готовых компонентов в виде IP ядер для решения широкого круга задач — от реализаций стеков TCP и/или TCP&UDP до готовых блоков кодирования/декодирования сообщений распространенных протоколов — FIX, FastFIX, ITCH, Native etc. [24][11][25][26]. Кроме проприетарных решений в последнее время появляются и решения с открытым исходным кодом, что позволяет входить на этот рынок не только компаниям, но и отдельным разработчикам.

References

[1] John W. Lockwood, Adwait Gupte, Nishit Mehta, “A Low-Latency Library in FPGA Hardware for High-Frequency Trading (HFT)”, IEEE 20th Annual Symposium on High-Performance Interconnects, 2012

[2] A. Group, “High Frequency Trading in the Futures Markets”, 2010.

[3] Christian Leber, Benjamin Geib, “High Frequency Trading Acceleration using FPGAs”, 21st International Conference on Field Programmable Logic and Applications (FPL 2011), 2011.

[4] J.A. Brogaard, "High Frequency Trading and its Impact on Market Quality," 5th Annual Conference on Empirical Legal Studies, 2010.

[5] S. Larsen and P. Sarangam, "Architectural Breakdown of End-to-End Latency in a TCP/IP Network," International Journal of Parallel Programming, Springer, 2009.

[6] NetFPGA project http://netfpga.org

[7] Myricon products, https://www.myricom.com/products/network-adapters.html

[8] H. Subramoni, F. Petrini, V. Agarwal, and D. Pasetto, "Streaming, lowlatency communication in on-line trading systems," International Symposium on Parallel & Distributed Processing, Workshops (IPDPSW), 2010.

[9] HFT Risk Gateway FPGA based solution, http://www.gatelab.com/products/hft.htm

[10] Ullink, http://www.ullink.com/

[11] nxTCP for Xilinx, http://www.enyx.com/nxtcp-xilinx/

[12] Xilinx Kintex-7 FPGA KC705 Evaluation Kit, http://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html

[13] Native Trading Gateway 10.1 Protocol Specification, London Stock Exchange (LSE), http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit203v101.pdf

[14] MicroBlaze Processor Reference Guide Embedded Development Kit EDK 11.4, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/mb_ref_guide.pdf

[15] G.W. Morris, D.B. Thomas, and W. Luk, "FPGA Accelerated LowLatency Market Data Feed Processing," 2009 17th IEEE Symposium on High Performance Interconnects, Aug. 2009.

[16] IEEE Standard 1800™-2012 SystemVerilog LRM, http://standards.ieee.org/getieee/1800/download/1800-2012.pdf

[17] FMC QUAD SFP/SFP+ for GbE/10GbE FMC105, http://www.vadatech.com/media/FMC105_FMC105-Data%20Sheet.pdf

[18] FMC Debug Mezzanine Card XM105, http://www.xilinx.com/support/documentation/boards_and_kits/ug537.pdf

[19] RoHS-6 Compliant 10Gb/s 850nm Multimode Datacom SFP+ Transceiver, https://www.finisar.com/sites/default/files/downloads/ftlx8571d3bcl_10gbs_850nm_multimode_datacom_sfp-plus_transceiver_product_specification_revd.pdf

[20] Intel® Ethernet Converged Network Adapter X520-DA2, http://ark.intel.com/ru/products/39776/Intel-Ethernet-Converged-Network-Adapter-X520-DA2

[21] EXACTPRO SailFish testing tool, http://www.exactprosystems.com/testtools/sf

[22] MillenniumIT, http://www.millenniumit.com/

[23] Zynq-7000 All Programmable SoC, http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[24] Algo-Logic Systems, http://algo-logic.com/lowlatency

[25] R. Mueller, J. Teubner and G. Alonso, "Streams on wires: a query compiler for FPGAs", Proc. VLDB Endowment, vol. 2, no. 1, 2009

[26] Accelize, http://www.accelize.com/products/fpga-platform-components/finance-ip/market-data-decoders.html

[27] MIT 303 Level 2 — MITCH Specification, London Stock Exchange (LSE), http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit303.pdf