# Цели и задачи исследования

Цель исследования: определить возможность использования проекта hXDP для использования в качестве основы для создания межсетевого экрана. В процессе исследования должны быть получены ответы на следующие вопросы:

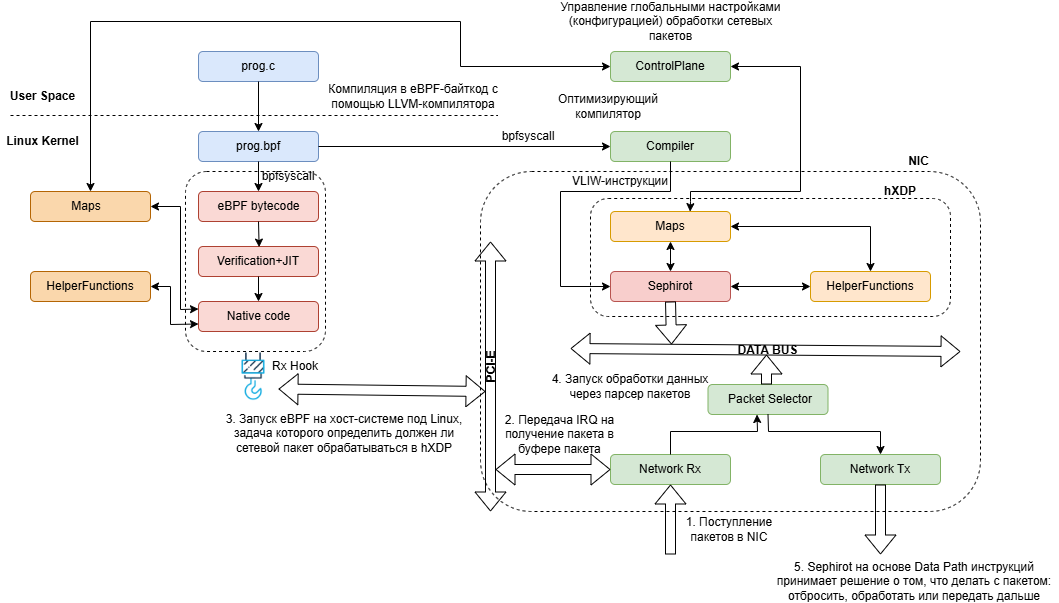
1. Как должен функционировать hXDP в качестве аппаратного ускорителя для обработки сетевых пакетов
2. Что нужно для того, чтобы собрать и запустить проект для работы на произвольном кристалле, в частности, перенос с Virtex-7 (SUME) на Kintex-7 (Base-C)
3. Как проверить работоспособность межсетевого экрана, построенного на базе проекта hXDP

Исходя из цели данного исследования были поставлены следующие задачи:

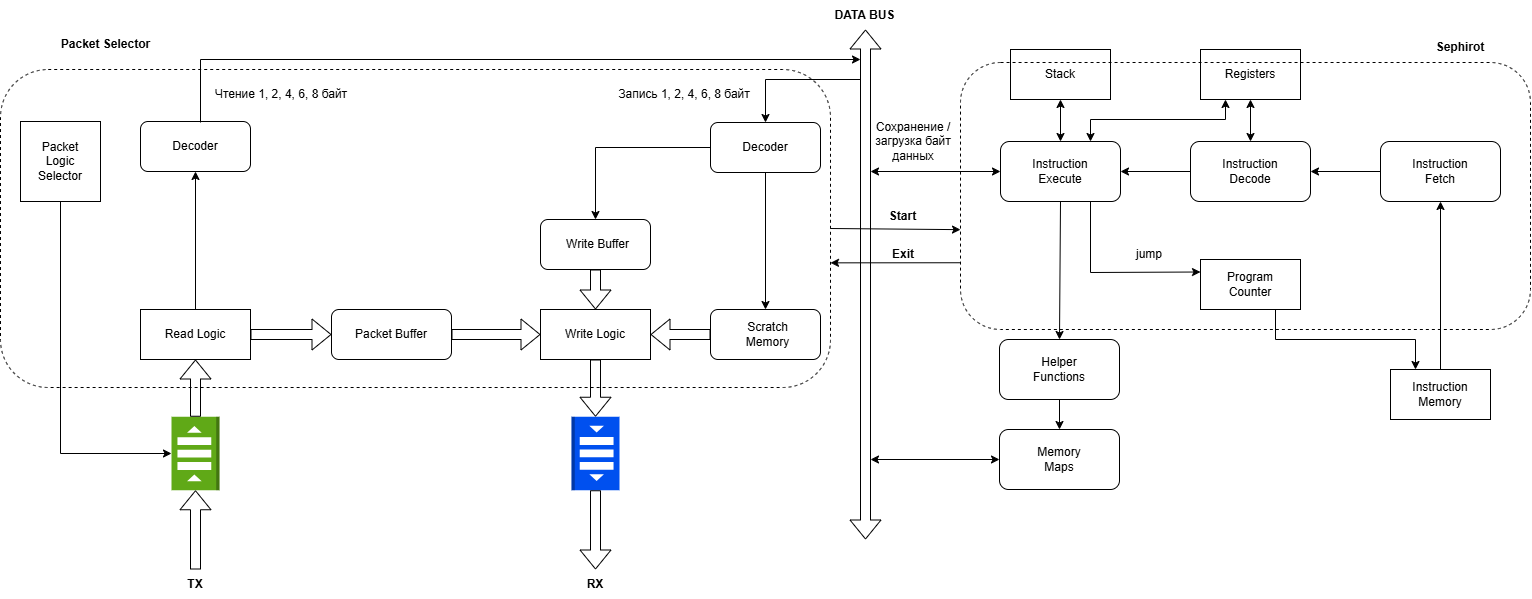
1. Описать функционирование решения hXDP при работе в качестве ускорителя обработки сетевых пакетов
2. Провести портирование проекта на иную аппаратную платформу, а именно, плату Kintex-7 Base-C
3. Разработать методику тестирования решения, построенного на базе hXDP для фильтрации сетевых пакетов

## 1 Функционирование решение hXDP и возможность использования его в качестве основы для реализации межсетевого экрана

hXDP – hardware extended data path, проект аппаратного ускорения обработки сетевых пакетов на основе технологий XDP и eBPF, реализован на плате SUME с ПЛИС Virtex-7. Ключевая особенность XDP – запуск пользовательской обработки сетевых пакетов в режиме ядра операционной системы семейства Linux с использованием виртуальной машины (песочницы) для запуска приложений без написания модулей ядра, а для запуска XDP-програм необходима только лишь среда запуска (виртуальная машина) этого байткода. В ПЛИС Virtex-7 и Kintex-7 (кристалл платы на которую портируется проект hXDP) отсутствует процессорное ядро, на котором можно запустить Linux, но авторами проекта hXDP была разработана eBPF виртуальная машина **Sephirot** с VLIW-архитектурой (несколько eBPF-инструкций объединены в одну длинную инструкцию для параллельного выполнения). Для функционирования hXDP необходима передача данных между сетевым интерфейсом и Sephirot. Упрощенная схема использования проекта hXDP на аппаратной платформе SUME c Virtex-7 для обработки сетевых пакетов приведена на рис.1.

Рис.1. Упрощенная схема обработки сетевых пакетов с использованием hXDP.

Укрупненная структурная схема hXDP, захватывающая селектор выбора сетевых пакетов (Packet Selector) и Sephirot приведена на рис.2

Рис. 2. Внутреннее устройство PacketSelector и Sephirot.

Согласно полученным результатам описания работы можно сделать следующие выводы относительно функционирования проекта hXDP на плате SUME:

1. Подключение платы SUME к PCI-E с наличием сетевого драйвера для данной платы в хост-системе

2. XDP-программа, выполняющаяся на хост-системе, которая могла бы передать обработку выбранных пакетов в hXDP

3. Управление обработкой и конфигурация настроек обработки через интерфейс ControlPlane, в случае с SUME это должны быть регистры, в которые можно осуществлять запись напрямую с хост-системы

4. Для изменения конфигураций обработки сетевых пакетов или же для добавления возможностей по обработке необходимо в лучшем случае потребуется сбросить состояние hXDP, а в худшем — дополнить инструкции Sephirot и новые варианты Data Path обработки.

В значительной степени качество обработки сетевых пакетов и возможность полноценного использования hXDP в качестве основы для разработки межсетевого экрана зависит от того насколько правильно функционирует оптимизирующий компилятор и может ли он корректно может объединить инструкции для параллельного выполнения.

### 1.1 Внутреннее устройство и работа Sephirot

Sephirot — это VLIW процессор, или по-другому виртуальная машина, разработанная для выполнения eBPF-программ с использованием ресурсов FPGA. Эта архитектура предназначена для ускорения вычислений и оптимизации обработки данных за счет параллельного выполнения нескольких инструкций за один такт. Основные особенности Sephirot:

* Sephirot полностью совместим с eBPF (extended Berkeley Packet Filter), что позволяет запускать безопасные и изолированные программы, широко применяемые в системах мониторинга, сетевой фильтрации и трассировки.
* Использование VLIW позволяет исполнять несколько инструкций одновременно, что значительно увеличивает производительность по сравнению с классической RISC- и тем более по сравнению с CISC-архитектурой.
* FPGA обеспечивает гибкость настройки и высокую производительность за счет аппаратной параллельности. Sephirot использует преимущества FPGA для выполнения ресурсоемких операций, таких как маршрутизация пакетов, вычисление хэшей или фильтрация данных в реальном времени.
* Архитектура может быть адаптирована для выполнения специфичных задач за счет модификации конфигурации FPGA и добавления специализированных аппаратных блоков, таких как обработчики сетевых пакетов, модули криптографии и сжатия данных.
* Sephirot позволяет достигать минимальной задержки при обработке потоковых данных благодаря прямому аппаратному выполнению eBPF-программ без необходимости передачи управления традиционным процессорам.

Согласно рис.2 Sephirot состоит из следующих модулей:

1. Блок декодирования инструкций (Instruction Decoder)

Назначение: Интерпретирует входящие eBPF инструкции и преобразует их в сигналы управления для других модулей.

Особенности:

* поддержка набора инструкций eBPF, включая арифметические, логические, условные и переходы.
* распределение инструкций между функциональными блоками VLIW-архитектуры.
* возможность декодирования нескольких инструкций за такт (зависит от ширины VLIW).

2. Функциональные блоки (Execution Units)

Назначение: Выполнение инструкций eBPF, состав этого модуля:

* арифметико-логический блок (ALU для выполнения базовых арифметических операций, таких как сложение, умножение, а также логические операции.
* блок работы с памятью (Memory Access Unit), с поддержка операций загрузки и сохранения данных.
* блок сравнения (Comparison Unit) с реализацией инструкций условных переходов и ветвлений.
* блок криптографии (опционально): Акселерация операций шифрования или хеширования (например, SHA или CRC).
* VLIW: Позволяет параллельно исполнять несколько инструкций с минимальными зависимостями.

3. Регистр файлов (Register File)

Назначение: Хранение значений промежуточных данных и переменных.

Особенности:

* 10 регистров, определенных стандартом eBPF (R0-R9), плюс дополнительные временные регистры для аппаратных оптимизаций.
* высокоскоростной доступ для минимизации задержек.

4. Блок управления потоком (Control Flow Unit)

Назначение: Управляет выполнением программы, включая обработку условных переходов и вызовов функций.

Особенности:

* реализация стека для функций и поддержки возвратов.
* ускоренная обработка прямых переходов и циклов.

5. Память инструкций и данных (Instruction and Data Memory)

Назначение: Хранение кода программы (инструкций eBPF) и данных для обработки.

Особенности:

* инструкции хранятся в памяти, совместимой с FPGA BRAM.
* данные могут быть организованы в виде кольцевого буфера для потоковой обработки.

6. Интерфейс ввода/вывода (I/O Interface)

Назначение: Обеспечивает связь между Sephirot и внешними компонентами.

Особенности:

* поддержка интерфейсов AXI4 или Avalon для взаимодействия с другими модулями FPGA.
* высокоскоростные каналы для передачи данных и управления.

7. Контроллер VLIW (VLIW Controller)

Назначение: Синхронизирует выполнение нескольких инструкций, обеспечивая параллельное выполнение без конфликтов ресурсов.

Особенности:

* распределяет инструкции между функциональными блоками.
* управляет зависимостями между операциями.

8. Модуль отладки и мониторинга (Debug and Monitoring Unit)

Назначение: Обеспечивает диагностику работы виртуальной машины.

Особенности:

* регистры состояния для отслеживания выполнения инструкций.
* поддержка внешних инструментов отладки, таких как JTAG.

9. Модуль оптимизации выполнения (Execution Optimization Unit, опционально)

Назначение: Оптимизация выполнения eBPF программ.

Особенности:

* рискованное выполнение инструкций.
* адаптивное управление энергопотреблением для экономии ресурсов FPGA.

10. Блок памяти контекста (Context Memory Unit)

Назначение: Хранение параметров eBPF-программы и метаданных.

Особенности: прямой доступ к параметрам eBPF-контекста (например, сетевые пакеты или параметры вызова).

## 2 Портирование hXDP на Kintex Base-C

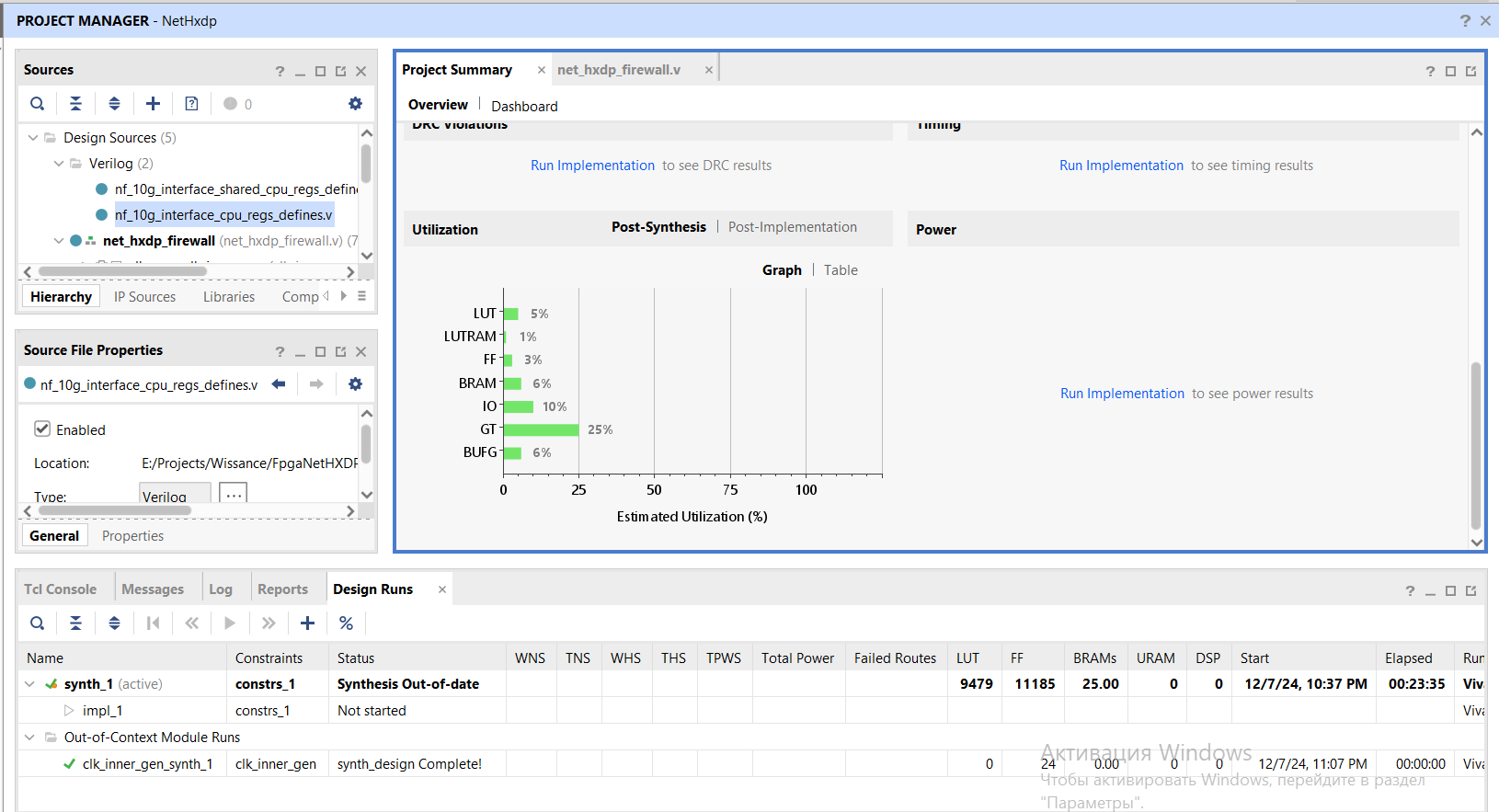
Для того, чтобы проверить работоспособность решения hXDP необходимо в первую очередь провести портирование проекта с аппаратной платформы SUME Virtex-7 на Base-C Kintex-7. Следует отметить, что все материалы по работе с проектом: <https://github.com/Wissance/FpgaNetHXDP> . Имеющийся в открытом доступе проект hXDP20 не может быть скомпилирован простым переключением кристалла в проекте Vivado, поэтому для синтеза данного проекта необходимо повторно создавать данный проект с нуля, при этом в процессе портирования нужно учитывать физические отличия в платах, а именно:

* у Base-C 2 QSFP порта в отличие от 4 у SUME
* у QSFP-разъемов на Base-C платах отсутствует подключение пина tx\_fault
* у Base-C иная схема тактирования (50 МГц) чем у SUME (200 МГц )
* PCI-E у Base-C x4, а у SUME - x16

Данные ограничения были преодолены при переносе проекта на Base-C. Также при синтезе данного проекта было сделано следующее:

* проведен upgrade значительно числа IP-ядер
* IP-ядра nf\_10g\_interface и большое количество их зависимости не удалось перенести в проект как IP-ядра, они были добавлены в качестве модулей

Как результат этого процесса, все IP-ядра и модули, которые были напрямую связаны с модулем верхнего уровня top.v были синтезированы, результат синтеза приведен на рис.3.

Рис. 3. Результат сборки портированного модуля верхнего уровня и всех явных зависимостей под Kintex-7 Base-C.

Однако несмотря на то, что часть проекта, которая была явным образом связана с модулем верхнего уровня, в проекте не были на сегодняшний день портированы сабмодули блочного дизайна в виду того, что результат использования разных методик не дал никакого результата эти модули необходимо портировать аналогично зависимостям модуля верхнего уровня, т. е. создавать достаточно весомую часть вручную. Основная проблема связана с ядром pcie, использующееся в проекте, предназначено именно для кристалла Virtex-7, и необходимо будет произвести замену на ядро для Kintex с соответствующими настройками и пере-подключениями. Использование фрэймворка и IP-ядра RIFFA (используется в оригинальном hXDP) для Kintex-7 остается вопросом отдельного изучения, но если RIFFA не подойдет, то нужно будет искать альтернативные пути имплементации PCI-E интерфейса в проект. В процессе портирования также был сконфигурирован файл временных ограничений для проверки наличия возможных ограничений для обработки пакетов на интерфейсах с пропускной способностью 1Гбит/с с учетом того, что работа всего решения происходит в нескольких частотных доменах, файл временных ограничений также находится в git-репозитории проекта. Ограничения на тактовые частоты определены следующим образом:

# 50 MHz Clock - FPGA.

create\_clock -period 20.000 -name fpga\_cry\_clk [get\_ports fpga\_cry\_clk]

# 156.25 MHz Clock - SFP.

create\_clock -period 6.400 -name xphy\_refclk\_p [get\_ports xphy\_refclk\_p]

Также, что важно отметить, был адаптирован testbench аппаратной части проекта, согласно данным тестам удается наблюдать наличие всех необходимых сигналов (см. рис.4), однако, в виду того, что сабмодули не были портированы, то результат работы тестбенча — пустой файл, если единственная причина это сабмодули, то полное портирование проекта приведет к тому, что тест будет полностью работоспособным.

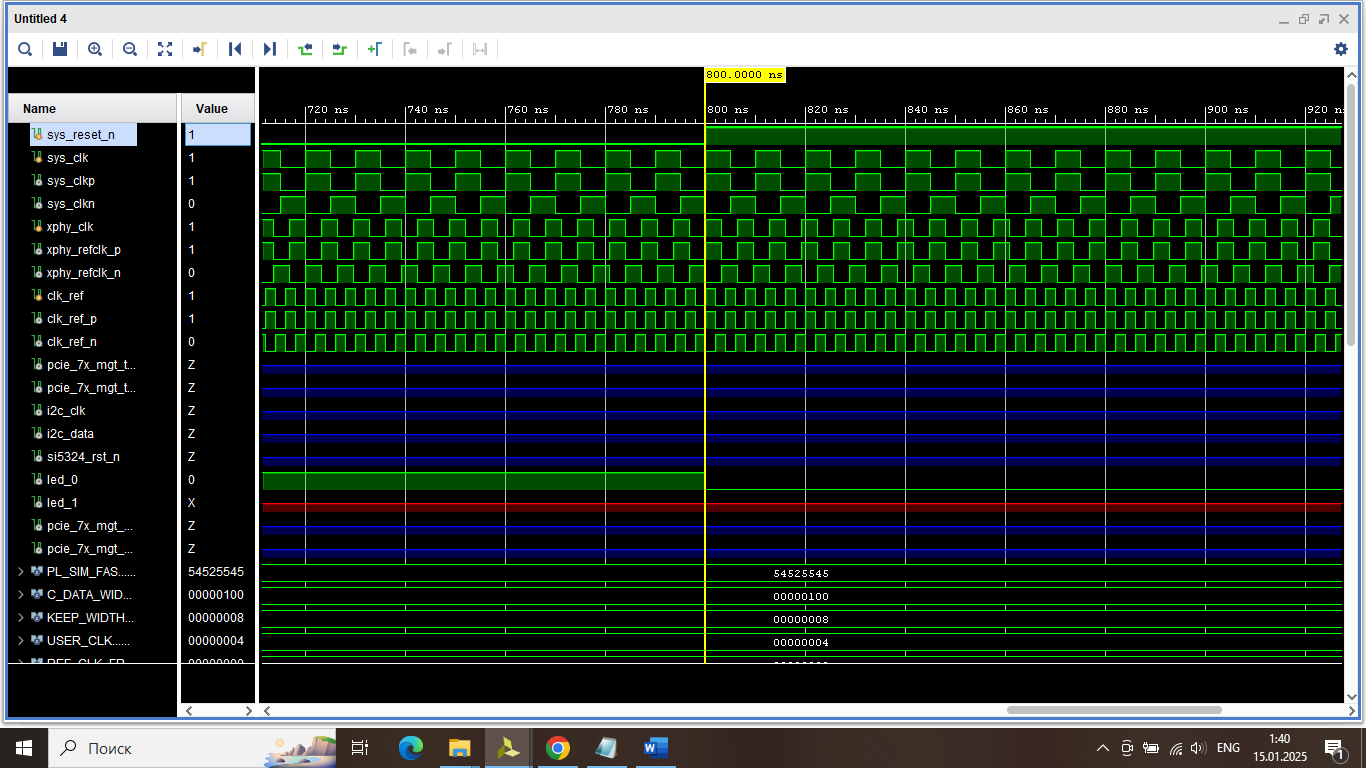
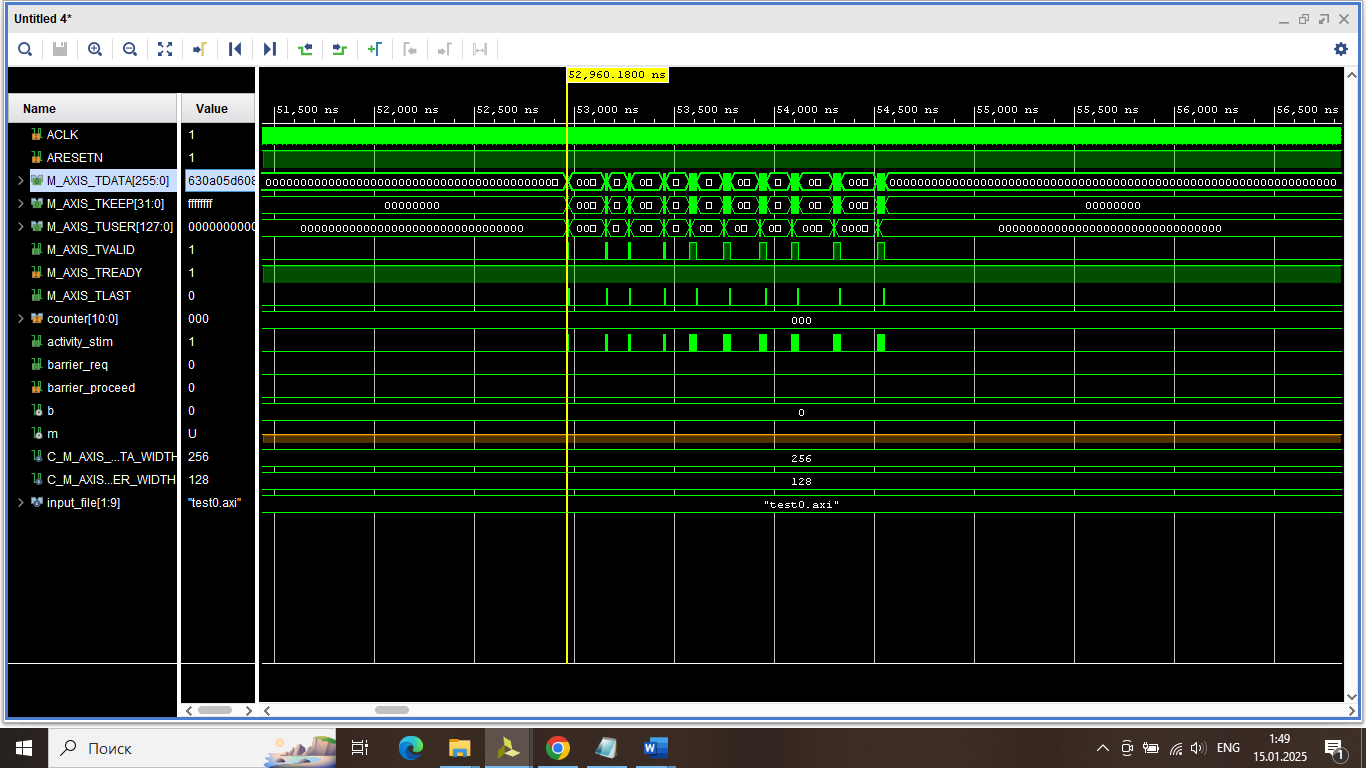


Рис.4. Результат работы testbench.

Сам testbench представляет собой последовательную загрузку данных из тестовых файлов test0.axi - test4.axi, вызываемые в модулях axis\_sim\_stim\_0 - axis\_sim\_stim\_4 соответствнно (рис. 5). Данные тестовые файлы представляют из себя набор команд, записанных по axi\_stream protocole.

Рис.5. Загрузка тестовых команд в модуль axis\_sim\_stim\_0

Тестовый файл представлен в виде последовательности команд:

@ 50000

630a05d608b8caa6066e0040b150ba000045000883761e592100963936420c00, ffffffff, 00000000000000000000000004100040,

613a356401148e00000000009d21e43d185066a3a4ce35ff00103705fcbc517c, ffffffff, 00000000000000000000000000000000.

#cbe11a9b4514ad34219e2bb94843f9223911adda78147494bd3a303364656464, ffffffff, 00000000000000000000000000000000,

#383a303664656464613a360c05070e043a35662e64656464613a37d55c935ed5, ffffffff, 00000000000000000000000000000000,

#0756c94b81efcb18d23a3633646570706f72643a373a30662e3664656464613a, ffffffff, 00000000000000000000000000000000,

#6f72643a38d65b149667aed5c8948f4d4f643d57ddcfd561f9935ed5cb1b70dd, ffffffff, 00000000000000000000000000000000,

#000000000000000000000000000000000000000000000000000d0a0d30203a6e, 000000ff, 00000000000000000000000000000000.

+ 182

630a05d608b8caa6066e0040b150ba000045000883761e592100963936420c00, ffffffff, 00000000000000000000000004100040,

3220312e312f50545448000084c5890018506780c463416017834bf65000a57c, ffffffff, 00000000000000000000000000000000.

#30203a6874676e654c2d746e65746e6f430a0d746e65746e6f43206f4e203430, ffffffff, 00000000000000000000000000000000,

#3a657461440a0d6c6d74682f74786574203a657079542d746e65746e6f430a0d, ffffffff, 00000000000000000000000000000000,

#0a0d544d472033313a35333a3031203031303220626546203632202c69724620, ffffffff, 00000000000000000000000000000000,

#6f69746365746f72502d5353582d580a0d302e322f454647203a726576726553, ffffffff, 00000000000000000000000000000000,

#000000000000000000000000000000000000000000000000000d0a0d30203a6e, 000000ff, 00000000000000000000000000000000.

+ 100

630a05d608b8caa6066e0040b150ba000045000883761e59210096393642aaaa, ffffffff, 00000000000000000000000004100040,

f268043e6614769e37410000012080067600000000609905a600d90d56d51a50, ffffffff, 00000000000000000000000000000000.

#18508bb236644d6fd3de3c41a6c0b0900b52000000000000b0900b520220ef95, ffffffff, 00000000000000000000000000000000,

#d37497664372d2c02448471eec212634905c93d9e614d904db3f0000d2e294fc, ffffffff, 00000000000000000000000000000000,

#243afcd1337c9588d9a504d380d6ccf3612ac580db072597c6077479c5fd7ae4, ffffffff, 00000000000000000000000000000000,

#55df11861e3b9d13fa8f76f98f369b7323dd5f9ea61173007a73688ed8e2f73e, ffffffff, 00000000000000000000000000000000,

#0000000000000000000000000000000000000000000000009de5e39e10beb0fa, 000000ff, 00000000000000000000000000000000.

Данные поступают в модуль ebpf4fpga\_datapath позволяющий выполнять полученные команды на ядре Sephirot (рисунок 6).

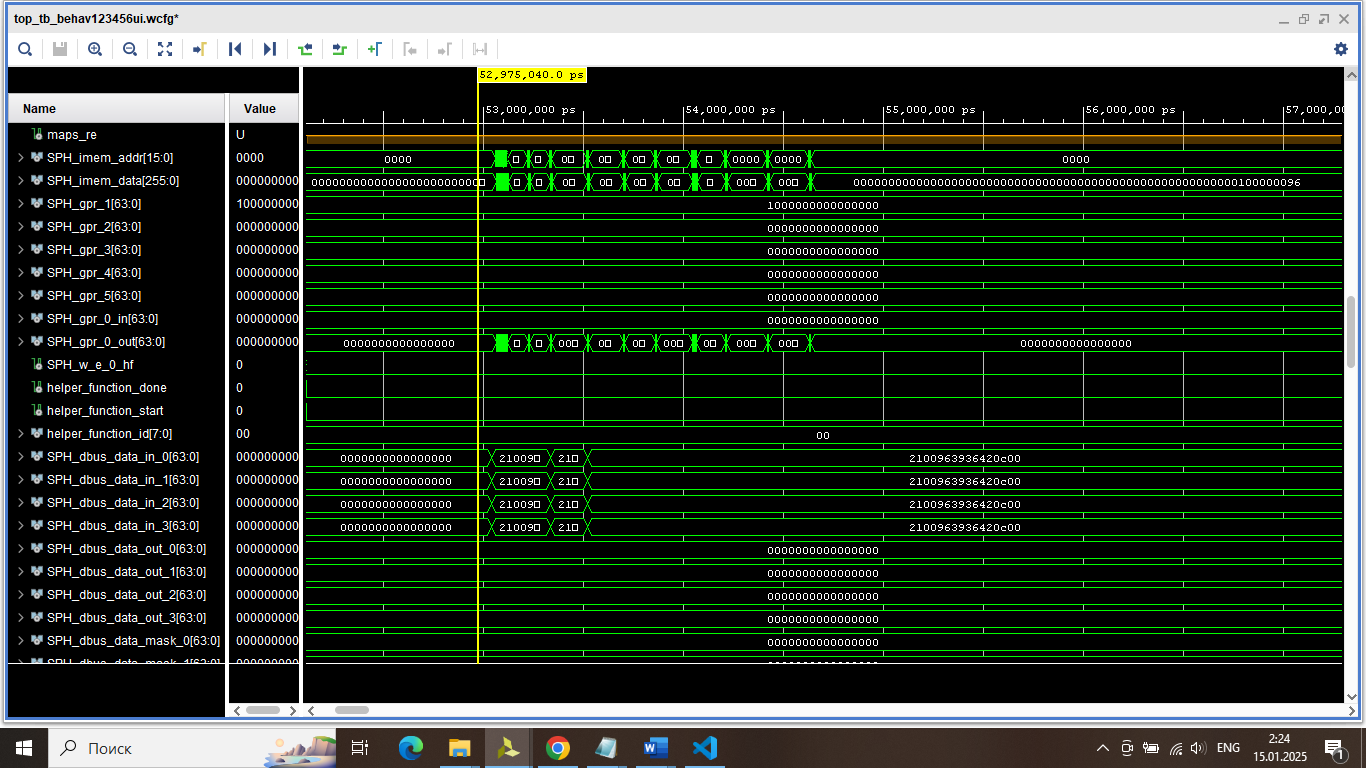


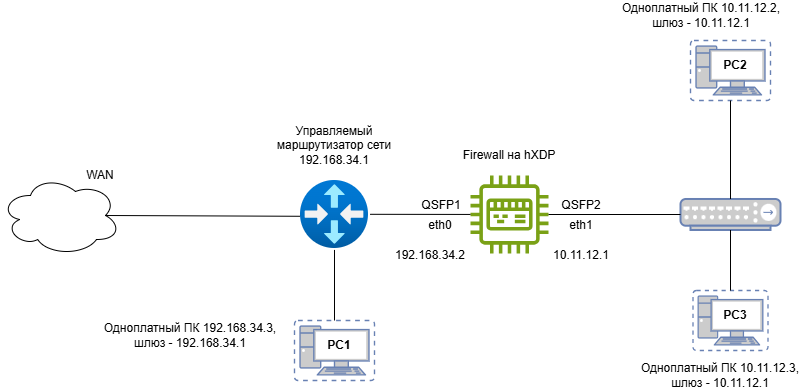
Рис.6. Пример работы модуля Sephirot

Данный тест позволяет в полном объеме быть уверенным в работоспособности проекта. Из дополнений, которые можно добавить дополнительно к тестовому окружению – добавление новых тестовых файлов и проверка работы проекта в условиях работы под высокой нагрузкой, исправить проблему с запуском записи результирующих файлов . Так же стоит отметить, что в данный тестбенч невозможно добавить измерение значения полноты покрытия тестами, так как данная функция реализована в программе Vivado2021.2 и выше.

## 3 Методика тестирования решения hXDP

Для тестирования решения для построения межсетевого экрана необходимо отдельно тестировать как аппаратную часть (модуль Sephirot, модуль интеграции через PCI-e), а также программную часть решения.

Для полноценного тестирования программной части решения была разработана методика на основе серии небольших утилит, тестирующих строго одну функцию в рамках тестов. Простейшая схема сети для проведения тестов приведена на рис.7.

Рис.7. Схема сети для проведения тестов hXDP в качестве основы для построения межсетевого экрана.

При этом предлагается провести следующий набор функциональных тестов (методика тестирования) для проверки возможности использования hXDP в качестве основы для построения межсетевого экрана:

1. Проверка установки ограничения трафика на сетевом (L3) уровне, но по MAC-адресу, например один из одноплатных компьютеров внесен в белый список MAC-адресов, другой — в черный. Поскольку данная проверка может быть выполнена только в рамках сети 10.11.12.0/24, следовательно один из одноплатных компьютеров не сможет отправлять пакеты другому или в сеть интернет в зависимости от настроек.
2. Проверка установки ограничения трафика на сетевом уровне по одному из сетевых протоколов, например, ICMP. Эта проверка должна осуществляться внутри подсети 10.11.12.0/24 с помощью стандартной утилиты ping
3. Проверка установки ограничения трафика по транспортному протоколу по входящему трафику, например, блокирование всего входящего извне трафика по транспортному протоколу UDP и разрешение трафика по TCP. Для этого теста потребуется клиент-серверное приложение, клиентское приложение должно быть установлено на 1 из одноплатных компьютеров PC1 или PC3, а серверное на компьютере PC3. При выполнении этого теста серверное приложение должно открывать 2 TCP-порта (1 – UDP, 1 – TCP), клиент отправляет по N запросов к серверу к каждому из портов при этом в ответ приходят только N ответов, проверка за счет подсчета количества пакетов
4. Проверка контент-фильтрации исходящего не зашифрованного сетевого трафика для этого также потребуется клиент-серверное приложение, серверное приложение развернуто на PC3, клиентское на PC1 или PC2. Проверку фильтрации также можно осуществлять через подсчет количества дошедших до сервера пакетов. Например, можно блокировать пакеты, содержащие слова, drug, bomb или что-то еще, зная количество таких пакетов на PC1 мы можем определить, что контент-фильтрация идет правильно если число полученных пакетов равно общее число пакетов отправленных из клиента минус число пакетов, содержащих запрещенный контент.

Данный набор сценариев не учитывает проверку достаточно сложных сценариев с использованием технологий VLAN и других специализированных протоколов, используемых в сетевых маршрутизаторах.

Кроме проверки hXDP на выполнение функций межсетевого экрана необходимо проверить возможность обработки пакетов на скорости не менее 1 Гбит/с, что является не простой задачей, т. к. требует быстрой отправки сетевых пакетов, но конечные устройства в сети работают под управлением операционных систем, не являющихся системами реального времени, поэтому в этой схеме оценка может быть выполнена грубо. Для этого можно использовать файл большого объема 4-16 Гбайт, например, образ операционной системы. Для проведения тестов воспользуемся контент фильтрации и подмешаем сначала и в конце данных несколько пакетов содержащих запрещенные слова. На клиентском PC2 загружаем файл через поток данных в ОЗУ (потребуется одноплатный ПК с объемом ОЗУ не менее 8Гб), а серверное приложение должно в свою очередь также отдавать данные из памяти. Клиентское приложение замеряет время в течение которого был получен данный файл

## 4 Заключение

В процессе проведения данного этапа исследования были получены следующие результаты:

1. изучены и описаны принципы внутреннего устройства и особенности функционирование проекта hXDP как в настоящем документе, так и в wiki-страницах github-проекта
2. модуль верхнего уровня проекта и IP-ядра, связанные с этим модулем напрямую были успешно портированы на аппаратную платформу Kintex-7 Base-C и синтезированы, при этом к данному проекту были применены временные ограничения, а на сам проект запущены testbench, которые демонстрируют частичную работоспособность, для полного портирования и полностью работоспособных тестов необходимо портировать сабмодули из блок-дизайна, все результаты работы находятся в github-репозитории.
3. описана методика тестирования проекта для оценки его пригодности в качестве основы решения для построения межсетевого экрана, при этом следует отметить, что Sephirot не в полной мере реализует поддержку всех инструкций eBPF, а также по заявлению авторов его аппаратная часть не полностью реализована.

В целом на основе тех представлений, которые были получены в процессе изучения данного проекта предварительно можно, сделать вывод о том, что hXDP можно использовать в качестве основы для построения межсетевого экрана. Но для дальнейшего исследования его возможностей необходимо выполнить следующие задачи:

1. Портировать сабмодули из блок-дизайна.
2. Получить осмысленные результаты а тестбенчах.
3. Ответить на вопрос (исследовать) насколько хорошо и правильно работает оптимизирующий компилятор, написать/запустить соответствующие тесты.
4. Исследовать вопрос необходима ли полная реализация всех возможностей eBPF в Sephirot.
5. Экспериментально опробовать загрузку eBPF байткода в кристалл и описать практический алгоритм работы с устройством.
6. Провести тестирование проекта на базе hXDP в соответствие с описанными в настоящем документе методиками.