Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 12 № 1-2

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2022

Тематика номера:
Смешанная реальность, дополненная реальность, параллельное программирование, многопроцессорные системы реального времени, стохастическое тестирование, медицинская информатика, информационные технологии в издательском деле

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, математические исследования и вопросы численного анализа, история науки и техники.

The topic of the issue:

Mixed reality, augmented reality, parallel programming, multiprocessor systems of real time, stochastic testing, medical informatics, information technology in publishing

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies,
micro- and nanoelectronics, mathematical researches and problems of numerical analysis,
history of science and of technique.

Заведующий редакцией: В.Е.Текунов

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

СОДЕРЖАНИЕ

CONTENT

Технология создания пользовательских приложений с помощью Open GL ES в очках смешанной реальности Microsoft Hololens

И.К. Кисилевич[1], М.В. Михайлюк[2], Д.А. Кононов[3], Д.М. Логинов

[1,2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, [3]ИПУ РАН, Москва, Россия,
[1]ilya.kisilevich@gmail.com, [2]mix@niisi.ras.ru,

Аннотация. В работе рассматривается технология создания и внедрения OpenGL-проектов в очки смешанной реальности Microsoft HoloLens, а также актуальные вопросы, связанные с правильным подключением и корректной работой этих очков. Предлагается подход, в котором задействуется беспроводная передача данных с компьютера на очки, а для запуска приложений используется проект, переводящий вызовы OpenGL ES API в API, поддерживаемое и доступное для платформы Microsoft HoloLens.

Ключевые слова: Microsoft HoloLens, смешанная реальность, дополненная реальность, OpenGL, пространственное сопоставление, ANGLE

1. Введение

В настоящее время одним из важных направлений исследований является соединение реального мира и виртуальных (цифровых) миров, созданных с помощью компьютеров. Термин «виртуальная реальность» (VR, см. [1]) используется для полностью цифровых миров, в которых присутствуют только виртуальные объекты и среды, которые могут взаимодействовать друг с другом. Дополненная реальность (AR, см. [2]) добавляет к восприятию реального мира отдельные мнимые элементы (тексты или изображения), используемые обычно в качестве вспомогательной информации. Смешанная реальность (MR, см. [3]) позволяет соединить реальное и виртуальное пространство в новое окружение, в котором физические и виртуальные объекты могут сосуществовать и взаимодействовать друг с другом в режиме реального времени.

В настоящее время существует востребованность в разработке и применении систем смешанной реальности, как одной из перспективных областей практической деятельности. Уже сейчас она используется в медицине во время хирургических операций, в строительной индустрии в BIM-проектах [0], у пилотов современных боевых самолётов для отображения важной информации на фоне наблюдаемой обстановки прямо на шлеме пилота и др. Смешанная реальность даёт возможность людям обучаться и развивать навыки в режиме симуляции реальных сценариев (например, военный тренинг).

Для погружения пользователя в смешанную реальность используются специальные очки или шлемы. Они позволяют также управлять её объектами без помощи дополнительных устройств. Это становится возможным благодаря камерам и датчикам, установленным на таких устройствах. Камеры воспринимают жесты и движения рук пользователя. Изображение выводится в них на специальные линзы в виде проекций, а инерциальный измерительный блок (IMU, см. [5]) отслеживает повороты и наклоны головы пользователя. Смешанная реальность, которую создают вокруг пользователя такие очки, представляет из себя множество голограмм (виртуальных объектов), помещённых в реальное пространство. Голограммы могут быть анимированными и издавать звуки, а также реагировать на взгляд, жесты и голос пользователя.

Одними из самых современных устройств для смешанной реальности в настоящее время являются очки Microsoft Hololens. Фирма Microsoft разработала технологию создания пользовательских приложений для этих очков с использованием графической библиотеки Direct3D [6]. В данной статье рассматривается подход для запуска проектов, созданных с помощью библиотеки OpenGL ES.

2. Microsoft Hololens

Двуногий Microsoft HoloLens — это очки смешанной реальности, разработанные фирмой Microsoft (рис. 1). С их помощью можно видеть окружающее пользователя реальное пространство с встроенными в него виртуальными объектами, с которыми пользователь может взаимодействовать. Очки содержат 6 камер: в их центре расположена 2.4-х мегапиксельная RGB камера и инфракрасная камера для измерения расстояния до объектов реального мира. Еще по две камеры расположены с правой и левой сторон очков для отслеживания движения головы пользо-

вателя относительно его окружения. Сформированное изображение видимой части смешанной реальности выводится на два полупрозрачных экрана 1268 x 720 пикселов, расположенных перед глазами пользователя и обеспечивающих стереовосприятие этого пространства. Устройство обновляет информацию об окружающем мире 5 раз в секунду. Имеются также 4 микрофона для голосового управления и 2 динамика

Таблица 1. Microsoft HoloLens - Технические характеристики

Система	
ОС	Windows 10
Процессор - CPU	Intel Atom X5-Z8100
Внутренняя память	64Gb
RAM - Оперативная память	2Gb
GPU - Видеокарта	Hololens Graphics 114Mb
Актуальная версия ПО	32-bit Windows 10.0.17763.437
Дата последнего обновления	09.04.2019
Дисплей	
Тип	AR - Дополненная реальность, MR-Смешанная реальность
Размер дисплея	16:9, гориз. угол обзора – 30°, вертик. угол обзора –17°
Разрешение дисплея	1268×720 px.
Частота дисплея	60 Гц.
Камера	
Фронтальные камеры	Камеры глубины и камера для фото/видеосъёмки
Разрешение камеры	2MP Фото/HD Video Camera
Интерфейсы	
Wi-Fi	Wi-Fi 802.11 AC
USB	Micro USB 2.0
Bluetooth	Bluetooth 4.1 LE
Кнопки	Изменение яркости, изменение громкости
Индикаторы	Индикатор состояния аккумулятора
Мультимедиа	
Звук	4 микрофона, 2 динамика, Разъём Audio 3.5 mm
Питание	
Батарея	16500 мА-ч
Время работы	5.5 часов в среднем
Функции	
Управление	Голосовое управление, жесты
Отслеживание	До 6 степеней свободы: отслеживание вращения, отслеживание перемещений

пространственного звука, позволяющие слышать звуки, идущие от виртуальных объектов.

Рис. 1. Устройство Microsoft Hololens

Главным преимуществом Microsoft HoloLens перед другими очками смешанной реальности является то, что данное устройство не требует обязательного подключения к персональному компьютеру, т.е. является абсолютно самостоятельным и может работать автономно, так как компьютер встроен в сами очки. Компьютер Hololens имеет три процессорных чипа: ЦП (CPU), графический процессор (GPU) и блок голографической обработки (HPU). Встроенный аккумулятор рассчитан на 2-3 часа активного использования или 2 недели в режиме ожидания.

Сканирование окружающего пространства происходит при помощи ToF -датчика глубины [7]. Взаимодействие с виртуальными объектами в данных очках реализовано при помощи голосовых команд и жестов пальцами рук, которые отслеживаются встроенными датчиками. Голосовые команды воспринимаются при помощи четырёх микрофонов, спрятанных в передней части очков. Голосовые команды можно использовать для выполнения практически любых действий, например, для открытия приложений, выбора объекта, запуска и остановки программ, фото/видеосъёмки и др. Программное обеспечение очков позволяет распознать положение левой и правой руки пользователя, а также выполняемый им жест: клик пальцем в воздухе (air tap), соединение пальцев вместе (tap & hold), раскрытие ладони (bloom) и др. В Hololens при помощи взгляда можно управлять курсором, который позволяет точно указывать объекты для взаимодействия. Это позволяет пользователю создавать приложения, в которых можно перемещать, захватывать и вращать объекты.

Microsoft Hololens 1-го поколения имеют 5 физических кнопок на корпусе. Они позволяют включать, выключать и переводить в режим сна устройство, а также регулировать громкость звука и яркость изображения. Более подробные технические характеристики отображены в таблице 1.

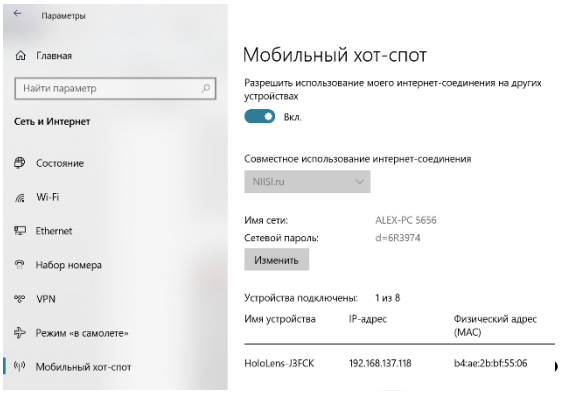

Рис. 2. Подключение устройства по Wi-Fi

Для подключения Microsoft Hololens к ПК необходимо подключить очки и ПК к одной Wi-Fi сети. Для этого можно присоединить ПК к интернету с помощью сетевого кабеля, а для присоединения устройства воспользоваться Wi-Fi адаптером в режиме раздачи сигнала. Wi-Fi адаптер должен работать в диапазоне частот 5 ГГц. После подключения адаптера нужно настроить мобильный хот-спот (Параметры → Сеть и интернет → мобильный хот-спот), и включить разрешение использования интернет-соединения на других устройствах (рис. 2). Затем необходимо надеть очки и подключить их к этой сети через настройки Wi-Fi Hololens.

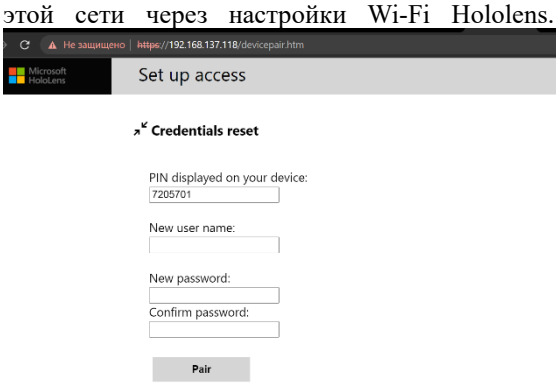

Рис. 3. Подключение портала устройств

Для настройки очков и управления ими удаленно через локальную сеть необходимо использовать портал устройств Windows (Windows Device Portal - WDP), который является веб-сервером, входящим в состав устройств Windows. Чтобы войти в портал устройств необходимо включить режим разработчика на вашем ПК (Параметры → Обновление и безопасность → Для разработчиков → Режим разработчика), ввести в адресную строку браузера IP-адрес устройства, который можно посмотреть в разделе мобильный хот-спот в настройках ПК или в настройках сети самих очков HoloLens. После этого потребуется зарегистрироваться в портале устройств (задать имя и пароль) и ввести PIN-код (который появится на очках, см. рис. 3).

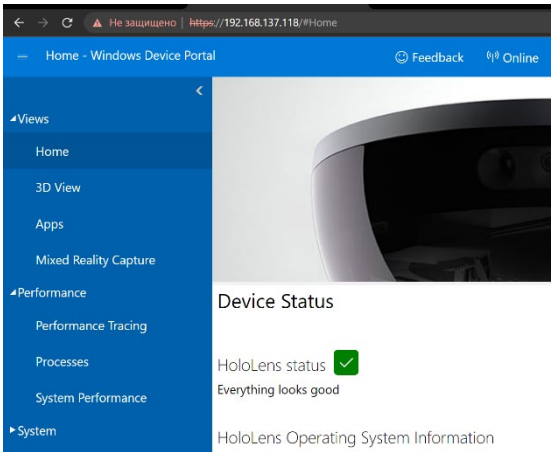

Рис. 4. Портал устройств

После нажатия кнопки «Pair» появится меню портала устройств (см. рис. 4). С помощью этого меню можно отслеживать производительность системы, загружать и устанавливать приложения, передавать между устройствами модели отсканированных помещений, включить запись видео и фото с устройства, включить видео трасляцию смешанной реальности с очков на монитор ПК, управлять установленными приложениями, запустить трассировку приложения и др.

Важным пунктом меню является 3D-view. Он дает возможность увидеть в специальном окне то, что видят очки Hololens и как они интерпретируют виденное (рис. 5).

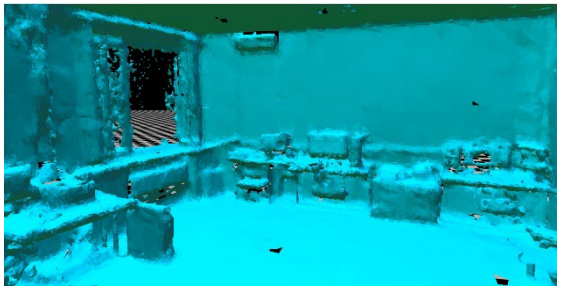

Рис. 5. Представление 3D-view в портале устройств Windows

3. Создание и запуск приложения

Для разработки приложений с использованием очков Hololens потребуется наличие на компьютере 64-bit Windows 10 Pro, Enterprise, или Education Edition. Компьютер должен обладать процессором с четырьмя или более ядрами (либо несколькими процессорами с общим количеством ядер не менее четырех), не менее 8 ГБ оперативной памяти, BIOS, в котором поддерживаются и включены следующие функции:

• виртуализация с поддержкой аппаратного обеспечения;

• преобразование адресов второго уровня (SLAT);

• аппаратное предотвращение выполнения программ из памяти, выделенной для данных (DEP).

Кроме того, необходим графический процессор с DirectX 11.0 или более поздней версии и драйвер WDDM 1.2 или более поздней версии.

Для создания пользовательского приложения можно использовать Visual Studio 2017 или более поздние версии [8]. Перед началом разработки необходимо включить виртуализацию в настройках BIOS (место этих настроек зависит от производителя видеокарты), а также включить компонент Hyper-V (Панель управления\Все элементы панели управления\Программы и компоненты → Включение и отключение компонентов Windows, см. рис. 6).

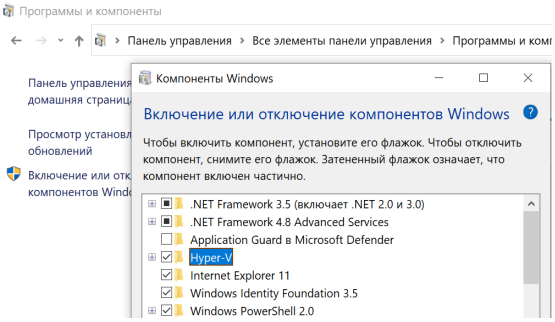

Рис. 6. Включение компонента Hyper-V

В HoloLens нужно включить режим разработчика в настройках (Settings→Update→For developers→Developer mode).

Для удобной разработки приложений в Visual Studio предусмотрена возможность запуска решения на удалённом компьютере (которым являются очки HoloLens), после того, как был указан IP-адрес в свойствах проекта (рис. 7). Выполнение таких программ можно остановить на очках либо в самой Visual Studio. После каждого запуска приложение сохраняется на очках поверх предыдущей версии этого приложения.

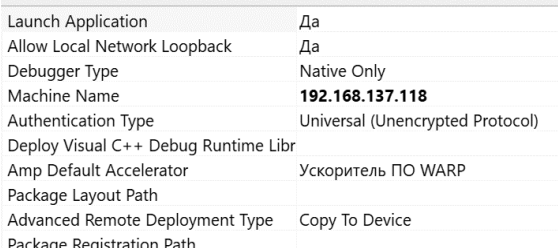

Рис. 7. Адрес удалённого устройства в Visual

В настоящее время создание графических приложений в очках Hololens возможно только с использованием DirectX. Для работы с OpenGL можно воспользоваться проектом ANGLE [9], который позволяет запускать приложения

OpenGL ES на нескольких платформах, переводя их на доступные аппаратно-поддерживаемые API. OpenGL ES (OpenGL for Embedded Systems — OpenGL для встраиваемых систем) — это подмножество графического интерфейса OpenGL, разработанное специально для встраиваемых систем — мобильных телефонов, карманных компьютеров и игровых консолей. OpenGL ES имеет некоторые отличия от OpenGL, хотя и основывается на её спецификации. Различия зависят от конкретных версий этих графических интерфейсов.

Программа ANGLE работает как оболочка для проекта OpenGL ES и переводит его на поддерживаемый очками DirectX. Для начала работы с ANGLE нужно выполнить несколько шагов:

На своём ПК создать копию репозитория ANGLE с сайта GitHub [10] (при помощи команды git clone или скачиванием zip-архива через кнопку “Code”).

При помощи команды git checkout ms-holographic-experimental проверить (обновить) файлы в каталоге ANGLE (см. источник [11] раздел Build ANGLE with Windows Holographic experimental support included, пункт 2)

Открыть angle.sln в Visual Studio и собрать библиотеки libAngle и libEGL в режимах Debug и Release.

Установить шаблон ANGLE для универсальной платформы Windows (в папке templates\ запустить install.bat)

Создать проект на основе шаблона (рис. 8).

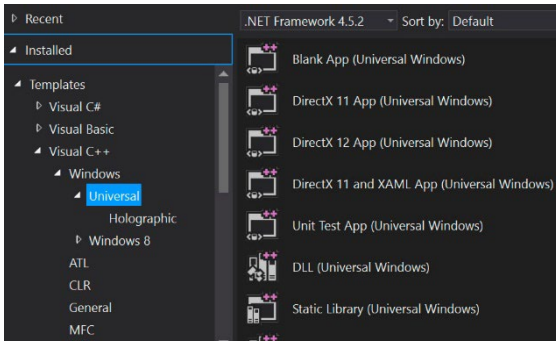

Рис. 8. Создание нового проекта для HoloLens с использованием ANGLE

Для удобства при отладке программ можно использовать эмулятор Microsoft HoloLens, который упрощает и ускоряет процесс тестирования различных версий программ и шаблонов. Эмулятор имеет несколько различных версий сборок, что позволяет тестировать проекты на каждой из них, без необходимости обновления ПО самих очков (рис. 9).

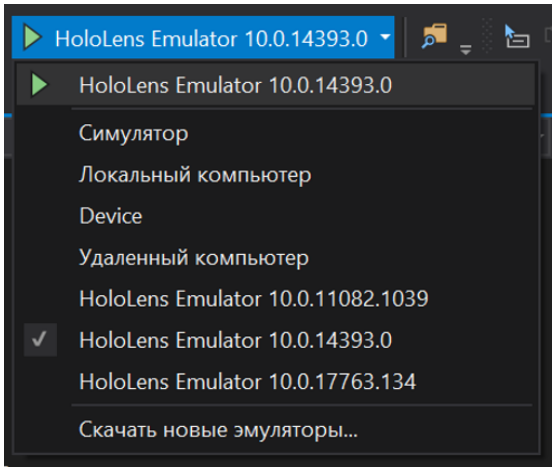

Рис. 9. Выбор эмулятора очков по версии

Благодаря шаблону программы для универсальной платформы Windows с использованием ANGLE становится возможным написание различных программ на её основе. На рис. 10 можно видеть пример такой программы, где в реальное пространство добавляется разноцветный кубик. Пользователь с помощью пальца может управлять перемещениями и поворотами этого кубика.

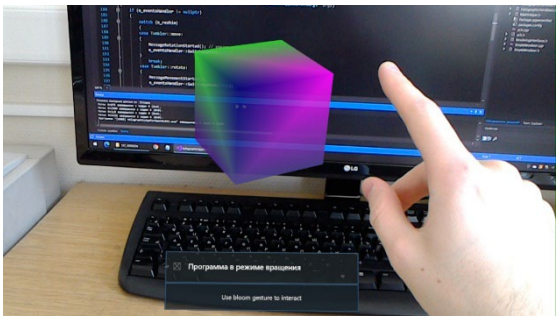

Рис. 10. Визуализация кубика в Microsoft Hololens с помощью OpenGL ES

6. Заключение

В данной работе рассматривается технология создания OpenGL-проектов для очков смешанной реальности Microsoft HoloLens, а также вопросы, связанные с правильным подключением и корректной работой этих очков. В предлагаемом подходе используется беспроводная передача данных с компьютера на очки, а также автоматический перевод вызовов OpenGL ES API в API, которое поддерживается в платформе Microsoft HoloLens.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0012 «Системы виртуального окружения: технологии,

методы и алгоритмы математического моделирования и визуализации. 0580-2022-0012».

Technology for Developing User Applications with Open GL ES in the Microsoft HoloLens Mixed-Reality Device

I.K. Kisilevich, M.V. Mikhaylyuk, D.A. Kononov, D.M. Loginov

Abstract. The paper discusses the technology of creating and implementing Open GL projects into Microsoft Hololens mixed-reality device as well as topical issues related to the correct connection and operation of this device. An approach is proposed, in which wireless data transfer from a computer to glasses is involved, and a project is used to launch applications that translates OpenGL ES API calls into an API supported and available for the Microsoft HoloLens platform.

Keywords: Microsoft HoloLens, mixed reality, augmented reality, OpenGL, spatial mapping, ANGLE

Литература

1. Виртуальная реальность. https://www.iberdrola.com/innovation/virtual-reality (дата обращения: 25.11.2021).

2. Дополненная реальность. https://en.wikipedia.org/wiki/Augmented_reality (дата обращения: 25.11.2021).

3. Mixed Reality. https://docs.microsoft.com/ru-ru/windows/mixed-reality/discover/mixed-reality (дата обращения: 25.11.2021).

4. BIM-проект. https://www.autodesk.ru/solutions/bim (дата обращения: 25.11.2021).

5. Инерциальный измерительный блок. https://docs.microsoft.com/en-us/hololens/hololens1-hardware#:~:text=inertial%20measurement%20unit%20(-,IMU,-).

6. Direct3D. http://ru.knowledgr.com/07934604/Direct3D (дата обращения: 30.11.2021).

7. ToF-camera. https://root-nation.com/ru/posts/tech/ru-tof-camera-v-smartfone/ (дата обращения: 30.11.2021).

8. Инструменты для разработки. https://docs.microsoft.com/ru-ru/windows/mixed-reality/develop/install-the-tools#installation-checklist (дата обращения: 30.11.2021).

9. ANGLE. https://chromium.googlesource.com/angle/angle (дата обращения: 30.11.2021)

10. GitHub ANGLE. https://github.com/microsoft/angle (дата обращения: 01.12.2021).

11. OpenGL ES для Hololens с использованием ANGLE. https://github.com/Microsoft/angle/tree/ms-holographic-experimental (дата обращения: 01.12.2021).

Оптимизация операции быстрого преобразования Фурье в среде OpenCL

А.А. Бурцев[1]

[1] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, burtsev@niisi.msk.ru

Аннотация. Статья посвящена применению технологии OpenCL, позволяющей использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Предлагаются разнообразные варианты параллельных программ, разработанных для ускорения операции быстрого преобразования Фурье в среде OpenCL. Рассматриваемые варианты сравниваются по производительности с целью выявления наиболее оптимального для обработки комплексных векторов различной длины.

Ключевые слова: параллельное программирование, технология OpenCL, гетерогенные системы, микропроцессоры семейства КОМДИВ, быстрое преобразование Фурье

1. Введение

В настоящее время для ускорения вычислительных программ предлагаются разнообразные технологии параллельной обработки. Одни из них (такие, как, например, OpenMP [1, п.5.1]) ориентированы на создание параллельной программы, призванной исполняться на многоядерных процессорах с общей памятью. Другие (например, MPI [1, п.5.2]) позволяют создавать вычислительные программы для их исполнения на семействе компьютеров, связанных сетью.

Для ускорения вычислений в рамках одного компьютера предлагаются также специализированные процессоры SIMD класса. К ним, в частности, относятся векторный сопроцессор CPV и сопроцессор цифровой обработки сигналов CP2 семейства КОМДИВ [2], разрабатываемого в ФНЦ НИИСИ РАН.

Ещё один вид технологии, позволяющий ускорить вычислительную программу за счёт её особого распараллеливания, предполагает использование в вычислительных программах ресурсов мощных графических процессоров. Почти каждая современная видеокарта содержит целый массив однородных специализированных процессоров, которые можно использовать как слаженный ансамбль параллельно функционирующих исполнителей для ускоренного решения одной вычислительной задачи.

Первыми подобную технологию (CUDA) предложила компания NVIDIA. Аналогичную технологию, но уже пригодную для видеокарт различных семейств, предложила компания Apple. Khronos Group приняла эту технологию за основу и довела до стандарта, дав ей название **OpenCL** (**Open C**omputing **L**anguage) [3].

В новую микросхему семейства КОМДИВ включёно графическое ядро с поддержкой технологии OpenCL (версии 1.2). Для испытания возможностей этого графического ядра по технологии OpenCL в НИИСИ РАН был разработан ряд прикладных программ обработки векторов и матриц, характерных для задач линейной алгебры и цифровой обработки сигналов.

Данная статья продолжает знакомство с технологией OpenCL, начатое автором в работах [4,5]. В них рассматривались приёмы разработки в среде OpenCL эффективных программ для решения отдельных задач линейной алгебры (в том числе, операции перемножения больших матриц). А также предлагался вариант программы, позволяющей ускорить операцию быстрого преобразования Фурье (БПФ) для комплексных векторов большой длины (вида $N=2^P$).

Этот первоначальный вариант OpenCL-программы ускорения БПФ был впоследствии усовершенствован с учётом иерархического строения памяти устройств OpenCL-среды. В результате были получены новые варианты OpenCL-программы, которые существенно повысили коэффициенты ускорения операции БПФ в среде OpenCL. Применённые при создании этих вариантов приёмы оптимизации и составляют основное содержание данной статьи.

Изложение материала статьи сопровождается таблицами, в которых представлены количественные показатели ускорения операции БПФ, полученные в результате многократных прогонов разных вариантов OpenCL-программы на различных OpenCL-платформах. Эти показатели характеризуют, какие коэффициенты ускорения операции БПФ удаётся получить в среде OpenCL на обычных (настольных) компьютерах, и какое ускорение можно обеспечить в среде OpenCL, поддерживаемой микросхемой графического ускорителя семейства КОМДИВ.

2. Базовый алгоритм быстрого преобразования Фурье

Все рассматриваемые далее варианты OpenCL-программы, призванные ускорить исполнение операции БПФ в OpenCL-среде за счёт её параллельной обработки ансамблем из нескольких исполнителей, будут сравниваться по производительности (т.е. по времени выполнения) с некоторой базовой (эталонной) программой, которая рассчитана на последовательное исполнение той же операции БПФ на одном процессоре.

В качестве базового алгоритма для выполнения операции БПФ последовательной программой использовался вариант известного алгоритма Кули-Тьюки с прореживанием по времени (см. гл.12 в [6]) для векторов комплексных чисел длиной $N=2^P$ (степени двойки), который был представлен (см. п.2.2 в [5]) в виде такой Си-функции:

```
void FFT(int N,complex *X,complex *Y,
 complex *W) { complex V,T;
long int P,t,s,h,G,R,d,k,u,j,a,b;
BRVPRST(N,X,Y);//ч1.Бит-реверсная перестановка
//ч2.Основной цикл исполнения бабочек Фурье:
 P=iLog2(N); // P= log2N (так что N=2^P)
 s=1;h=2;G=1;R=N/2;d=N/2;
 for(t=1;t<=P;t++) { // на t-ом этапе:
  for(k=0,u=0;k<G;k++,u=u+d){ V=W[u];
   for(j=0,a=k;j<R;j++,a=a+h)
   { b=a+s; BF(X[a],X[b],V,T); }
  }//for k
  s=s*2;h=2*s;G=G*2;R=R/2;d=d/2;
 }//for t
}//FFT
```

Эта функция принимает исходный вектор комплексных чисел **Y** длины **N** и формирует результат преобразования Фурье как новый вектор **X**, используя для вычислений приготовленную таблицу весовых коэффициентов **W**. Она сначала получает (см. **BRVPRST)** в векторе **X** битреверсную перестановку вектора **Y**, а затем в несколько этапов (t=1,..,P, где $P=\log_2 N$) осуществляет многократные вычисления операций **БФ** («**Б**абочка **Ф**урье») над всевозможными парами элементов вектора **X**.

Заметим, что на очередном t-ом этапе (шаге цикла по t) обрабатываемые пары элементов распределяются по группам так, чтобы в каждой группе обрабатывались те пары, для которых операция БФ выполняется с одним и тем же весовым коэффициентом. На каждом шаге цикла по k для k-ой группы вычисляется индекс u для получения из таблицы W их общего весового коэффициента V=W[u], с которым далее выполняются операции БФ для всех пар этой группы, перебираемых в цикле по j.

Такое распределение пар на группы можно наглядно продемонстрировать, если представить обрабатываемый на t-м этапе вектор X как матрицу из 2R строк и G столбцов, уложенную в том же самом месте памяти по строкам. Тогда каждый столбец этой матрицы будет представлять одну группу, а соседние элементы в нём образуют R пар для исполнения операций БФ.

Сама же операция **БФ** в этой Си-функции реализована в форме макроопределения:

```
#define BF(A,B,V,T) \
 { T=B*V; B=A-T; A=A+T; }
```

Кратко охарактеризуем используемые в этом алгоритме основные переменные, которые будут далее использованы с той же ролью и в алгоритмах процедур OpenCL-ядер:

P – кол-во этапов (итераций)=$\log_2 N$, ($N=2^P$)
t – номер очередного этапа (итерации) t=1,2,..,P
На каждом t-ом этапе:
G – количество групп =$2^{(t-1)}$ = 1,2,..,N/2
k – номер очередной группы =0,1,..,G-1
R – число пар в каждой группе =$2^{(P-t)}$ = N/2,..2,1;
так что G*R=N/2 (количество всех пар)
j – номер очередной пары в группе =0,1,..,R-1
s – расстояние между элементами пары =$2^{(t-1)}$
h – расстояние между парами = 2^t =2*s
a,b – индексы элементов j-ой пары в k-ой группе: a= k+j*h; b = a+s;
d – множитель индекса для доступа в таблицу W весовых коэффициентов =$2^{(P-t)}$, (d=R)
u – индекс доступа в таблицу весовых коэффициентов W для всех пар k-ой группы = k*d

3. Первоначальный вариант OpenCL-программы для БПФ

OpenCL-программа, т.е. программа, построенная по технологии OpenCL в расчёте на её параллельное исполнение в среде OpenCL, состоит из основной программы (OpenCL-приложения на языке Си) и нескольких так называемых процедур ядра (на языке OpenCL). Основная программа запускается на основном процессоре (хосте), подготавливает OpenCL-среду и периодически запускает в ней приготовленные процедуры ядра параллельно сразу на всех её вычислительных узлах − так называемых обрабатывающих элементах (processing element).

3.1. Процедуры OpenCL-ядра для распараллеливания алгоритма БПФ

Для осуществления в среде OpenCL параллельного бит-реверсного копирования вектора приготовлена такая процедура ядра:

```
__kernel void fft_brvprst (uint L,
__global complex *X,__global complex *Y,
__global const uint *BRT) { uint k,m;
 k=get_global_id(0); m=BRT[k];
 X[m]=Y[k];// m=бит-реверсное значение k
}//fft_brvprst
```

В ней для получения бит-реверсного значения индекса **m** используется передаваемая в качестве параметра заранее приготовленная таблица **BRT**, в которой каждому целочисленному значению k=0,1,..,N-1 сопоставлено его бит-реверсное значение длины **L** (L=$\log_2$N).

Для параллельного исполнения (на очередном **t**-ом этапе) всех операций БФ над выбранными парами элементов вектора приготовлена процедура ядра:

```
__kernel void fft_btfly
(int n,__global complex *X,
 __global complex *W, int t)
{ uint i,G,R,k,j,s,h,a,b,u;
 i=get_global_id(0);// номер исполнителя (ОЭ)
 G=1<<(t-1); // кол-во групп G= 2^(t-1)
 R=n>>t; // кол-во пар в группе R= 2^(P-t);
 k=i&(G-1); // номер группы = i mod G
 j=i>>(t-1);// номер пары в группе= i div G
 s=G;h=2*s; a=k+j*h; b=a+s; u=R*k;
 BTF(X[a],X[b],W[u]);
}//fft_btfly
```

Такая процедура запускается на каждом параллельном исполнителе на **t**-ом шаге цикла, организованном в теле функции **clFFT** (см. п.3.2).

Каждый такой исполнитель, вызвав функцию **get_global_id**(0), сначала узнаёт назначенный ему уникальный номер (i=0,1,..,N/2-1). Используя параметр t как номер этапа, вычисляет по формулам: $G=2^{(t-1)}$ и $R=2^{(P-t)}$, сколько на данном этапе образуется групп G и сколько пар R в каждой группе. Затем определяет, какую пару из всех имеющихся N/2 пар ему надлежит обрабатывать: из какой она должна быть группы (k) и какая она в ней по порядку (j). Распределение пар по исполнителям осуществляется по формулам: **k= i mod G**, **j= i div G**. Тем самым гарантируется, что каждой паре обязательно будет назначен исполнитель и только один.

Далее в теле процедуры ядра вычисляются индексы (**a**,**b**) элементов выбранной пары, а также индекс **u** для взятия весового коэффициента из таблицы, заданной параметром **W**. И наконец, над назначенной парой исполняется операция «**Б**абочка **Ф**урье» (**БФ**)**,** которая в теле процедуры ядра реализована в виде вызова макроса **BTF(**X[a],X[b],W[u]**)**:

```
#define complex float2
#define BTF(XA,XB,V) \
{complex Xa,Xb,XbV; \
 Xa=XA; Xb=XB; /* XbV=Xb×V) : */  \
 XbV.x= Xb.x*V.x - Xb.y*V.y; \
 XbV.y= Xb.x*V.y + Xb.y*V.x; \
 XA= Xa + XbV; XB= Xa - XbV; }
```

Эти процедуры ядра вместе с необходимыми макроопределениями сосредоточены в файле “FFT.cl”, строки которого анализируются при компоновке программного кода ядра в основной программе.

3.2. Основная программа для запуска исполнения БПФ в среде OpenCL

В основной программе сначала осуществляются все необходимые действия по подготовке OpenCL-среды: инициализируется OpenCL-платформа, дескрипторы OpenCL-устройств, подготавливается OpenCL-контекст, в котором создаётся дескриптор очереди команд.

Затем приготавливаются все объекты для представления в памяти OpenCL-данных, подлежащих обработке, и особый объект **prgrm**, содержащий в откомпилированной форме код всех процедур ядра. И для каждой процедуры ядра создаётся дескриптор типа cl_kernel:

```
kn1=clCreateKernel (prgrm, "fft_brvprst",NULL);
kn2=clCreateKernel (prgrm, "fft_btfly", NULL);
```

Как обеспечить все эти действия, было подробно описано в [4] (в п.2.3-2.5) и в [5] (в п.3.2).

Для выполнения в среде OpenCL операции БПФ оформлена функция **clFFT**:

```
void clFFT(int N,complex *X,complex *Y)
```

в теле которой предусмотрена последовательность действий из следующих 6-ти частей.

1. Загрузка вектора Y в буфер объекта **objY**:

```
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,
  objY,CL_TRUE,0,szC*N,Y,0,0,0);
```

2. Назначение 4-х фактических параметров для процедуры ядра fft_brvprst:

```
cl_uint szI,szM; cl_uint P=iLog2(N);
szI=sizeof(cl_uint); szM=sizeof(cl_mem);
clSetKernelArg(kn1,0,szI,&P);
clSetKernelArg(kn1,1,szM,&objX);
clSetKernelArg(kn1,2,szM,&objY);
clSetKernelArg(kn1,3,szM,&objBRT);
```

3. И запуск её в OpenCL-среде на множестве из N исполнителей для параллельного бит-реверсного копирования вектора из буфера объекта **objY** в буфер объекта **objX** с ожиданием её завершения всеми исполнителями:

```
size_t gWS[1]={N}; cl_event ev1;
clEnqueueNDRangeKernel(cmndQ,
 kn1,1,NULL,gWS,NULL,0,NULL,&ev1);
clFinish(cmndQ);
```

4. Подготовка к многократному запуску процедуры ядра fft_btfly с назначением для неё 3-х первых фактических параметров:

```
clSetKernelArg(kn2,0,szI,&N);
clSetKernelArg(kn2,1,szM,&objX);
clSetKernelArg(kn2,2,szM,&objW);
```

5. Организация основного цикла (по t), на каждом шаге которого процедура ядра fft_btfly запускается на множестве из N/2 исполнителей для параллельного выполнения всех операций БФ t-го этапа (t=1,..,P) с передачей ей значения номера этапа t в качестве 4-го параметра и ожиданием её завершения всеми исполнителями:

```
cl_uint t; cl_event ev2; gWS[0]=N/2;
```

```
for (t=1; t<=P; t++) {
 clSetKernelArg(kn2,3,szI,&t);
 clEnqueueNDRangeKernel(cmndQ,
 kn2,1,NULL,gWS,NULL,0,NULL,&ev2);
 clWaitForEvents(1,&ev2);
}//for t
```

6. Выгрузка в **X** результирующего вектора, полученного в буфере объекта **objX**:

```
clEnqueueReadBuffer(cmndQ,
 objX,CL_TRUE,0,szC*N,X,0,0,0);
```

Этот первоначально созданный вариант OpenCL-программы обозначим как вариант **A**. И оценим его производительность.

3.3. Результаты ускорения операции БПФ в среде OpenCL варианта A

Чтобы оценить, насколько удалось ускорить БПФ с помощью OpenCL-программы, основная программа OpenCL-приложения сначала запускала функцию FFT для обычного (последовательного) исполнения операции БПФ, а затем подготавливала OpenCL-среду и запускала в ней функцию clFFT для выполнения той же операции в среде параллельных исполнителей.

Первоначальный вариант (**A**) программы был разработан (на языке Си) как консольное приложение в среде MS Visual Studio 2017. Он был скомпонован и запускался на различных OpenCL-платформах.

1. Под ОС Windows-10 для платформы с аппаратной конфигурацией, содержащей основной универсальный процессор CPU Intel-i59400 с частотой 2.9 Ггц и встроенный графический процессор GPU UHD 630 с частотой 350 МГц. (далее будем называть её «платформа **Intel**»).

2. Под ОС Windows-7 с аппаратной конфигурацией, содержащей универсальный процессор Intel i3-2100 с частотой 3.1 Ггц и видеокарту NVidia GeForce 1050ti с частотой 1392 Мгц. (далее будем называть её «платформа **NVidia**»).

Скомпонованная программа многократно прогонялась на этих платформах для векторов комплексных чисел одинарной точности различной длины $N=2^P$ (P=8,9,..,24). Результаты этих прогонов были детально изложены в [5]. Здесь же кратко представим результаты этого первоначального варианта A в виде единой сводной таблицы, поместив в неё лишь итоговые коэффициенты так называемого общего и «чистого» ускорения операции БПФ на векторах различной длины (см. таблицу 1). Смысл и формулы расчета этих коэффициентов поясняются внизу приводимой таблицы.

Полученные результаты практически подтверждают, что применение технологии OpenCL действительно позволяет существенно ускорить операцию БПФ для векторов комплексных чисел большой длины вида $N=2^P$ (P=8,9,..,23,24).

Таблица 1. Ускорение операции БПФ OpenCL-программой варианта **A** на платформах Intel и NVidia

P	$N=2^P$	Intel		NVidia	
		K1	K2	K1	K2
8	256	0.09	1.94	0.18	4.59
9	512	0.18	3.92	0.29	6.86
10	1024	0.38	7.96	0.67	15.01
11	2048	0.75	14.40	**1.75**	31.88
12	4096	**1.49**	25.53	3.33	61.85
13	8192	3.02	42.69	5.67	129.35
14	16384	5.68	58.13	14.80	202.75
15	32768	9.83	**76.37**	31.20	299.08
16	65536	18.66	74.58	46.00	**420.17**
17	131072	29.40	75.56	50.00	293.10
18	262144	34.86	67.20	80.25	252.25
19	524288	35.51	51.80	139.33	303.69
20	1048576	39.92	50.93	161.45	303.52
21	2097152	40.89	47.19	160.58	309.70
22	4194304	36.78	40.27	179.70	310.10
23	8388608	35.55	37.87	186.67	312.86
24	16777216	33.55	35.23	**192.70**	**311.90**

$\mathbf{T_{CPU}}$ – время исполнения на CPU
$\mathbf{T_{CL}}$ – полное время исполнения в OpenCL-среде
$\mathbf{T_{Я}}$ – время исполнения OpenCL-процедур ядра
K1 –коэффициент общего ускорения = $\mathbf{T_{CPU}/T_{CL}}$
K2 –коэффициент «чистого» ускорения =$\mathbf{T_{CPU}/T_{Я}}$

3.4. Недостатки OpenCL-программы первоначального варианта

Рассмотренный первоначальный вариант (**A**) OpenCL-программы имеет ряд недостатков, избавившись от которых, можно рассчитывать получить более совершенные версии программы для ускорения операции БПФ. Эти недостатки обусловлены тем, что представленный вариант OpenCL-программы не использует в полной мере все возможности, предоставляемые средой OpenCL.

Запускаемые этой программой процедуры ядра не извлекают никакой выгоды из неоднородного строения памяти OpenCL-устройства. Слабо используют самую быстродействующую внутреннюю память самих обрабатывающих элементов (ОЭ) и совсем не используют локальную память рабочей группы, в которую можно объединять несколько ОЭ для совместной работы. Из-за этого каждый параллельный исполнитель вынужден всякий раз обращаться за данными в общую глобальную память, что, конечно же, замедляет их совместную работу.

К тому же для исполнения второй части БПФ организуется цикл, в котором на каждом шаге процедура ядра fft_btfly запускается заново на всех обрабатывающих элементах. Из-за чего увеличиваются накладные расходы, затрачиваемые на запуск параллельных исполнителей и синхронизацию их совместной работы.

Далее рассмотрим другие варианты OpenCL-программы, направленные на устранение отмеченных недостатков.

4. Вариант OpenCL-программы с одной рабочей группой

Будем запускать параллельные исполнители, выполняющие процедуры ядра, объединяя их все в одну рабочую группу (Working Group). Воспользовавшись средствами синхронизации, предоставляемыми рабочей группой, составим такую процедуру ядра, которая будет запускаться однократно, и сама будет организовывать весь цикл по этапам (цикл по t) для выполнения 2-ой части операции БПФ:

```
__kernel void fft_btflyLoopB
(int n,__global complex *X,
 __global complex *W, int P) {
uint i,G,R,k,j,s,h,a,b,u;
i=get_global_id(0);// номер исполнителя (ОЭ)
for(t=1;t<=P;t++){
 G=1<<(t-1); // кол-во групп G= 2^(t-1)
 R=n>>t; // кол-во пар в группе R= 2^(P-t);
 k=i&(G-1); // номер группы = i mod G
 j=i>>(t-1);// номер пары в группе= i div G
 s=G; h=2*s; a=k+j*h; b=a+s; u=R*k;
 BTF(X[a],X[b],W[u]);
 barrier(CLK_LOCAL_MEM_FENCE);
}//for t
}//fft_btflyLoopB
```

Каждый исполнитель такой процедуры в конце очередного шага цикла будет вызывать функцию **barrier**, чтобы синхронизировать свою дальнейшую работу с другими исполнителями рабочей группы и дождаться момента, когда они все вместе закончат этот шаг цикла.

Добавим в OpenCL-программу действия по созданию дескриптора новой процедуры ядра:

```
kn3=clCreateKernel(prgrm,"fft_btflyLoopB",NULL);
```

Изменим 4-й пункт действий функции clFFT:

```
cl_event ev3;
size_t lWS[1]={N/2}; gWS[0]=N/2;
clSetKernelArg(kn3,0,szI,&N);
clSetKernelArg(kn3,1,szM,&objX);
clSetKernelArg(kn3,2,szM,&objW);
clSetKernelArg(kn3,3,szI,&P);
```

И 5-й пункт списка действий функции clFFT:

```
clEnqueueNDRangeKernel(cmndQ,
 kn3,1,NULL,gWS,lWS,0,NULL,&ev3);
clWaitForEvents(1,&ev3);
```

Полученный вариант OpenCL-программы обозначим как вариант **B**. Результаты прогонов этого варианта программы на платформах Intel и NVidia представлены в таблице 2.

Таблица 2. Ускорение операции БПФ OpenCL-программой варианта **B** на платформах Intel и NVidia

P	$N=2^P$	Intel		NVidia	
		K1	K2	K1	K2
8	256	0.39	5.71	0.40	10.41
9	512	0.85	10.86	1.0	23.39
10	1024	---	---	2.00	45.76
11	2048	---	---	**4.67**	79.61

Сравнивая полученные «чистые» коэффициенты ускорения (K2) с теми, что обеспечивались вариантом A (в таблице 1), можно заметить, что программа варианта B значительно их улучшает: почти в 2 раза на платформе Intel и примерно в 3 раза на платформе NVidia.

Однако, эта OpenCL-программа варианта B обеспечивает такое улучшение лишь для векторов небольшого размера (длины $N \le 512$ для платформы Intel и $N \le 2048$ для платформы NVidia). Вектора же б**о**льшего размера программой варианта B обрабатываться, увы, не могут. Почему, в чём причина?

Дело в том, что для исполнения процедуры ядра fft_btflyLoopB параллельно запускается N/2 обрабатывающих элементов, которые все вместе должны быть объединены в одну рабочую группу. А количество элементов, которое можно собрать в одну группу, ограничено: оно не может превышать значения 256 для платформы Intel, и 1024 – для платформы NVidia.

Как же быть? Можно ли путём организации рабочих групп всё же повысить коэффициент ускорения и для векторов больших размеров?

5. Вариант OpenCL-программы со многими рабочими группами

Допустим, заранее известно, какое максимальное число (MW) обрабатывающих элементов можно объединить в одну рабочую группу. Подберём подходящее значение M для размера группы так, чтобы оно было степенью двойки ($M=2^q$) и не превышало разрешённый максимум: MW=256 (2^8) для платформы Intel и MW=1024 (2^{10}) для платформы NVidia.

Собранные в одну рабочую группу M рабочих элементов (working items) способны совместно исполнить первые несколько (PB) этапов 2-ой части БПФ над всеми парами элементов из выделенного группе непрерывного участка вектора длиной $NB=2 \cdot M$ ($PB=\log_2 NB$). Разбив весь вектор на участки длиной NB, можно каждый участок обрабатывать отдельной рабочей группой, и тем самым, хотя бы на этой первой стадии, охватывающей первые PB этапов (для t=1,..,PB), воспользоваться преимуществами рабочих групп для ускорения вычислительной обработки. А затем на второй стадии оставшиеся этапы (t=PB+1,..,P) можно обрабатывать прежним способом, как в варианте A.

Для выполнения первой стадии составим новую процедуру ядра (fft_btflyLoopC), которую будем запускать на всех обрабатывающих элементах всех рабочих групп. Схема назначения пар исполнителям и алгоритм вычисления параметров пары в этой процедуре поясняются на рисунке 1.

```
__kernel void fft_btflyLoopC
(int n,__global complex *X,
 __global complex *W, int PB) {
uint G,R,k,j,s,h,a,b,u;
uint g,i,WS,kl,jl,kG,jG;
i=get_local_id(0);//локальный номер исполнителя
WS=get_local_size(0);// размер рабочей группы
g=get_group_id (0);// номер рабочей группы
kG=0; jG= g*WS; /* jG=g*WS/G , но G=1*/
for(t=1;t<=PB;t++){
 G=1<<(t-1); R=n>>t; s=G; h=2*s;
 kl=i&(G-1);jl=i>>(t-1);//kl= i%G; jl=i/G
 k=kG+kl;j=jG+jl; u=R*k;
 a=k+j*h; b=a+s; jG=jG>>1;
 BTF(X[a],X[b],W[u]);
 barrier(CLK_LOCAL_MEM_FENCE);
}//for t
}//fft_btflyLoopC
```

В этой процедуре требуется обеспечить, чтобы на каждом этапе исполнители одной и той же рабочей группы выбирали свою пару элементов именно того участка вектора, который поручено обрабатывать данной рабочей группе. И параметры **k** и **j**, задающие пару элементов, для которой данному исполнителю (с локальным номером **i** в рабочей группе **g**) надлежит выполнить операцию БФ на **t**-ом этапе, вычисляются немного сложнее. Сначала определяются их локальные представители в пределах рабочей группы: **kl**= **i mod G**, **jl**= **i div G**, а потом уже сами **k**=**kG**+**kl** и **j**=**jG**+**jl**. При этом **kG** и **jG** задают базовые номер группы элементов и номер пары в этой группе, с которых начинается перечисление всех пар элементов выделенного этой рабочей группе участка вектора.

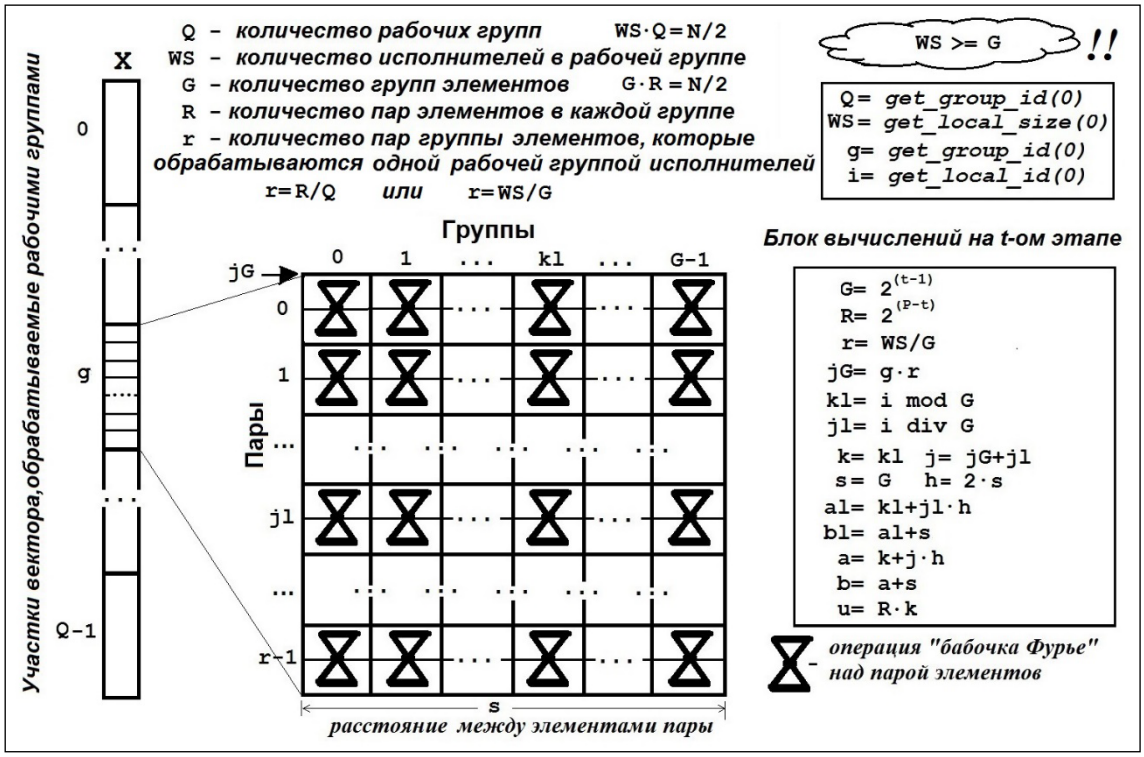

Рис. 1. Схема назначения пар в процедуре ядра 1-й стадии варианта C

Внесём необходимые изменения в OpenCL-программу. Подкорректируем имя новой процедуры ядра при создании для неё дескриптора:

```
kn3=clCreateKernel(prgrm,"fft_btflyLoopC",NULL);
```

Добавим в 4-й пункт функции clFFT назначение параметров для новой процедуры ядра:

```
cl_uint PB=Log2(NB);
clSetKernelArg(kn3,0,szI,&N);
clSetKernelArg(kn3,1,szM,&objX);
clSetKernelArg(kn3,2,szM,&objW);
clSetKernelArg(kn3,3,szI,&PB);
```

И составим 5-й пункт списка действий функции clFFT, выделив в нём две стадии:

```
cl_uint t; cl_event ev2,ev3;
size_t lWS[1]={NB/2}; gWS[0]=N/2;
{ 1-ая стадия } // t=1,..,PB; PB= log2(NB);
clEnqueueNDRangeKernel(cmndQ,
 kn3,1,NULL,gWS,lWS,0,NULL,&ev3);
clWaitForEvents(1,&ev3);
{ 2-ая стадия } // t=PB+1,..,P; P= log(N);
for (t=PB+1; t<=P; t++){
 clSetKernelArg(kn2,3,szI,&t);
 clEnqueueNDRangeKernel(cmndQ,
 kn2,1,NULL,gWS,NULL,0,NULL,&ev2);
 clWaitForEvents(1,&ev2); }//for t
```

Полученный вариант OpenCL-программы обозначим как вариант **C**. Результаты прогонов этого варианта программы на платформах Intel и NVidia представлены в таблице 3.

Сравнивая их «чистые» коэффициенты ускорения (K2) с вариантом A, можно заметить, что они по-прежнему (как и в варианте B) значительно увеличились (в 2-3 раза) лишь для векторов небольшой длины. С ростом же длины вектора (N) процент повышения коэффициента K2 становится всё меньше. И если для векторов длины N=8192 он ещё повышается на 54% на платформе Intel и на 87% на платформе NVidia, то при N=8388608 (2^{23}) коэффициент K2 варианта C превосходит K2 варианта A всего лишь на 10% (Intel) и 13% (NVidia) соответственно.

Таблица 3. Ускорение операции БПФ OpenCL-программой варианта **C** на платформах Intel и NVidia

P	**N=2^P**	**Intel**		**NVidia**	
		K1	**K2**	**K1**	**K2**
8	256	0.39	5.70	0.40	10.40
9	512	0.87	10.96	1.25	23.50
10	1024	1.31	18.79	3.00	46.21
11	2048	2.17	30.26	3.50	79.69
12	4096	3.77	46.98	7.75	138.21
13	8192	6.56	59.97	11.33	240.73
14	16384	11.42	79.26	24.33	319.15
15	32768	18.57	**95.78**	52.33	421.79
16	65536	29.36	85.77	70.00	515.48
17	131072	41.92	83.90	150.50	341.24
18	262144	**46.25**	73.78	160.00	306.92
19	524288	35.64	51.55	159.80	342.92
20	1048576	39.99	50.13	177.90	353.94
21	2097152	37.38	43.02	192.40	355.49
22	4194304	37.64	40.97	205.11	**372.95**
23	8388608	35.34	37.86	208.05	352.56
24	16777216	34.18	36.04	**224.57**	367.75

Выигрыш использования программы варианта C на векторах большой длины оказывается, увы, не таким значительным, как для векторов малых размеров. Чем это можно объяснить? Объяснение простое: с ростом N увеличивается доля той обработки, которая выполняется на 2-й стадии, а доля 1-ой стадии, для ускорения которой и были применены рабочие группы, соответственно уменьшается, поэтому и общий выигрыш в повышении коэффициента K2 становится всё меньше.

Возникает вопрос: можно ли применить рабочие группы и на 2-й стадии тоже? Ответ: можно, но потребуется усложнить запускаемую процедуру ядра. И такой усложнённый вариант программы мы рассмотрим далее (в п.7). Но прежде попробуем оптимизировать полученную OpenCL-программу варианта C, применяя ещё одну полезную возможность, которую обеспечивает рабочая группа. Речь пойдёт об использовании локальной памяти.

6. Использование локальной памяти в рабочих группах

Рабочим элементам, т.е. обрабатывающим элементам, объединённым в одну и ту же рабочую группу, предоставляется общая для них всех локальная память. Предполагается, что доступ к ней осуществляется быстрее, чем к глобальной памяти OpenCL-устройства. Поэтому для повышения общей производительности следует стараться сохранять в этой локальной памяти ту часть элементов вектора, которая наиболее активно обрабатывается всеми исполнителями рабочей группы.

Учитывая это, модифицируем процедуру ядра, запускаемой на первой стадии, следующим образом. Пусть на первом этапе (при **t**=1) каждый исполнитель группы, вытащив из глобальной памяти пару элементов (с индексами **a** и **b**) и выполнив с ними операцию БФ, помещает результаты в локальную память (с индексами **al** и **bl**). На последующих этапах (при **t**=2,..,PB-1) все исполнители совершают операции БФ со своими парами, взятыми из локальной памяти. А на последнем этапе (при **t**=PB), взяв элементы из локальной памяти, каждый исполнитель помещает результат операции БФ обратно в глобальную память.

Выполнив такую модификацию, получим следующую процедуру ядра (fft_btflyLoopCL):

```
kernel void fft_btflyLoopCL
(int n,__global complex *X,
 __global complex *W, int PB
 __local complex *XL ){
uint G,R,WS,k,j,s,h,a,b,u;
uint g,i,kG,jG, kl,jl,al,bl;
i=get_local_id(0);//локальный номер исполнителя
WS=get_local_size(0); // размер раб.группы
g=get_group_id (0); // номер раб.группы
kG=0; jG=g*WS; /* jG=g*WS/G , но G=1*/
get_GRkjuabL(1); a=k+j*h; b=a+s;
if(PB==1){BTF(X[a],X[b],W[u]);}else{
BTFLY(X[a],X[b],W[u],XL[al],XL[bl]);
for(t=2;t<=PB-1;t++){
 barrier(CLK_LOCAL_MEM_FENCE);
 get_GRkjuabL(t);
 BTF(XL[al],XL[bl],W[u]);
}//for t
 barrier(CLK_LOCAL_MEM_FENCE);
 get_GRkjuabL(PB); a=k+j*h;b=a+s;
 BTFLY(XL[al],XL[bl],W[u],X[a],X[b]);
}//if PB==1
}//fft_btflyLoopCL
```

В этой процедуре на каждом этапе для определения параметров обрабатываемой пары кроме переменных **G**,**R**, **k**,**j**, **u**, **a**,**b**, **kl**, **jl**, необходимо ещё и вычислять индексы **al**,**bl**, задающие места расположения элементов пары в локальной памяти (см. рис. 1). Чтобы не записывать такой блок вычислений несколько раз, предложено

макроопределение **get_GRkjuabL**, которое вызывается в этой процедуре 3 раза:

```
#define get_GRkjuabL(t) { \
 G=1<<(t-1); R=n>>t; s=G; h=2*s; \
 kl=i&(G-1); jl=i>>(t-1); \
 k=kG+kl; j=jG+jl; u=R*k; \
 al=kl+jl*h; bl=al+s; jG=jG>>1; }
```

Отметим также, что на первом и последнем этапе для исполнения операции БФ применяется другое макроопределение **BTFLY**, которое позволяет поместить результат операции в другое место памяти:

```
#define complex float2
#define BTFLY(XA,XB,V,nXA,nXB) \
{complex Xa,Xb,XbV; \
 Xa=XA; Xb=XB; /* XbV=Xb×V) : */  \
 XbV.x= Xb.x*V.x - Xb.y*V.y; \
 XbV.y= Xb.x*V.y + Xb.y*V.x; \
 nXA= Xa + XbV; nXB= Xa - XbV; }
```

И ещё заметим, что в этой процедуре отдельно обрабатывается особый случай, когда процедура ядра должна выполнить лишь один этап (PB=1). В этом случае результат операции БФ должен сразу помещаться в глобальную память, и локальная память тогда не используется.

Изменим немного и OpenCL-программу. Поменяем имя процедуры ядра:

```
kn3=clCreateKernel(prgrm,"fft_btflyLoopCL",0);
```

И добавим назначение 5-го параметра (**XL**) процедуры ядра, задав в качестве его характеристики требуемый размер (szC·NB) для вектора комплексных чисел, не уточняя, где именно он будет располагаться в локальной памяти:

```
clSetKernelArg(kn3,4,szC*NB,NULL);
```

Полученный вариант OpenCL-программы обозначим как вариант $\mathbf{C^L}$. Результаты прогонов этого варианта программы на платформах Intel и NVidia представим в таблице 4.

Таблица 4. Ускорение операции БПФ OpenCL-программой варианта $\mathbf{C^L}$ для платформ Intel и NVidia

P	$N=2^P$	Intel		NVidia	
		K1	K2	K1	K2
8	256	0.39	4.90	0.50	15.66
9	512	0.87	10.05	0.83	26.00
10	1024	1.16	17.90	2.00	56.31
11	2048	2.18	29.76	4.67	99.88
12	4096	3.74	45.93	7.50	158.97
13	8192	6.22	64.29	17.00	286.46
14	16384	11.38	78.70	37.00	421.76
15	32768	17.83	95.11	52.00	541.96
16	65536	30.43	85.08	69.00	666.94
17	131072	40.91	83.57	75.00	392.85
18	262144	45.39	72.70	107.00	346.71
19	524288	41.71	54.94	167.20	399.06
20	1048576	45.78	54.40	197.33	388.94
21	2097152	47.02	52.95	192.70	389.61
22	4194304	40.89	44.40	206.65	383.04
23	8388608	39.24	41.37	213.33	380.57
24	16777216	36.23	37.78	221.83	367.68

Чтобы наглядно ощутить, насколько использование локальной памяти рабочих групп позволяет повысить коэффициенты ускорения, приведём сравнительную таблицу (см. таблицу 5), в которой для OpenCL-программ вариантов C и C^L, опробованных на платформе NVidia, представлены «чистые» коэффициенты ускорения K2 вместе с показателями, характеризующими их рост по отношению к коэффициентам K2 варианта A.

Таблица 5. Сравнение коэффициентов ускорения операции БПФ вариантов **C** и $\mathbf{C^L}$ платформы NVidia

P	$N=2^P$	вариант **C**		вариант $\mathbf{C^L}$	
		K2	**K2^**	**K2**	**K2^**
8	256	10.40	3.41	15.66	3.41
9	512	23.50	3.42	26.00	**3.79**
10	1024	46.21	3.09	56.31	3.75
11	2048	79.69	2.51	99.88	3.13
12	4096	138.21	2.17	158.97	2.57
13	8192	240.73	**1.87**	286.46	**2.21**
14	16384	319.15	1.60	421.76	2.08
15	32768	421.79	1.40	541.96	1.81
16	65536	515.48	1.21	666.94	1.59
17	131072	341.24	1.17	392.85	1.34
18	262144	306.92	1.22	346.71	1.37
19	524288	342.92	1.18	399.06	1.31
20	1048576	353.94	1.17	388.94	1.28
21	2097152	355.49	1.15	389.61	1.26
22	4194304	372.95	1.14	383.04	1.24
23	8388608	352.56	**1.13**	380.57	**1.22**
24	16777216	**367.75**	1.13	367.68	1.18

K2 – коэффициент ускорения = $\mathbf{T_{CPU}}$ / **Tя**
K2^ – коэффициент роста = **K2** / **K2** варианта **A**

Из этой таблицы видно, что использование локальной памяти заметно улучшает коэффициент ускорения K2 при обработке векторов любой длины. Так, например, для векторов длины N=8388608 (2^{23}) этот коэффициент подрастает уже на 22% (вместо 13% варианта C), а для векторов длины N=8192 – на 121% (вместо 87%).

Поэтому далее во всех вариантах будем активно использовать локальную память рабочих групп везде, где это возможно. Например, можно аналогичным образом модифицировать процедуру ядра варианта B, чтобы она тоже работала с локальной памятью (полученный вариант OpenCL-программы назовём вариантом $\mathbf{B^L}$).

7. Двойное применение рабочих групп с локальной памятью

Итак, применение рабочих групп с локальной памятью позволило заметно ускорить первую стадию операции БПФ (на этапах t=1,..,PB). Попробуем усилить полученный эффект и применить этот же приём оптимизации и на второй стадии (этапах t= PB+1,..,P) тоже.

На 1-ой стадии БПФ каждой рабочей группе, состоящей из M исполнителей, выделялся для совместной обработки непрерывный участок вектора из NB=2·M элементов, образующих M пар на каждом этапе. На 2-ой стадии каждой рабочей группе тоже предстоит вместе поэтапно обрабатывать одну и ту же цепочку из нескольких (N/NB) элементов. Но таких элементов, которые уже не стоят в заданном векторе непрерывно друг за другом, а следуют в нём с некоторым постоянным шагом (**s**=$2^{\mathbf{PB}}$).

На каждом этапе требуется корректно распределять пары этих элементов между исполнителями группы. А для этого в процедуре ядра необходимо задать формулы, по которым на этапе **t** исполнитель с локальным номером **i** мог бы правильно вычислить параметры (**k**,**j**,**a**,**b**,**u**) именно той пары элементов, которая должна назначаться ему для исполнения операции БФ. Для цепочки элементов, разбросанных по вектору, выразить такие формулы становится сложнее, чем для сплошного блока элементов, как это было в процедуре ядра для 1-ой стадии.

Представим вектор как матрицу, размещённую в том же месте памяти по строкам размером по **s** элементов. Тогда цепочки элементов, которую должны обрабатывать исполнители одной и той же рабочей группы, окажутся столбцами этой матрицы (см. рис. 2).

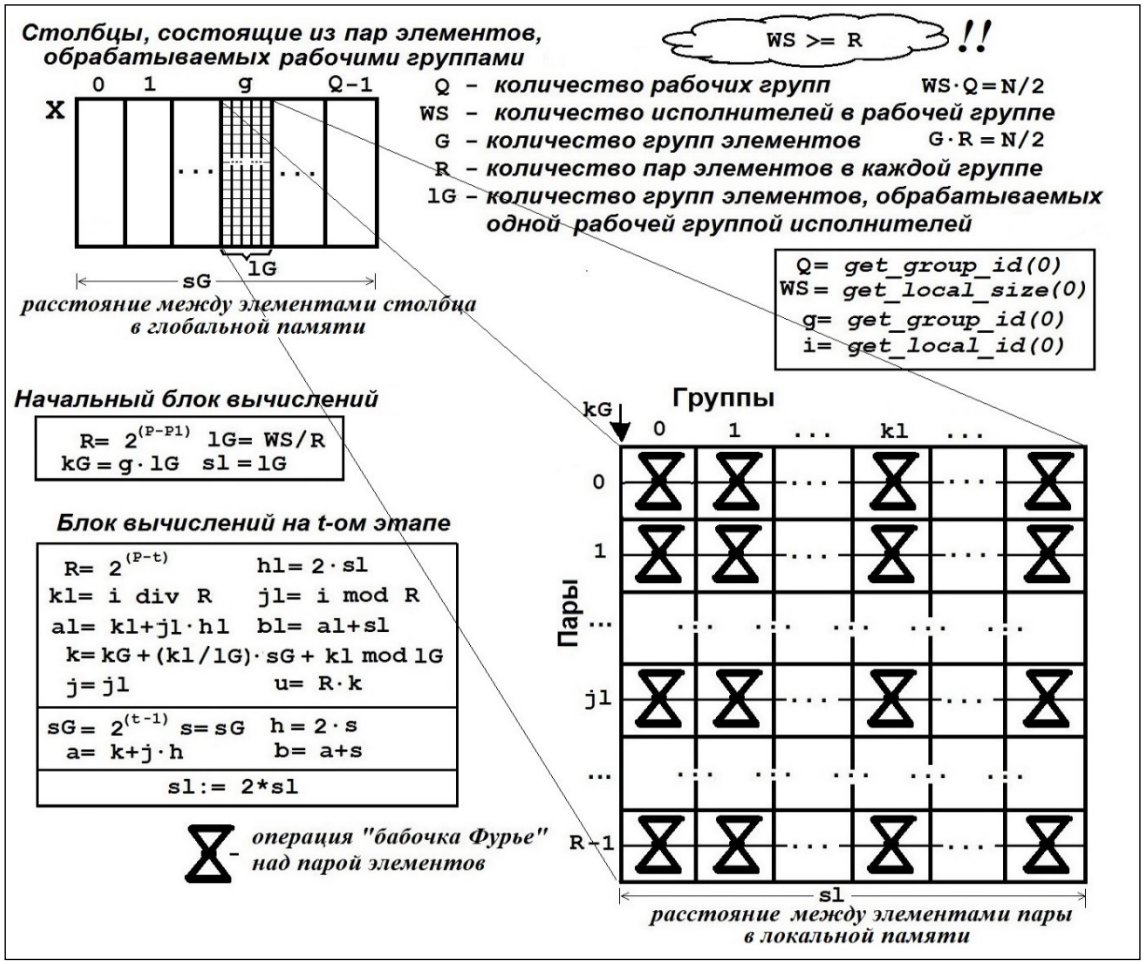

Рис. 2. Схема назначения пар в процедуре ядра 2-й стадии варианта D

Такое представление позволяет пояснить алгоритм вычисления параметров назначаемой пары в процедуре ядра, вызываемой на 2-ой стадии:

```
kernel void fft_btflyLoopTL
(int n,__global complex *X,
 __global complex *W, int P1,
 int P2, __local complex *XL ){
uint lG,R,k,j,sl,h,a,b,u;
uint g,i,WS,kl,jl,kG,sG;
i=get_local_id(0);//локальный номер исполнителя
WS=get_local_size(0);// размер раб.группы
g=get_group_id (0);// номер раб.группы
R=n>>P1; sG=1<<(P1-1);
lG=WS/R; //предполагается, что WS >= R !! */
kG=g*lG;//lG=к-во групп эл-тов в рабочей группе
sl=lG; get_RkjuabL(P1); sl=sl<<1;
a=k+j*2*sG; b=a+sG;
if(P1==P2){BTF(X[a],X[b],W[u]);}else{
BTFLY(X[a],X[b],W[u],XL[al],XL[bl]);
for(t=P1+1;t<=P2-1;t++){
 barrier(CLK_LOCAL_MEM_FENCE);
 get_RkjuabL(t); sl=sl<<1;
 BTF(XL[al],XL[bl],W[u]);
}//for t
 barrier(CLK_LOCAL_MEM_FENCE);
 get_RkjuabL(P2); sG=1<<(P2-1);
 a=k+j*2*sG; b=a+sG;
BTFLY(XL[al],XL[bl],W[u],X[a],X[b]);
}//if P1==P2
}//fft_btflyLoopTL
```

Для вычисления параметров обрабатываемой на очередном этапе пары в этой процедуре ядра используется другое макроопределение:

```
#define get_RkjuabL(t) { \
 R=n>>t; jl= i % R; kl= i / R; \
 k=kG+(kl/lG)*sG+kl%lG; j=jl; \
 al=kl+2*jl*sl; bl=al+sl; u=R*k; }
```

Дополним OpenCL-программу, чтобы получить дескриптор для новой процедуры ядра:

```
kn4=clCreateKernel(prgrm,"fft_btflyLoopTL",NULL);
```

Назначим ей параметры при выполнении 4-го пункта списка действий функции clFFT:

```
cl_uint PB1=PB+1;
clSetKernelArg(kn4,0,szI,&N);
clSetKernelArg(kn4,1,szM,&objX);
clSetKernelArg(kn4,2,szM,&objW);
clSetKernelArg(kn4,3,szI,&PB1);
clSetKernelArg(kn4,4,szI,&P);
clSetKernelArg(kn4,5,szC*(N/NB),NULL);
```

И чтобы запускать такую процедуру ядра на 2-ой стадии БПФ, поставим в 5-й пункт списка действий функции clFFT (вместо цикла по t) вот такой фрагмент:

```
{ 2-ая стадия }// t=PB+1,..,P; P= log(N);
lWS[0]= (N/NB)/2;
clEnqueueNDRangeKernel(cmndQ,
 kn4,1,NULL,gWS,lWS,0,NULL,&ev4);
clWaitForEvents(1,&ev4);
```

После произведённых преобразований получим новый вариант OpenCL-программы, который будем называть далее вариантом **D**. Поскольку он отличается от предыдущего варианта C^L иным исполнением 2-ой стадии БПФ, то и ощутить разницу в производительности этих вариантов можно будет лишь на векторах такой длины, при обработке которых эта 2-я стадия исполняется. Приведём сравнительную таблицу коэффициентов ускорения этих вариантов для платформы NVidia, где эта разница может проявиться при длине N≥4096 (см. таблицу 6).

Таблица 6. Сравнение коэффициентов ускорения операции БПФ вариантов $\mathbf{C^L}$ и **D** платформы NVidia

P	$N=2^P$	вариант C^L		вариант **D**	
		K2	**K2^**	**K2**	**K2^**
12	4096	158.97	2.57	102.97	1.66
13	8192	286.46	2.21	187.78	1.45
14	16384	421.76	2.08	299.09	1.48
15	32768	541.96	1.81	445.64	1.49
16	65536	666.94	1.59	608.41	1.45
17	131072	392.85	1.34	**545.52**	**1.86**
18	262144	346.71	1.37	495.14	1.96
19	524288	399.06	1.31	576.73	1.90
20	1048576	388.94	1.28	566.46	1.87
21	2097152	389.61	1.26	562.78	1.82
22	4194304	383.04	1.24	521.69	1.68

Как видно из этой таблицы, вариант D действительно улучшает коэффициент «чистого» ускорения (K2) операции БПФ, но лишь для векторов длиной $N\geq 2^{17}$(=131072).

Получается, что произведённая модификация 2-ой стадии БПФ путём замены итерационного запуска процедуры ядра fft_btfly на одиночный запуск процедуры ядра fft_btflyoopTL с применением рабочих групп и локальной памяти оказывается оправданной в том случае, если эта 2-ая стадия состоит как минимум из 6-ти этапов. А при меньшем числе этапов усложнённые вычисления параметров назначаемых пар, исполняемые процедурой fft_btflyoopTL, проигрывают по быстродействию циклическим вызовам процедуры fft_btfly, в которой параметры предназначенной исполнителю пары определяются значительно проще.

Заметим также, что вариант D не может применяться для векторов с длиной $N>2^{22}$ (на платформе NVidia) и $N>2^{18}$ (на платформе Intel). Возникает вопрос, почему? Всё дело в том, что в этом варианте операция БПФ ограничивается проведением 2-х стадий, каждая из которых может охватывать не более PW этапов. Причём, величина PW зависит от максимально допустимого количества (MW) исполнителей (рабочих элементов) в рабочей группе и определяется как $\log_2(MW)+1$. Поскольку для платформы Intel MW=256, то для неё PW=9; а для платформы NVidia MW=1024 и следовательно PW=11. Вот этим и объясняется ограниченность варианта D.

Дополнить же этот вариант ещё одной стадией с циклическим вызовом процедуры fft_btfly (как в вариантах C или C^L) не представляется возможным, так как процедура ядра

fft_btflyLoopTL рассчитана на работу лишь при соблюдении условия: WS≥R, означающего, что количество пар (R) элементов в каждой группе не должно превосходить количества исполнителей (WS), имеющихся в каждой рабочей группе.

В целях дальнейшей оптимизации операции БПФ при последующем развитии OpenCL-программы попробуем снять это ограничение, а также будем активно использовать имеющуюся у каждого обрабатывающего элемента свою собственную внутреннюю память.

8. Активное использование внутренней памяти

Кроме доступа к глобальной памяти OpenCL-устройства и локальной памяти рабочей группы каждый обрабатывающий элемент может использовать ещё и свою внутреннюю или личную (private) память. Конечно, объём этой памяти очень небольшой, но зато обращение к ней выполняется намного быстрее, чем к локальной (и тем более к глобальной) памяти.

Попробуем выгодно использовать такую память для хранения отдельных участков обрабатываемого вектора, чтобы ещё более ускорить операцию БПФ. Для этого изменим OpenCL-программу варианта D следующим образом.

Заметим, что на начальных этапах 2-й части БПФ обрабатываются пары соседних элементов, взятые из непрерывного участка вектора. Значит, каждому обрабатывающему элементу можно поручить исполнить все операции БФ на нескольких **PA** первых этапах со всеми парами элементов назначенного ему участка длиной **NA**=$2^{\mathbf{PA}}$, сохраняя при этом эти элементы в своей личной памяти для ускорения доступа к ним.

Чтобы осуществить это, добавим в алгоритм 2-й части БПФ предварительную (нулевую) стадию (помимо уже рассмотренных двух), на которой будем запускать такую процедуру ядра:

```
#define MAX_NA 16
kernel void fft_btflyLoopA(int n,
 __global complex *X,
 __global complex *W, int PA) {
uint i,S,A,t,s,h,G,r,d,u,k,j,a,b;
complex XA[MAX_NA]; complex V;
i= get_global_id(0);// глобальный номер ОЭ
S=1<<PA;// S=2^PA размер обрабат. участка вектора
A=i*S; //A= его начальный индекс в векторе X
s=1;h=2;G=1;d=n/2;r=S/2; V=W[0];
if(PA==1){ // это если только один этап
 for(j=0,a=0;j<r;j++,a=a+h)
 {b=a+s; BTF(X[A+a],X[A+b],V);}
}else{
 for(j=0,a=0;j<r;j++,a=a+h){ b=a+s;
  BTFLY(X[A+a],X[A+b],V,XA[a],XA[b]);}
 for(t=2;t<Z;t++){
  for(k=0,u=0;k<G;k++,u=u+d){ V=W[u];
   for(j=0,a=k;j<r;j++,a=a+h)
    {b=a+s;BTF(XA[a],XA[b],V);}
  }//for k
  h=h*2;s=s*2;G=G*2;r=r/2;d=d/2;
 }//for t
 for(k=0,u=0;k<G;k++,u=u+d){ V=W[u];
  for(j=0,a=k;j<r;j++,a=a+h){ b=a+s;
  BTFLY(XA[a],XA[b],V,X[A+a],X[A+b]);}
 }//for k
}//if PA==1
}//fft_btflyLoopA
```

Выполняя эту процедуру, каждый параллельный исполнитель самостоятельно осуществляет первые **PA** этапов операций БФ со всеми парами элементов на выделенном ему участке вектора. Алгоритм этой процедуры, в основном, заимствует алгоритм 2-ой части функции FFT (см. п.2). Он отличается от него лишь тем, откуда читаются элементы вектора, над которыми выполняется операция БФ, и куда помещаются затем её результаты. На 1-ом этапе такие элементы берутся из глобальной памяти, а результаты помещаются в личную память. На последнем, наоборот, берутся из личной памяти, а результаты сохраняются в глобальной памяти. А на всех промежуточных этапах обрабатываются пары элементов, хранящиеся в личной памяти.

Для запуска этой процедуры ядра внесём необходимые изменения в OpenCL-программу.
Прежде всего, требуется получить дескриптор новой процедуры ядра:

```
kn5= clCreateKernel(prgrm,"fft_btflyLoopA",NULL);
```

Перед запуском назначим этой процедуре параметры (в 4-ом пункте функции clFFT):

```
#define NA 16
cl_uint PA=iLog2(NA);
clSetKernelArg(kn5,0,szI,&N);
clSetKernelArg(kn5,1,szM,&objX);
clSetKernelArg(kn5,2,szM,&objW);
clSetKernelArg(kn5,3,szI,&PA);
```

Размер участков, обрабатываемых на этой нулевой стадии, зафиксируем заранее (**NA**=16).

Чтобы запускать такую процедуру ядра на нулевой стадии БПФ, добавим в начало 5-го пункта функции clFFT вот такой фрагмент:

```
{ 0-ая стадия } // t=1,..,PA; PA= log2(NA);
gWS[0]=N/NA;// кол-во парал. исполнителей
clEnqueueNDRangeKernel(cmndQ,
 kn5,1,NULL,gWS,0,0,NULL,&ev5);
clWaitForEvents(1,&ev5);
```

Необходимо также немного изменить запускаемую на 1-ой стадии процедуру ядра. Ведь теперь, после исполнения PA этапов нулевой стадии, работа 1-ой стадии должна начинаться не с 1-го этапа, а с этапа PA+1. Поэтому вместо одного параметра PB поместим в заголовок процедуры два параметра (P1 и P2), а в теле процедуры поменяем некоторые операторы, которые зависят от этих параметров (см. фрагменты, помеченные жирным шрифтом):

```
kernel void fft_btflyLoopEL
(int n,__global complex *X,
 __global complex *W, int P1, int P2,
 __local complex *XL ) {
uint G,R,WS,k,j,s,h,a,b,u;
uint g,i,kG,jG, kl,jl,al,bl;
i=get_local_id(0);// локальный номер исполнителя
WS=get_local_size(0);// размер раб.группы
g=get_group_id(0);// номер раб.группы
G=1<<(P1-1); R=n>>P1;
kG=0;jG=g*WS/G;
get_GRkjuabL(P1); a=k+j*h; b=a+s;
if(P1==P2){BTF(X[a],X[b],W[u]);}else{
BTFLY(X[a],X[b],W[u],XL[al],XL[bl]);
for(t=P1+1;t<=P2-1;t++){
 barrier(CLK_LOCAL_MEM_FENCE);
 get_GRkjuabL(t);
 BTF(XL[al],XL[bl],W[u]);
}//for t
 barrier(CLK_LOCAL_MEM_FENCE);
 get_GRkjuabL(P2); a=k+j*h;b=a+s;
 BTFLY(XL[al],XL[bl],W[u],X[a],X[b]);
}//if P1==P2
}//fft_btflyLoopEL
```

Потребуется сделать изменения и в OpenCL-программе. Сменим имя процедуры ядра:

```
kn3=clCreateKernel(prgrm,"fft_btflyLoopEL",NULL);
```

И фрагмент назначения ей параметров:

```
cl_uint PA1=PA+1;
clSetKernelArg(kn3,3,szI,&PA1);
clSetKernelArg(kn3,4,szI,&PB);
clSetKernelArg(kn3,5,szC*NB,NULL);
```

В результате получим вариант **E** OpenCL-программы. Проверим, удаётся ли таким вариантом ускорить операцию БПФ и насколько.

В таблице 7 представлены коэффициенты ускорения, полученные в результате прогонов этого варианта на платформах Intel и NVidia.

Таблица 7. Ускорение операции БПФ OpenCL-программой варианта **E** для платформ Intel и NVidia

P	N=2^P	Intel		NVidia	
		K2	K2^	K2	K2^
12	4096	28.68	1.12	67.67	1.09
13	8192	50.16	1.17	130.64	1.01
14	16384	69.25	1.19	228.76	1.13
15	32768	77.95	1.02	332.92	1.11
16	65536	73.17	0.98	472.52	1.12
17	131072	72.42	0.96	394.59	1.35
18	262144	53.79	0.80	367.04	1.46
19	524288	---	---	443.02	1.46
20	1048576	---	---	443.84	1.46
21	2097152	---	---	455.04	1.47
22	4194304	---	---	443.23	1.43

Как видно из этой таблицы, на платформе NVidia вариант E, увы, почему-то совсем не улучшает коэффициент «чистого» ускорения (K2) операции БПФ, полученный вариантом D. И на платформе Intel ожидаемого улучшения тоже не наблюдается. Казалось бы, почему?

А всё дело в том, что из-за введения нулевой стадии приходится сокращать количество этапов, обрабатываемых на 1-ой стадии (t=PA+1,..,PB). А охватить 1-ой стадией следующие PB этапов (t=PA+1,..,PA+PB) не удаётся по той причине, что процедура ядра fft_btflyLoopEL (как и fft_btflyLoopCL) настроена на обработку лишь непрерывного участка вектора длиной **NB**= 2^{PB} элементов.

Чтобы избавиться от этого ограничения, надо усовершенствовать процедуру ядра, используемую исполнителями рабочей группы. Требуется сделать её универсальной, чтобы она могла обрабатывать требуемое количество этапов (t=P1,..,P2) на любом произвольно заданном участке вектора, элементы которого следуют друг за другом с некоторым фиксированным и не обязательно единичным шагом.

9. Универсальная процедура ядра для операции БПФ

Представленная ниже процедура ядра (fft_btflyLoopUL) позволяет на любых этапах (заданными параметрами P1 и P2) исполнять операции БФ с парами элементов внутри своей рабочей группы при различных соотношениях между WS и R: как при условии WS≥R, так и при условии WS<R. Схема назначения пар элементов исполнителям и алгоритм вычисления параметров назначаемой пары, применённый в этой процедуре, поясняются на рисунке 3.

В первом случае (WS≥R, см. на рис. 3 слева) исполнителям одной рабочей группы предстоит совместно обрабатывать несколько групп элементов, каждая из которых сосредоточена в отдельном столбце воображаемой матрицы, размещённой на месте заданного вектора по строкам размером по **sG** элементов (**sG**=2^{P1-1}).

А во втором случае (WS<R, см. на рис. 3 справа) столбец каждой такой группы элементов разделяется на несколько (**L=R/WS**) частей для того, чтобы обработку каждой из них можно было осуществить отдельной рабочей группой исполнителей.

С такой процедурой можно разбить обработку всех P этапов 2-ой части операции БПФ на 4 стадии и выполнять её по следующему алгоритму. На 0-ой стадии вызывать процедуру fft_btflyLoopA для исполнения первых **PA** этапов. Далее на 1-ой и 2-й стадии вызывать процедуру fft_btflyLoopUL, чтобы исполнить следующие **PB** этапов и **PC** этапов для обработки участков из **NB**=2^{PB} и **NC**=2^{PC} элементов соответственно. А на заключительной стадии опять-таки циклически вызывать процедуру fft_btfly. Причём количественное распределение этих стадий можно было бы варьировать, т.е. задавать

значения **PA**, **PB**, **PC** (или **NA**, **NB**, NC) как параметры при запуске OpenCL-программы.

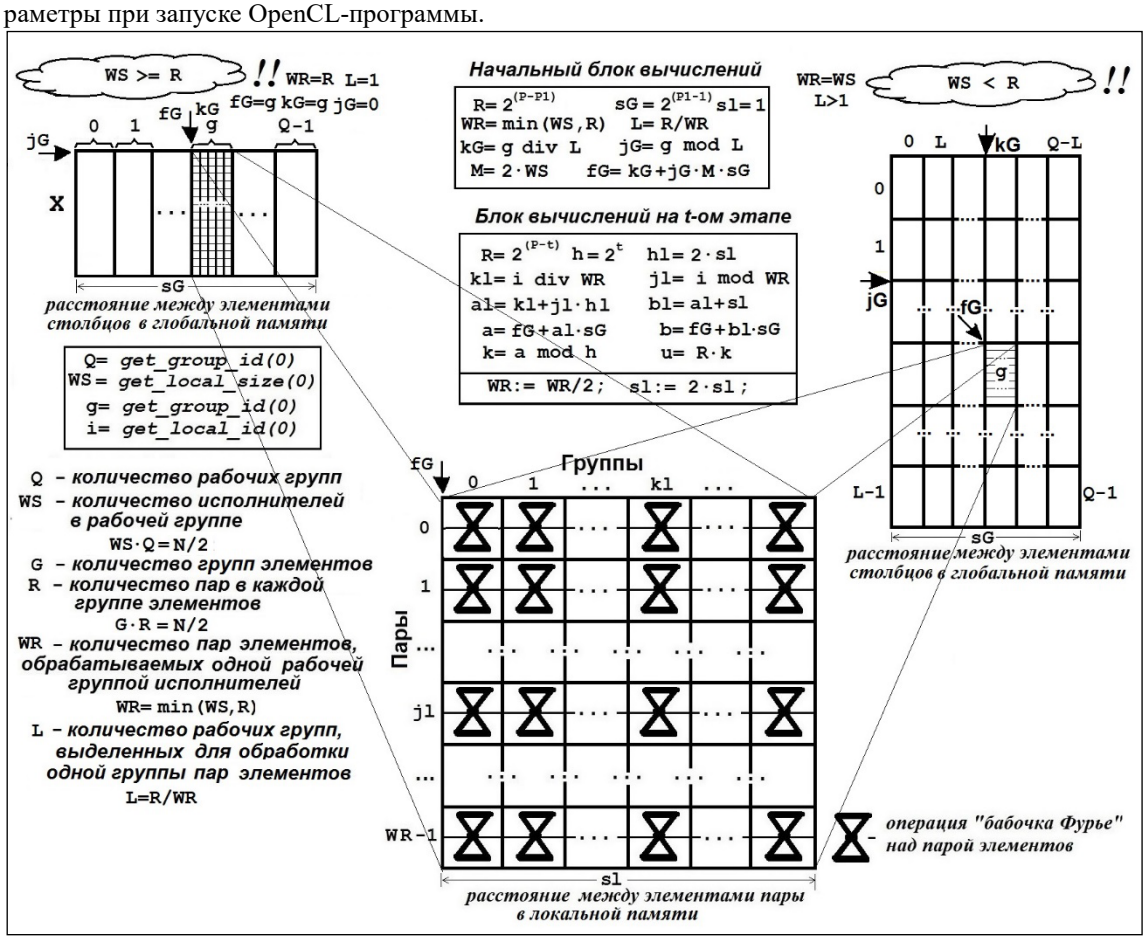

Рис. 3. Схема назначения пар в универсальной процедуре ядра 1-й и 2-й стадии варианта F

```
kernel void fft_btflyLoopUL
(int n,__global complex *X,
 __global complex *W, int P1,
 int P2, __local complex *XL ){
uint R,k,j,kl,jl,sl,h,a,b,al,bl,u;
uint g,i,WS,L,kG,jG,sG,fG;
i=get_local_id(0);// локальный номер исполнителя
WS=get_local_size(0);// размер раб.группы
g=get_group_id (0);// номер раб.группы
R=n>>P1; WR=MIN(WS,R);
L=R/WR;//L=к-во раб.групп для обработки X-группы
kG=g/L;jG=g%L; sG=1<<(P1-1);
fG=kG+jG*2*WS*sG;
sl=1; get_kjuabL(P1);
if(P1==P2){BTF(X[a],X[b],W[u]);}else{
BTFLY(X[a],X[b],W[u],XL[al],XL[bl]);
R=R>>1; WR=WR>>1; sl=2;
for(t=P1+1;t<=P2-1;t++){
 barrier(CLK_LOCAL_MEM_FENCE);
 get_kjuabL(t);
 BTF(XL[al],XL[bl],W[u]);
 R=R>>1; WR=WR>>1; sl=sl<<1;
}//for t
 barrier(CLK_LOCAL_MEM_FENCE);
 get_kjuabL(P2);
BTFLY(XL[al],XL[bl],W[u],X[a],X[b]);
}//if P1==P2
}//fft_btflyLoopUL
```

Для вычислений параметров пары элементов, назначаемой исполнителю на t-ом этапе, используется макроопределение **get_kjuabL**:

```
#define get_kjuabL(t) { \
 jl= i % WR; kl= i / WR; \
 al=kl+2*jl*sl; bl=al+sl; \
 a=fG+al*sG; b=fG+bl*sG; \
 k=a&((1<<t)-1);/*k=a mod h*/;u=R*k; }
```

Для осуществления такой обработки внесём необходимые изменения в OpenCL-программу:

```
kn3=clCreateKernel(prgrm,"fft_btflyLoopUL",NULL);
kn4=clCreateKernel(prgrm,"fft_btflyLoopUL",NULL);
```

Назначим процедуре ядра параметры её вызовов (в 4-м пункте действий функции clFFT):

```
cl_uint PB1=PA+1; cl_uint PB2=PA+PB;
cl_uint PC=Log2(NC);
clSetKernelArg(kn3,3,szI,&PB1);
clSetKernelArg(kn3,4,szI,&PB2);
clSetKernelArg(kn3,5,szC*NB,NULL);
cl_uint PC1=PB2+1; cl_uint PC2=PB2+PC;
clSetKernelArg(kn4,3,szI,&PC1);
clSetKernelArg(kn4,4,szI,&PC2);
```

```
clSetKernelArg(kn4,5,szC*NC,NULL);
```

И перепишем 5-й пункт списка действий функции clFFT, состоящий из 4-х стадий:

```
cl_uint t; cl_event ev2,ev3,ev4,ev5;
size t lWS[1]={NB/2}; gWS[0]=N/2;
{ 0-ая стадия } // t=1,..,PA; PA= log2(NA);
gWS[0]=N/NA;// кол-во парал. исполнителей
clEnqueueNDRangeKernel(cmndQ,
 kn5,1,NULL,gWS,0,0,NULL,&ev5);
clWaitForEvents(1,&ev5);
{ 1-ая стадия } // t=PA+1,..,PA+PB; PB= log2(NB);
clEnqueueNDRangeKernel(cmndQ,
 kn3,1,NULL,gWS,lWS,0,NULL,&ev3);
clWaitForEvents(1,&ev3);
{ 2-ая стадия } // t=PA+PB+1,..,PA+PB+PC;
lWS[0]= NC/2; // PC= log2(NC);
clEnqueueNDRangeKernel(cmndQ,
 kn4,1,NULL,gWS,lWS,0,NULL,&ev4);
clWaitForEvents(1,&ev4);
{ 3-ая стадия } // t=PA+PB+PC+1,..,P; P=log2(N);
for (t=PA+PB+PC+1; t<=P; t++) {
 clSetKernelArg(kn2,3,szI,&t);
 clEnqueueNDRangeKernel(cmndQ,
 kn2,1,NULL,gWS,NULL,0,NULL,&ev2);
 clWaitForEvents(1,&ev2);
}//for t
```

Полученный вариант OpenCL-программы обозначим как **F**. Обеспечиваемые им коэффициенты ускорения, полученные в результате прогонов на платформах Intel и NVidia, представлены в таблице 8.

Таблица 8. Ускорение операции БПФ OpenCL-программой варианта **F** для платформ Intel и NVidia

P	N=2^P	Intel		NVidia	
		K2	K2^	K2	K2^
8	256	2.15	1.11	7.55	1.64
9	512	4.80	1.23	12.48	1.82
10	1024	10.64	1.34	28.23	1.88
11	2048	20.41	1.42	55.81	1.75
12	4096	33.65	1.32	94.16	1.52
13	8192	49.87	1.17	169.69	1.31
14	16384	34.63	0.60	250.66	1.24
15	32768	48.03	0.63	275.00	0.92
16	65536	53.69	0.72	195.71	0.47
17	131072	58.22	0.77	197.15	0.67
18	262144	55.63	0.83	225.77	0.90
19	524288	49.45	0.95	319.08	1.05
20	1048576	**56.31**	**1.11**	358.84	1.46
21	2097152	52.56	1.11	395.74	1.28
22	4194304	**48.90**	**1.21**	415.87	1.34
23	2097152	**46.64**	**1.23**	**441.27**	**1.41**
24	4194304	**43.39**	**1.23**	**473.99**	**1.52**

Сравнивая их с коэффициентами предыдущих вариантов, можем сделать вывод, что этот вариант (**F**) предпочтительно применять лишь для векторов очень больших размеров: на платформе Intel он даёт выигрыш для векторов длиной $N \geq 2^{20}$, а на платформе NVidia − для векторов длиной $N \geq 2^{23}$.

10. Результаты оптимизации для платформ Intel и NVidia

Сравнив по производительности (по коэффициенту K2) все рассмотренные выше варианты OpenCL-программы выполнения операции БПФ на платформах Intel и NVidia, выберем для каждой длины самый оптимальный из вариантов и представим их все вместе в итоговых таблицах (см. таблицу 9 для платформы Intel и таблицу 10 для платформы NVidia).

Таблица 9. Результаты оптимизации OpenCL-программы операции БПФ для платформы Intel

P	N=2^P	V	K1	K2	K2^	K3	%K
8	256	$\mathbf{C^L}$	0.39	4.90	2.52	3.04	74.31
9	512	$\mathbf{C^L}$	0.87	10.05	**2.57**	**3.06**	75.78
10	1024	$\mathbf{C^L}$	1.16	17.90	2.25	2.56	79.94
11	2048	$\mathbf{C^L}$	2.18	29.76	2.07	2.29	82.55
12	4096	$\mathbf{C^L}$	3.74	45.93	1.80	1.95	84.03
13	8192	$\mathbf{C^L}$	6.22	64.29	1.51	1.58	87.13
14	16384	$\mathbf{C^L}$	11.38	78.70	1.35	1.40	87.58
15	32768	$\mathbf{C^L}$	17.83	**95.11**	1.25	1.28	**87.48**
16	65536	$\mathbf{C^L}$	30.43	85.08	1.14	1.18	70.99
17	131072	$\mathbf{C^L}$	40.91	83.57	1.11	1.14	72.26
18	262144	$\mathbf{C^L}$	45.39	72.70	1.08	1.12	65.03
19	524288	$\mathbf{C^L}$	41.71	54.94	1.06	1.10	**62.08**
20	104576	**F**	49.57	56.31	1.07	1.10	65.42
21	2097152	$\mathbf{C^L}$	47.02	52.95	1.12	1.19	69.22
22	4194304	**F**	45.12	48.90	1.21	1.31	69.51
23	8388608	**F**	44.24	46.64	**1.23**	1.34	67.85
24	16777216	**F**	41.52	43.39	**1.23**	1.36	64.35

Таблица 10. Результаты оптимизации OpenCL-программы операции БПФ для платформы NVidia

P	N=2^P	V	K1	K2	K2^	K3	%K
8	256	**D**	0.75	15.72	3.42	4.94	61.61
9	512	**D**	1.00	26.38	3.85	5.61	61.76
10	1024	**D**	2.00	56.74	3.78	5.15	66.76
11	2048	**D**	4.67	100.20	3.14	3.91	73.72
12	4096	$\mathbf{C^L}$	7.50	158.97	2.57	3.11	74.56
13	8192	$\mathbf{C^L}$	17.00	286.46	2.21	2.62	75.06
14	16384	$\mathbf{C^L}$	37.00	421.76	2.08	2.41	76.86
15	32768	$\mathbf{C^L}$	52.00	541.96	1.81	2.02	**79.70**
16	65536	$\mathbf{C^L}$	69.00	**666.94**	1.59	1.74	79.49
17	131072	**D**	150.00	545.52	1.86	2.35	73.79
18	262144	**D**	160.50	495.14	**1.96**	**2.72**	56.41
19	524288	**D**	209.00	**576.73**	1.90	2.66	**54.41**
20	1048576	**D**	222.00	566.46	1.87	2.53	55.42
21	2097152	**D**	240.88	562.78	1.82	2.36	60.01
22	4194304	**D**	**243.12**	521.69	1.68	2.05	65.00
23	8388608	**F**	235.79	441.27	1.41	1.57	72.08
24	16777216	**F**	257.80	473.99	**1.52**	1.73	71.58

Таким образом, применяя усовершенствованные варианты (B-F) OpenCL-программы, можно добиться более значительного ускорения операции БПФ в среде OpenCL. Как видно из приведённых таблиц, для векторов небольшой длины ($N \leq 2048$) удаётся увеличить коэффициент «чистого» ускорения (K2) примерно в 1,5-2

раза на платформе Intel и в 3-4 раза на платформе NVidia (см. столбец **K2^**). А для векторов очень большого размера ($N \geq 2^{22}$) – лишь примерно в 1,2 раза на платформе Intel и примерно в 1,5 раза на платформе NVidia.

Встаёт вопрос: а можно ли добиться ещё б**о**льшего ускорения? Чтобы на него ответить, вспомним, что операция БПФ складывается из двух частей: 1) бит-реверсной перестановки (копирования) элементов вектора; и 2) поэтапного цикла вычислений БФ («бабочек Фурье») над всевозможными парами элементов.

Все рассмотренные выше варианты (B-F) были нацелены лишь на ускорение 2-й части БПФ. Насколько удалось ускорить эту часть (по сравнению с вариантом A), как раз и показывает коэффициент **K3**, приведённый в таблицах 9-10. А значение из последнего столбца **(%K**) этих таблиц характеризует, какую долю (в процентах) эта часть занимает в общей длительности операции БПФ.

Чтобы прояснить, как влияет коэффициент **K3** и доля **%K** на итоговый коэффициент **K2**, решим следующую задачу.

Задача Операция **P** складывается из двух частей $\mathbf{P_1}$ и $\mathbf{P_2}$: $P=P_1+P_2$, для которых известны временн**ы**е доли их исполнения $\mathbf{d_1}$ и $\mathbf{d_2}$:

$$T=T_1+T_2,\ \mathbf{d_1}=T_1/T,\ \mathbf{d_2}=T_2/T.$$

Если ускорить исполнение части $\mathbf{P_1}$ в $\mathbf{u_1}$ раз, а части $\mathbf{P_2}$ — в $\mathbf{u_2}$ раз, то во сколько раз (**u**) ускорится исполнение всей операции **P**?

Решение Выразив новые длительности исполнения для P_1, P_2, P:
$T_1'=T_1/u_1=T\cdot d_1/u_1$, $T_2'=T_2/u_2=T\cdot d_2/u_2$, $T'=T_1'+T_2'$.
получим:
$T'=T\cdot d_1/u_1 + T\cdot d_2/u_2 = T\cdot(d_1\cdot u_2+ d_2\cdot u_1)/(u_1\cdot u_2)$.
Тогда коэффициент **u** ускорения операции P, равный отношению $u=T/T'$, должен вычисляться по формуле:

$$\mathbf{u= (u_1\cdot u_2) / (d_1\cdot u_2+ d_2\cdot u_1)} \quad \{\mathbf{Ф1}\}.$$

А в том случае, когда часть P_2 в новом исполнении исчезает совсем ($\mathbf{d_2}=0$ или $\mathbf{u_2}\to\infty$), вычисление коэффициента **u** можно упростить:
$T_2'=0$, $T'=T\cdot d_1/u_1$ и тогда $\mathbf{u= u_1/d_1}$ {**Ф2**}.
А при $\mathbf{u_1}=1$ это означает, что $\mathbf{u= 1/d_1}$ {**Ф3**}.

Вопрос: Какой можно ожидать итоговый выигрыш в ускорении всей операции, если добиваться ускорения лишь 2-й её части?

В качестве ответа на него приведём таблицу зависимости коэффициента ускорения **u** всей операции от коэффициента ускорения её 2-ой части $\mathbf{u_2}$ (при неизменном значении $\mathbf{u_1}=1$) для различных значений её доли $\mathbf{d_2}$ в общей операции (см. таблицу 11). В ней наглядно демонстрируется, что с ростом доли 1-ой части ($\mathbf{d_1}$) добиваться ускорения всей операции только за счёт ускорения её 2-й части становится всё проблематичней. И дальнейший рост коэффициента **u** упирается в «потолок»: его значение не сможет превысить 2.0 при $\mathbf{d_1}$=50%, 3⅓ − при $\mathbf{d_1}$=30%, 5.0 − при $\mathbf{d_1}$=20%, и 10.0 − при $\mathbf{d_1}$=10%.

Таблица 11. Зависимость общего коэффициента **u** от коэффициента $\mathbf{u_2}$ для разных значений $\mathbf{d_2}$ при $\mathbf{u_1}$=1

u= u_2 / ($d_1\cdot u_2+d_2$)					
u_2 \ d_1+d_2	**0,5+0,5**	**0,4+0,6**	**0,3+0,7**	**0,2+0,8**	**0,1+0,9**
2	1.33	1.43	1.54	1.66	1.82
3	1.5	1.67	1.875	2.14	2.5
4	1.6	1.82	2.11	2.5	3.08
5	1.67	1.92	2.27	2.77	3.57
6	1.71	2.0	2.4	3.0	4.0
7	1.75	2.06	2.5	3.18	4.375
8	1.77	2.11	2.58	3.33	4.71
9	1.8	2.14	2.65	3.46	5.0
10	1.82	2.17	2.7	3.57	5.26
20	1.9	2.33	2.985	4.16	6.9
50	1.96	2.43	3.185	27.78	8.47
100	1.98	2.46	3.26	4.8	9.17
$u_2\to\infty$	**2.0**	**2.5**	**3.33**	**5.0**	**10.0**

Анализируя таблицы 9 и 10, заметим, что после проведённой оптимизации 2-й части, доля 1-ой части БПФ становится весьма значительной: теперь она варьируется от 12.52% до 37.92% на платформе Intel и от 20.03% до 45.59% на платформе NVidia. А это значит, что дальнейшее ускорение операции БПФ в значительной степени зависит от того, удастся ли (и насколько!) ускорить 1-ю часть БПФ, т.е. бит-реверсную перестановку элементов вектора.

Но в среде параллельных исполнителей с глобальной памятью выполнить такую перестановку быстрее, увы, вряд ли возможно. Ведь в процедуре ядра fft_brvprst после вызова функции **k**=get_global_id(0) (для получения номера исполнителя) далее следуют лишь два оператора. Первый оператор m=BRT[k] получает значение бит-реверсного индекса из общей таблицы, а второй оператор X[m]=Y[k] просто копирует элемент одного вектора в другой. Каждый исполнитель при выполнении этой процедуры осуществляет 3 обращения в глобальную память. И сократить число таких обращений тут никак не получится.

Можно, конечно, отказаться от использования общей таблицы BRT и вычислять бит-реверсное значение в самой процедуре ядра (как это было описано, например, в [5,п.3.1]). Но это даст выигрыш лишь в том случае, когда такое вычисление будет осуществляться быстрее, чем чтение результирующего значения из общей таблицы, приготовленной в глобальной памяти. И такой вариант, вообще говоря, возможен, например, в случае наличия у процессорного ядра специальной команды, позволяющей для заданного целочисленного значения сразу же получить его бит-реверсное значение.

Но те платформы, на которых проводились

прогоны разработанных OpenCL-программ, такими процессорами с аппаратной поддержкой вычисления бит-реверсного значения, увы, не располагают. И поскольку ускорить 1-ую часть БПФ никак не удаётся, трудно рассчитывать на дальнейшее усовершенствование OpenCL-программы в целях ускорения операции БПФ.

11. Результаты оптимизации для платформы КОМДИВ

Проверим теперь, насколько позволяет применение описанных приёмов оптимизации повысить коэффициенты ускорения операции БПФ в среде OpenCL, обеспечиваемой в настоящее время микросхемой графического ускорителя семейства КОМДИВ. Все рассмотренные варианты (A-F) OpenCL-программы испытывались на аппаратной платформе, обеспечиваемой этой микросхемой, со следующими частотами: частота основного процессора (CPU) 400 Мгц, частота памяти 700 Мгц, частота системной шины 300 Мгц и частота графического процессора (GPU) 200 Мгц (Далее такую платформу будем сокращённо называть «платформа КОМДИВ».)

OpenCL-программа каждого варианта многократно прогонялась на такой платформе для векторов комплексных чисел (одинарной точности) различной длины $N=2^P$ (P=8,9,..,23). Полученные в результате этих прогонов усреднённые коэффициенты «чистого» ускорения (K2) операции БПФ различных вариантов представлены в таблице 12.

Таблица 12. Коэффициенты «чистого» ускорения (K2) операции БПФ различных вариантов OpenCL-программы, полученные на платформе КОМДИВ

P	$N=2^P$	A	C^L	D	E	F	K2!
8	256	0.07	**0.36**	0.35	0.25	0.25	**4.86**
9	512	0.15	0.53	0.49	0.41	**0.60**	**3.96**
10	1024	0.31	0.91	1.22	0.95	**1.31**	**4.21**
11	2048	0.63	1.51	2.25	1.87	**2.30**	**3.65**
12	4096	1.54	2.71	**3.75**	3.14	3.67	**2.44**
13	8192	3.6	4.77	**5.51**	4.93	4.55	**1.53**
14	16384	5.58	**6.40**	5.97	5.58	5.78	**1.15**
15	32768	7.81	**7.94**	6.59	6.31	6.99	**1.02**
16	65536	**10.47**	9.96	7.74	7.56	8.54	1.00
17	131072	**12.40**	11.54	---	---	8.99	1.00
18	262144	**12.63**	11.59	---	---	8.15	1.00
19	524288	**12.97**	11.87	---	---	8.42	1.00
20	1048576	**12.86**	11.79	---	---	8.62	1.00
21	2097152	**12.48**	11.48	---	---	8.97	1.00
22	4194304	**12.45**	11.54	---	---	9.38	1.00
23	8388608	**12.28**	10.91	---	---	8.97	1.00
K2! – наилучший коэффициент роста (K2^) среди всех вариантов = наибольший **K2** / **K2** варианта **A**							

Заметим, что на этой платформе разрешается объединять в рабочую группу не более 128 обрабатывающих элементов (MW=128), поэтому на ней вариант B можно применять только для векторов длиной не более 256 (2^8), а варианты D и E могут быть применены лишь для векторов длиной не более 65536 (2^{16}).

Как видно из этой таблицы, проведённая оптимизация позволила повысить коэффициент «чистого» ускорения (K2) операции БПФ на этой платформе лишь для векторов длиной не более 2^{15}(32768). Для векторов же б**о**льшего размера ($\geq 2^{16}$) такая оптимизация не дала ожидаемого эффекта, так что в итоге первоначально разработанный вариант OpenCL-программы оказался для них самым производительным.

Описанные приёмы оптимизации дают реальный практический эффект, когда память OpenCL-устройства действительно построена иерархически. Так что обращение к такой памяти для каждого обрабатывающего элемента подчиняется правилу: самая высокая скорость обеспечивается для доступа к своей внутренней памяти, затем, чуть помедленней – к локальной памяти его рабочей группы, а самая низкая – к глобальной общей памяти всего устройства.

Но микросхема графического ускорителя семейства КОМДИВ такую иерархию памяти OpenCL-устройства в настоящее время в полной мере не обеспечивает. Поэтому описанные приёмы оптимизации не получается эффективно применить (на этой платформе КОМДИВ) для обработки данных большого объёма.

12. Заключение

Полученный автором опыт разработки разнообразных вариантов программ, ускоряющих операцию быстрого преобразования Фурье в среде OpenCL, позволяет сделать следующий вывод. Для разработки эффективных программ по технологии OpenCL требуется не только изучить богатый арсенал её средств, но и освоить также приёмы оптимизации процедур ядра с учётом иерархического строения памяти устройств OpenCL среды.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения», Рег. № 122041100063-0.

Optimization of Fast Fourier Transform in OpenCL Environment

A.A. Burtsev

Abstract. The article focuses on the use of OpenCL technology, which allows to use powerful GPU resources to improve the performance of computing programs. Various variants of parallel programs designed to accelerate fast Fourier transform operations in an OpenCL environment are supposed. Considered variants are compared in terms of performance in order to determine the most optimal for processing complex vectors of different lengths.

Keywords: parallel programming, OpenCL technology, heterogeneous systems, microprocessors of the KOMDIV family, fast Fourier transform

Литература

1. В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. Спб., БХВ-Петербург, 2004.

2. Официальный сайт ФНЦ НИИСИ РАН. Разработка СБИС. Развитие микропроцессоров с архитектурой КОМДИВ, https://www.niisi.ru/devel.htm

3. Официальный OpenCL–сайт организации Khronos Group, http://www.khronos.org/opencl/

4. А.А. Бурцев. Оптимизация операции перемножения матриц на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, 100–112.

5. А.А. Бурцев. Ускорение быстрого преобразования Фурье на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 11 (2021), № 4, 27–37.

6. Стивен Смит. Цифровая обработка сигналов. Практическое руководство для инженеров и научных работников. М., Додека-XXI, 2012.

Составление многопроцессорного расписания в системе реального времени с ограничениями на связи между процессорами

М.Г. Фуругян[1]

[1]ФИЦ ИУ РАН, Москва, Россия, rtsccas@yandex.ru

Аннотация. Рассматривается задача построения допустимого расписания комплекса программных модулей в многопроцессорной системе реального времени с неполным графом связей между процессорами. Заданы длительности выполнения модулей и их директивные интервалы. При выполнении модулей допускаются прерывания и переключения с одного процессора на другой с учетом ограничений на связи между процессорами и на число переключений в фиксированный момент времени. Параллельное выполнение нескольких модулей одним процессором и одного модуля несколькими процессорами не допускается. Решение задачи основано на ее сведении к многопродуктовой потоковой задаче в сети.

Ключевые слова: многопроцессорная система, допустимое расписание, директивный интервал, многопродуктовый поток в сети, линейная система булевых соотношений

1. Введение

Публикации по вычислительным системам реального времени в нашей стране и за рубежом стали появляться в семидесятых годах прошлого века. В это же время эти системы начали разрабатываться и использоваться для нужд гражданского строительства и в военном деле. Наиболее широкое распространение системы реального времени получили в тех областях, где необходимо было решать задачи планирования вычислений при жестких временных ограничениях. Так, в конце семидесятых годов прошлого века руководством авиационной промышленности СССР перед предприятиями отрасли была поставлена задача сократить сроки испытания самолетов с пяти до полутора лет. Одним из элементов решения этой задачи являлось создание вычислительной системы реального времени для проведения летных экспериментов. Стояла задача обработки в режиме реального времени большого объема информации, поступающей с борта самолета с частотой 50 герц с тем, чтобы успевать передавать летчику корректирующие установки выполнения полета. Подобного рода задачи возникали и возникают в настоящее время и в других областях, как например, при разработке системы космической обороны, при решении сложных логистических задач, в гражданском и военном строительстве, при проектировании и эксплуатации объектов энергетики, в частности, атомных электростанций, при разработке систем экологического, экономического и технического мониторинга различных объектов, во многих других областях.

При создании многопроцессорных вычислительных систем реального времени необходимо решить широкий круг задач, связанных с построением такой структуры связей между процессорами и определением таких производительностей последних, которые обеспечивали бы выполнение определенных вычислений при строгих временных ограничениях. Кроме того, необходимо установить, в какие моменты времени и на каких устройствах нужно выполнять эти вычисления.

Одно из основных требований, предъявляемых к таким системам, заключается в том, что все вычисления должны производиться при заданных временных ограничениях. Для решения этой задачи могут быть использованы различные методы построения оптимальных расписаний [1 – 3]. В [4] рассмотрена задача оптимизации выполнения комплекса работ с нефиксированными длительностями, для решения которой применяется метод ветвей и границ и построении многогранников устойчивости решений. В [5 ─ 7] разработана методика проверки выполнения ограничений реального времени, заключающихся в том, что каждая работа должна выполняется в заданном директивном интервале. Проведенные исследования выполнены для многоядерной вычислительной системы реального времени и основаны на построении имитационной модели с использованием обобщенных конечных автоматов с остановкой таймера. С помощью этой модели строится временная диаграмма работы системы, позволяющая осуществить непосредственную проверку выполнения ограничений

реального времени. В [8] предложен псевдополиномиальный алгоритм решения задачи построения оптимального по быстродействию расписания исполнения заданий с логическими условиями предшествования. В этой задаче для каждого задания дан список его непосредственных предшественников, а также число завершенных непосредственных предшественников, необходимое для начала его выполнения. Задача сведена к циклической игре. В [9] для решения задач построения оптимальных расписаний с прерываниями в многопроцессорных системах реального времени предложен новый подход, основанный на модификации потоковых алгоритмов в сети. В [10] исследована задача минимизации времени выполнения комплекса непрерываемых заданий. Предложен алгоритм, сочетающий в себе жадную схему с процедурой ограниченного перебора, которая вызывается в случаях, когда жадный выбор не может привести к оптимальному решению.

В настоящей статье рассматривается многопроцессорная система реального времени с неполным графом связей между процессорами. Разработан алгоритм, позволяющий для заданной совокупности программных модулей, допускающих прерывания и переключения, с известными характеристиками (длительности и директивные интервалы) определить, существует ли допустимое расписание их выполнения и определить его в случае положительного ответа. Дополнительно задается ограничение на число прерываний и переключений в каждый момент времени. Алгоритм основан на сведении исходной задачи к многопродуктовой потоковой задачи в сети.

2. Постановка задачи

Имеется p идентичных процессоров, занумерованных от 1 до p, с помощью которых требуется выполнить m программных модулей, занумерованных от1 до m. Модули выполняются во временном отрезке [0; T], который разделен на T отрезков единичной длины $I_k = [t_{k-1};\ t_k]$, $k = \overline{1,T}$, $t_0 = 0$, $t_T = T$. Длительность выполнения i-го модуля составляет τ_i, а его директивный интервал $-$ $[b_i;\ f_i]$ (т.е. он может начать выполняться не ранее момента времени b_i и должен завершиться не позднее момента времени f_i), $i = \overline{1;m}$. Не допускается параллельное выполнение одного модуля несколькими процессорами. В каждом интервале I_k, $k = \overline{1,T}$, одним процессором может выполняться не более одного модуля.

Между процессорами существует связь, позволяющая переключать выполнение модулей с одного процессора на другой. Эта связь задается двумерным массивом $C(j_1;\ j_2)$, $j_1, j_2 = \overline{1,p}$, элементы которого принимают значения 0 или 1. При этом $C(j_1;\ j_2) = 1$, если при выполнении модуля процессором j_1 в некотором интервале I_{k_1} допускается переключение этого модуля в момент времени t_{k_1} на процессор j_2, которым он будет выполняться в некотором интервале I_{k_2} $(k_1 < k_2)$.

Такое переключение модуля требует суммарных временных затрат процессоров j_1 и j_2, равных d. Число переключений в каждый момент времени t_k не должно превосходить заданного числа r_k, $k = \overline{1,T-1}$. Предполагается, что $\tau_i, T, d \in N$, $b_i, f_i \epsilon \{t_0, t_1, \dots, t_T\}$, $i = \overline{1;m}$, $N-$ множество натуральных чисел.

Требуется определить, существует ли допустимое расписание выполнения m программных модулей и найти его в случае положительного ответа. Допустимое расписание – это такое расписание, при котором:

1) суммарное время выполнение процессорами каждого модуля $i = \overline{1;\ m}$ (за вычетом временных издержек на переключения) составляет τ_i;

2) все переключения выполняются в соответствии с массивом C;

3) не нарушаются ограничения по числу переключений в моменты t_k, $k = \overline{1,T-1}$;

4) каждый модуль $i = \overline{1;\ m}$ выполняется в своим директивном интервале $[b_i;\ f_i]$.

Как известно [1], такая задача является NP-трудной. Мы сведем ее решение к многопродуктовой потоковой задаче с дополнительными ограничениями.

3. Алгоритм решения задачи

Построим m-продуктовую потоковую сеть $H = (V, A)$ (см. рисунок), где V – узлы, A – ориентированные дуги. Множество V состоит из следующих узлов:

s_i, v_i – соответственно источник и сток i-го продукта;

x_{kj}, y_{kj} – соответственно начало и окончание выполнения некоторого модуля в интервале I_k процессором j.

Множество A состоит из следующих дуг:

(s_i, x_{kj}) – начало выполнения модуля i процессором j в интервале I_k, $i = \overline{1;m}$, $j = \overline{1;p}$, $k = \overline{b_i, f_i}$;

$(x_{kj},\ y_{kj})$ – выполнение некоторого модуля процессором j в интервале I_k, $k = \overline{1,T}$, $j = \overline{1;p}$;

(y_{kj}, v_i) – завершение выполнения модуля i процессором j в интервале I_k, $i = \overline{1;m}$, $j = \overline{1;p}$, $k = \overline{b_i, f_i}$;

$(y_{k_1 j_1}, x_{k_2 j_2})$ – переключение выполнения некоторого модуля с процессора j_1 в момент времени t_{k_1} на процессор j_2 в момент времени t_{k_2}, $j_1, j_2 = \overline{1;p}$, $C(j_1;\ j_2) = 1$, $k_1, k_2 = \overline{1;T}$, $k_1 < k_2$. Отметим, что при $j_1 = j_2$, $k_1 = k_2$ переключения не происходит и временные издержки отсутствуют. Издержки на переключение происходят только при $j_1 \neq j_2$.

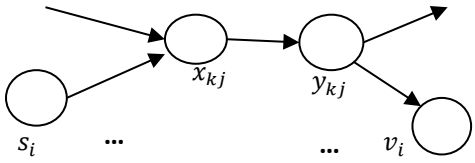

Рисунок. Фрагмент потоковой сети *H*

Пропускная способность $U(a,b)$ каждой дуги сети H полагается равной единице: $U(a,b) = 1$, $(a,b) \in A$.

В сети H будем рассматривать m-продуктовый поток $g_i(a,b)$, $i = \overline{1;m}$, $(a,b) \in A$, принимающий значения 0 или 1 и обладающий следующими свойствами:

$$\sum_{k=b_i}^{f_i} \sum_{j=1}^{p} g_i(s_i, x_{kj}) = 1\,, i = \overline{1;m}, \qquad (1)$$

т.е. из источника s_i исходит ровно одна единица *i*-го продукта;

$$\sum_{k=b_i}^{f_i} \sum_{j=1}^{p} g_i(y_{kj}, v_i) = 1\,, i = \overline{1;m}, \qquad (2)$$

т.е. в сток v_i входит ровно одна единица *i*-го продукта.

Единицу потока *i*-го продукта по дуге (s_i, x_{kj}) будем рассматривать как начало выполнения модуля *i* в момент времени t_{k-1} процессором *j*. Далее, модуль *i* выполняется процессором *j* в интервале I_k.

Единицу потока *i*-го продукта по дуге (y_{kj}, v_i) будем рассматривать как завершение выполнения модуля *i* в момент времени t_k.

Единица потока *i*-го продукта по дуге (x_{kj}, y_{kj}) означает выполнение *i*-го модуля в интервале I_k процессором *j*, в результате чего к суммарному времени его выполнения прибавляется единица.

И наконец, единица потока *i*-го продукта по дуге $(y_{k_1 j_1}, x_{k_2 j_2})$ означает переключение выполнения *i*-го модуля с процессора j_1 в момент времени t_{k_1} на процессор j_2 в момент времени t_{k_2}, в результате чего из суммарного времени его выполнения вычитается величина *d*.

Таким образом, к соотношениям (1), (2) следует добавить следующие соотношения:

$$\sum_{i=1}^{m} g_i\,(x_{kj}, y_{kj}) \leq 1, k = \overline{1,T}, j = \overline{1;p}, \qquad (3)$$

т.е. в каждом интервале $\boldsymbol{I_k}$ одним процессором выполняется не более одного модуля;

$$\sum_{k=b_i}^{f_i} \sum_{j=1}^{p} \boldsymbol{g_i(x_{kj}, y_{kj})} - \sum_{k_1,\, k_2 = \overline{b_i, f_i}, k_1 < k_2} \ \sum_{j_1,\, j_2 = \overline{1;p}, C(j_1;j_2)=1} \boldsymbol{g_i\,(y_{k_1 j_1}, x_{k_2 j_2})} = \boldsymbol{\tau_i},\ \boldsymbol{i = \overline{1,m}}, \qquad (4)$$

т.е. суммарная длительность выполнения модуля *i* за вычетом затрат на переключения составляет $\boldsymbol{\tau_i}$;

$$\boldsymbol{g_i(s_i, x_{kj})} + \sum_{k_1 = \overline{b_i, f_i}, k_1 < k} \ \sum_{j_1 = \overline{1;p}, j_1 \neq j, C(j_1;j)=1} \boldsymbol{g_i\,(y_{k_1 j_1}, x_{kj})} = \boldsymbol{g_i(x_{kj}, y_{kj}), j = \overline{1,p}, k = \overline{b_i, f_i} -} \qquad (5)$$

сохранение потока *i*-го продукта в узлах $\boldsymbol{x_{kj}}$;

$$\boldsymbol{g_i(x_{kj}, y_{kj})} = \sum_{k_1 = \overline{b_i, f_i}, k_1 > k} \ \sum_{j_1 = \overline{1;p}, j_1 \neq j, C(j;j_1)=1} \boldsymbol{g_i\,(y_{kj}, x_{k_1 j_1})} + \boldsymbol{g_i(y_{kj}, v_i), j = \overline{1,p}, k = \overline{b_i, f_i} -} \qquad (6)$$

сохранение потока *i*-го продукта в узлах$\boldsymbol{y_{kj}}$,

$$\boldsymbol{g_i(a,b) = 0 \vee 1,\ i = \overline{1;m},\ (a,b) \in A.} \qquad (7)$$

Таким образом, исходная задача о составлении допустимого расписания сведена к системе булевых ограничений (1) – (7), в которой $O(mpT^2)$ переменных $g_i(a,b)$, $i = \overline{1;m}$, $(a,b) \in A$, и $O(m + pT)$ линейных соотношений. А именно, допустимое расписание в исходной задаче существует в том и только том случае, когда система (1) – (7) имеет решение. Если $g_i(x_{kj}, y_{kj}) = 1$, то модуль *i* выполняется процессором *j* в интервале I_k. Если $g_i(y_{k_1 j_1}, x_{k_2 j_2}) = 1$, то модуль *i* после выполнения процессором j_1 в интервале I_{k_1} переключается на процессор j_2, которым он будет выполняться в интервале I_{k_2}.

6. Заключение

В данной работе рассмотрена задача построения допустимого расписания комплекса программных модулей в многопроцессорной системе с идентичными процессорами и неполным графом связей между ними. Считаются заданными длительности выполнения модулей и их директивные интервалы. При выполнении модулей допускаются прерывания и переключения с одного процессора на другой с учетом ограничений на связи между процессорами и на число переключений в фиксированный момент времени. Учитываются временные издержки при выполнении переключений. Параллельное выполнение нескольких модулей одним процессором и одного модуля несколькими процессорами не допускается. Задача сведена к линейной системе булевых соотношений. Решение основано на использовании теории многопродуктовых потоков в сети.

Creation of a Multiprocessor Schedules in Real-Time System with Communication Restrictions Between Processors

Meran Furugyan

Abstract. The problem of constructing an admissible schedule for a complex of program modules in a multiprocessor real-time system with an incomplete graph of connections between processors is considered. The duration of the modules execution and their directive intervals are set. When executing modules, interrupts and switching from one processor to another are allowed, taking into account the restrictions on communications between processors and on the number of switches at a fixed point in time. Parallel execution of several modules by one processor and one module by several processors is not allowed. The solution of the problem is based on its reduction to a multicommodity flow problem in the network.

Keywords: multiprocessor system, admissible schedule, directive interval, multiproduct flow in the network, linear system of Boolean relations

Литература

1. В.С. Танаев, В.С. Гордон,Я.М. Шафранский. Теория расписаний. Одностадийные системы. М.: Наука, 1984, 383 с.

2. P. Brucker. Scheduling Algorithms. Heidelberg: Springer, 2007, 378 с.

3. А.А. Лазарев. Теория расписаний. Оценка абсолютной погрешности и схема приближенного решения задач теории расписаний. М.: МФТИ, 2008, 222 с.

4. А.В. Мищенко, П.С. Кошелев. Оптимизация управления работами логистического проекта в условиях неопределенности. «Изв. РАН. ТиСУ», (2021), № 4, 86 – 101.

5. А.Б. Глонина, В.В. Балашов. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов. «Моделирование и анализ информационных систем», Т. 25 (2018), № 2, 174 – 192.

6. А.Б. Глонина. Обобщенная модель функционирования модульных вычислительных систем реального времени для проверки допустимости конфигураций таких систем. «Вестник ЮУрГУ. Сер. Вычисл. математика и информатика», Т. 6 (2017), № 4, 43 – 59.

7. А.Б. Глонина. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем. «Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика», (2020), № 3, 16 – 29.

8. Д.В. Алифанов, В.Н. Лебедев, В.И. Цурков. Оптимизация расписаний с логическими условиями предшествования. «Изв. РАН. ТиСУ», (2009), № 6, 88 – 93.

9. В.А. Костенко, А.С. Смирнов. Алгоритм построения однопроцессорных статико-динамическихрасписаний. «Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика», (2018), № 1, 45 - 52

10. В.А. Костенко. Алгоритмы комбинаторной оптимизации, сочетающие жадные стратегии и ограниченный перебор. «Изв. РАН. ТиСУ», (2017), № 2, 48 - 56.

Случайные тесты с перебором классов инструкций

А.С. Куцаев[1]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, koutsaev@niisi.msk.ru, +7 916 174-61-16

Аннотация. Преимущество генераторов случайных тестов в простоте применения и возможности выдачи больших объемов кода. Стохастическое тестирование выявляет как обычные, так и специфические ошибки (редкие сочетания и недочеты спецификации). При этом увеличение покрытия теста не зависит прямо от управления генератором. Перебор размещений из *n* классов инструкций по *k* позволяет получить покрытие по соответствующим комбинациям инструкций при приемлемых размерах кода.

Ключевые слова: случайные тесты, покрытие, верификация, перебор классов инструкций

1. Введение

Стохастическое тестирование является существенной составляющей средств функциональной верификации [1], [2]. Оно позволяет выявить ошибки проекта, которые трудно обнаружить другими способами. Преимущество стохастического тестирования в возможности получать тестовый код большого объема. При достаточном разнообразии кода это позволяет вырабатывать тестовые ситуации, которые трудно предусмотреть. Однако большой объем теста не гарантирует эффективности: например, может оказаться, что некоторые сочетания инструкций не возникают вообще. Необходимо, чтобы с увеличением объема росло и покрытие теста, определенное каким-либо способом. Эффективность теста зависит от большого числа параметров, выбор которых рассматривается в данной работе.

В генераторах семейства tergen, разработанных во ФГУ ФНЦ НИИСИ РАН [3], [4], используется два основных подхода к выработке тестовых ситуаций:

- Массовая выдача случайных инструкций со случайными аргументами. Для нее задаются только режимы генерации и вероятности выбора.
- Создание тестовых ситуаций на основе описания, полученного при анализе ранее возникших проблем. Возможно частичное и полное задание фрагментов кода наряду со случайным выбором.

Исходные данные для генерации теста – это шаблон, который включает задание управляющих параметров и программу построения теста (заготовку тестового кода) на интерпретируемом языке. В предыдущей работе [5] подробно описаны возможности некоторых средств генератора тестов для создания специальных тестовых ситуаций. Цель настоящей работы – описание методики создания тестов большого объема. Эта задача определяет требования как к программе построения теста, так и к управляющим параметрам.

В главе 2 обсуждаются вопросы получения покрытия при различных вариантах перебора. В главе 3 приводится пример классификации инструкций и возможные нарушения порядка классов при переборе. Глава 4 посвящена возможностям настройки генератора для инструкций определенных классов. В главе 5 приведены выводы.

Далее T32 и T64 – это обозначения версий генератора тестов для 32- и 64-битового процессора, соответственно.

2. О переборе и покрытии

Покрытие теста можно определять разными способами. При случайном тестировании сложно указать зависимость величины покрытия от управляющих параметров генератора тестов. Для выбора этих параметров можно дать лишь самые примитивные рекомендации. Например, инструкции могут выбираться случайно, независимо друг от друга, согласно заданным вероятностям выбора. Чтобы тестировать инструкции, связанные с некоторой частью проекта, можно увеличить вероятности выбора этих инструкций, однако говорить о связи с каким-либо покрытием при этом нельзя.

В случае перебора по комбинациям инструкций покрытие по ним обеспечивается автоматически. Например, при тестировании используются переборы всех пар инструкций "каждая за каждой", всех троек и т.п. Такой подход оправдан, поскольку некоторые ошибки выявляются при определенном сочетании двух, трех и более инструкций. Однако, когда число инструкций процессора составляет несколько сотен, даже од-

нократный перебор всех допустимых троек инструкций дает значительный объем кода. При этом однократного перебора недостаточно минимум по двум причинам: неизбежные нарушения порядка следования инструкций, а также произвольность выбора их аргументов. Это делает перебор, скажем, всех четверок инструкций мало реальным.

Способ уменьшить объем кода при переборе за счет введения дополнительной случайности состоит в следующем. Все инструкции объединяются в классы по сходству тех или иных признаков. Например, класс может включать инструкции, использующие устройство целого умножения-деления. Затем вместо перебора самих инструкций делается перебор их классов, в ходе которого из очередного класса случайно выбирается одна инструкция. Такой способ делает реальным неоднократный перебор всех троек и четверок классов. Предполагается, что индивидуальные особенности отдельных инструкций в классе мало влияют на взаимовлияние их выполнения. Главная идея перебора - создание тестовых ситуаций за счет сочетания инструкций разного вида – сохраняется.

На первый взгляд порядок перебора неважен, лишь бы он был полным. Однако это не совсем так. Равномерность распределения инструкций каждого класса по последовательности также имеет значение. Например, если большинство регистровых переходов собрано в одной части кода, инструкции загрузки целевых адресов будут сильнее искажать результат перебора.

Рассмотрим этот вопрос подробнее. Пусть есть множество из n объектов (классов инструкций). Для наглядности будем представлять классы их номерами от 0 до n-1. Рассмотрим перебор, т.е. последовательность всех размещений с повторением из n по k, где k - целое, от 2 и выше (случай k=1 допустим, но интереса не представляет). Число размещений в этой последовательности $N = n^k$. Для размещения $\{i_0, i_1, \dots i_{k-1}\}$ введем номер i, задаваемый выражением

$$i = i_0 n^{k-1} + i_1 n^{k-2} + \cdots + i_{k-1} \qquad (1)$$

Последовательность всех размещений из n классов по k, упорядоченную по возрастанию номеров, объявим основной. Раскрывая скобки, перейдем к последовательности классов, длина которой $k \times N$. Если ее зациклить, можно заметить, что каждая подпоследовательность длины k встречается в ней ровно k раз (далее для краткости будем называть это свойством k вхождений). Есть и более сложные способы перебора, например, последовательность де Брёйна, в которой каждая подпоследовательность длины k встречается ровно один раз. Однако при случайном тестировании, где требуется именно неоднократное повторение комбинаций, такое усложнение лишено смысла.

В основной последовательности классы распределены неравномерно. Рассмотрим, например, размещения из 4 классов по 2, т.е. все пары классов с номерами 0, 1, 2, 3.

0,0, 0,1, 0,2, 0,3, 1,0, 1,1, 1,2, 1,3,
2,0, 2,1, 2,2, 2,3, 3,0, 3,1, 3,2, 3,3,...

В каждой четверти этой последовательности один из классов присутствует 5 раз, а остальные - по одному. Желательно иметь более равномерное распределение классов, но с сохранением "свойства k вхождений".

Зададим перестановку основной последовательности в виде $j = j\,(i)$,

$$j(i) = (i \times M)_{mod\ N}. \qquad (2)$$

где i и j - порядковые номера размещений в последовательности до и после перестановки, а M - любое число, взаимно простое с N (или с n, что то же). Легко показать, что обратная перестановка имеет тот же вид:

$$i(j) = (j \times L)_{mod\ N}. \qquad (3)$$

где L находится из соотношения $(L \times M)_{mod\ N} = 1$. Непосредственная проверка для значений n и k, представляющих интерес, показывает, что такие перестановки сохраняют "свойство k вхождений".

Оценка равномерности распределения номеров для разных M делается стандартно. Выберем номер класса i, разобьем последовательность на равные части, найдем число номеров i в каждой части и оценим среднеквадратичное отклонение (СКО). Оказывается, что у основной последовательности ($M = 1$) оценка СКО максимальна для любого номера, т.е. это самая неравномерная последовательность.

Если число частей разбиения равно числу классов n, то при $k > 1$ и $M = n^2 + 1$ оценка СКО нулевая: в каждой части число вхождений выбранного номера одно и то же. Однако если разбивать последовательность на другое число частей, оценки СКО сильно разнятся и не позволяют выбрать наилучшее M. Одновременно можно заметить, что при указанном выборе M результат содержит довольно сильную регулярность. Например, начало последовательности размещений из 5 по 3 после перестановки с $M = 5^2 + 1 = 26$ имеет вид:

{0,0,0),(1,0,1},(2,0,2},{3,0,3},{4,0,4},...

Далее первый и третий номера в размещениях начинают различаться, но закономерность изменения остается очевидной.

При анализе СКО с различными числами от-

резков для $k \le 4$ оказывается, что не худшие результаты для M дают полиномы от n степени k - 1, в которых коэффициенты преимущественно разные, а свободный член равен 1 либо n - 1. Последовательность номеров видимой регулярности не проявляет.

Количество инструкций, получаемое за один перебор, не слишком велико. Например, для размещений из 5 классов по 4 получим последовательность из 625 размещений, что дает 2500 инструкций. В случайном тесте перебор можно повторить несколько раз, при этом число вхождений каждой подпоследовательности длины 4 будет расти равномерно.

3. Классы инструкций MIPS32

Для инструкций MIPS32, исключая сопроцессор плавающей точки (FPU), можно рассмотреть следующие классы:

- Загрузка-сохранение.
- Вычисления: арифметика, сдвиги, логика, битовые перестановки.
- Целочисленное умножение-деление (длительные операции).
- Переходы: условные и безусловные.
- Пересылки.
- Некоторые специальные инструкции.

Здесь можно заметить, что не все сочетания допустимы. Два перехода подряд запрещены, а два умножения-деления приведут к большой задержке, не говоря уже о возможной потере результатов. При задании классификации можно пометить отдельные классы как не допускающие повторов. При переборе из двух вхождений такого класса останется только первое.

Для задания классов в программе шаблона используются переменные типа tree. Такие же переменные используются и для случайного выбора инструкций согласно заданным весам, но здесь каждое поддерево верхнего уровня задает отдельный класс. По умолчанию веса поддеревьев нулевые. Ненулевой вес поддерева верхнего уровня означает, что данный класс не допускает повторов при переборе. Более того, если задать один и тот же ненулевой вес для двух классов, то при удалении повторов эти два класса будут считаться одним. Ниже показан пример задания классов инструкций в шаблоне (для экономии места часть инструкций не включена).

```
tree tt_alu = {
 Lost {
  Align=20 {lb=5, lw=5, sb=5, sw=5},
  NAlig=20 {lwl=5, lwr=5, swl=5, swr=5}},
 Calcul {
  AImm=20 {addi=5, andi=5, ori=5},
  ABin=20 {add=5, sub=5, and=5, or=5},
  AExt=20 { slt=5, sltu=5, clo=5, clz=5 } },
 MulDiv=1  {
  MD1=20 {div=5, mult=5, mul=5 },
  MAS=20 {madd=5, msub=5 } },
 BrJmp=2  {
  Jumps=20 { jr=5, jalr=5, j=5, jal=5},
  Bran=20 { beq=5, bne=5, bgez=5, bltz=5},
  Likely=20 {beql=5, bgezl=5, bltzl=5 } },
 Move=3 {movf=5, movt=5, movn=5, movz=5 }
};
```

Это дерево имеет 5 поддеревьев верхнего уровня. Два из них имеют веса по умолчанию (0), т.е. при переборе нет ограничений на соседство инструкций, выбранных из этих поддеревьев. Три поддерева имеют ненулевые веса. Это значит, что соседство инструкций из одного и того же поддерева не допускается. Для поддеревьев BrJmp и MulDiv смысл запрета обсуждался выше, а для поддерева Move это просто исключение повторений несложных инструкций.

Последовательность инструкций (классов) может также нарушаться из-за следующих действий и приемов, используемых при генерации случайных тестов.

- Предзагрузка. Это прием, который отделяет инструкции загрузки адреса в регистр от использования этого регистра. Предзагрузка позволяет исключить предопределенные комбинации с участием инструкций, использующих загружаемые регистры. Место для инструкций предзагрузки выбирается случайно либо задается, а сами инструкции (T32) или данные для них (T64) формируются позже, после вычисления содержимого регистра.
- Перезагрузка. Если регистровые переходы происходят слишком часто, предзагрузка не успевает их подготовить и регистр перезагружается непосредственно перед переходом. То же делается при наработке регистров баз адресов данных, поэтому стоит выполнить ее заранее, до перебора.
- Исключения ALU и/или FPU. Инструкции, их вызывающие, могут добавляться случайно для усложнения условий выполнения. Вместе с тем они искажают порядок комбинаций классов.
- Разделители. Между некоторыми инструкциями требуется вставка 1-2 разделяющих инструкций в силу особенностей реализации процессора. Генератор вставляет разделители автоматически. Так, чтение регистра hi или lo и запись в них разделяются двумя инструкциями (T32).

- Загрузка условий перехода. При генерации условия перехода должны быть известны. Это может приводить к случайному выбору условия и перезагрузке регистра непосредственно перед переходом.
- Самопроверка. При массовой выдаче инструкций вызов процедуры самопроверки вставляется в код, как только наберется достаточное число измененных регистров.

С учетом нарушений последовательности понятно, что перебор классов инструкций в случайном тесте нужно повторить несколько раз, чтобы дать больше возможностей для реализации всех комбинаций классов.

Инструкции FPU при классификации можно разделить на инструкции одинарной и двойной точности. Это вызвано требованием архитектуры к совместимости по формату выходных и входных регистров. Данное требование является естественным при программировании, однако в случайных тестах его требуется обеспечивать отдельно. В действительности, когда содержимое регистра записано в регистровый файл, оно может интерпретироваться в любом формате. То же относится к результату пересылки или загрузки из памяти, когда формат результата не определен. В остальных случаях, таких как передача данных через bypass, лучше избегать смешивания форматов FPU. Классы инструкций одинарной либо двойной точности можно использовать совместно с классами ALU (см. выше).

4. Настройки и программа

Ниже даны некоторые рекомендации по выбору параметров настройки и пример программы шаблона. В составных именах параметров настройки имена подразделов разделены двоеточием.

4.1. Регистровые аргументы

Общий выбор регистров используется там, где не нужен (или не дал результата) учет специфики инструкций, таких как обращения к памяти и переходы. Традиционным приемом для создания тестовых ситуаций здесь является выбор недавно использованных регистров. Это может создавать зависимости по данным в конвейере. Вероятности (веса) заданы в подразделе Register раздела :Settings.

При случайном выборе регистра может последовательно применяться несколько попыток. Сначала случайно выбирается вариант возможного использования регистра, записанного предыдущей инструкцией: на чтение, на запись либо никак. Если условия позволяют, выбор принимается. Если выбор принят, перед текущей инструкцией может быть вставлена еще одна, заведомо вызывающая исключительную ситуацию ALU или FPU по заданным вероятностям LastRegExcALU или LastRegExcFPU, соответственно.

Вторая попытка основана на т.н. возрасте использования регистра. Это число тактов, проходящих от предыдущего использования на чтение или на запись до возможного использования в текущей инструкции. Если возраст использования попадает в заданный интервал, регистр считается использованным, иначе - новым. Все регистры, допустимые для данного аргумента, делятся на новые или использованные на чтение и/или запись - всего 4 подмножества, одно из которых выбирается случайно, а в нем выбирается произвольный регистр. Если выбранное подмножество пусто, выбирается произвольный регистр из всех допустимых. Такие случаи нередки, так как инструкции имеют разное время выполнения, и настроить границы возраста использования на все сразу нельзя.

4.2. Инструкции перехода

Для условных переходов генератор позволяет случайно выбирать по весам условие перехода "истина" или "ложь". Однако если содержимое регистра условия не соответствует выбору, генератор добавляет в код инструкции перезагрузки регистра, лишние с точки зрения задачи тестирования. Случайное содержимое регистров условия примерно отвечает вероятностям выбора 50:50. Если это допустимо, то можно отменить обязательную перезагрузку регистров. Она останется только для случая, когда содержимое регистра условия неизвестно.

Для выбора целевого адреса обычно достаточно задать выравнивание адреса от 8 до 64 байт. Если в шаблоне описана одна область кода, больше ничего не нужно. Иначе предусмотрен случайный выбор перехода либо в ту же самую область (близкий переход), либо в любую другую. Для близких переходов задаются также допустимые границы смещения адреса.

Как уже отмечалось, для регистровых переходов предзагрузка регистра позволяет высвободить место перед переходом для любых допустимых инструкций. Для больших тестов удобно использовать случайную предзагрузку, которая вставляется автоматически. Случайная предзагрузка настраивается в подразделе Preload_Insertion. Очередная предзагрузка вставляется через N инструкций после места использования предыдущей, где N выбирается каждый раз случайно. Чем выше частота выдачи регистровых переходов, тем меньше должно быть N. Если предзагрузка не успевает готовить реги-

стры, нужно снизить частоту выдачи (вес) регистровых переходов по сравнению с переходами с другой адресацией.

4.3. Загрузка и сохранение данных

Эти инструкции используют адресацию база+смещение (T32), а также база+индекс (T64). Повторное использование регистров адреса снижает число дополнительных инструкций перезагрузки. Чтобы содержимое этих регистров не затиралось случайно, используется режим сохранения баз (установка Tergen:KeepBaseRegs ≥ 1). В этом режиме для операции загрузки-сохранения сначала случайно выбирается адрес данных, а затем подбираются регистры, содержимое которых позволяет получить этот адрес. Если таких регистров не меньше KeepBaseRegs, регистр адреса выбирается из них случайно. Иначе выбирается еще один регистр, в который загружается подходящая база. Этот регистр становится защищенным от случайной записи. Такая техника выбора регистров часто позволяет обойтись вообще без загрузки баз, за исключением начального участка теста. Условием для этого служит подходящее распределение памяти: если областей памяти много, и для каждой требуется отдельная база, регистров может не хватить даже при минимальных требованиях (KeepBaseRegs=1).

4.4. Самопроверка

При самопроверке вызов процедуры сравнения в шаблоне возможен как явно, при помощи функции check, так и по счетчику готовых регистров. Когда один оператор генерирует много инструкций, функция check не подходит, остается только вариант со счетчиком. Установка Tergen:AutoTest задает количество измененных регистров с известным содержимым, при котором добавляется вызов процедуры сравнения. Обычно задается значение от 8 до 10: при меньшем значении растет число вызовов и накладные расходы памяти, при большем растет риск пропустить ошибку из-за повторной записи в тот же регистр.

Самопроверка связана также с двумя областями памяти. В одну (область кадров) записываются данные о содержимом регистров на стадии генерации кода для последующего сравнения, в другую (системная область) - информация о найденной ошибке. Размер области кадров нужно задавать с запасом, но обычно она занимает примерно столько же, сколько код.

4.5. Программа

Для перебора n классов инструкций служит оператор xgroup. Его параметры - дерево инструкций, число k и количество инструкций, которые нужно выработать. Число k - это число элементов в размещениях из n классов по k, $\{i_0, i_1, \ldots i_{k-1}\}$. По умолчанию при переборе используется основная последовательность размещений (M = 1). Для перехода к оптимизированной последовательности нужно увеличить значение k на 20. Здесь значение M задается полиномом $M = 2\ n^2 + n - 1$. Пример оператора с перебором троек классов:

xgroup (tt_alu, 23, 4000)

Пример дерева инструкций (переменная tt_alu типа tree) приведен выше.

Цикл времени выполнения в тестовом коде позволяет выполнить одни и те же инструкции в разных условиях: на первой итерации цикла инструкции загружаются в кэш, на следующих читаются из кэша. Для организации цикла нужно использовать оператор loop, иначе генератор может не обеспечить корректность выполнения второй и следующих итераций цикла. В частности, в цикле loop учитывается возможность изменения условий переходов. Нужно заметить, что в случае выдачи очень большого числа инструкций цикл может оказаться длиннее кэша, и тогда его использование мало что даст.

Оператор loop также обеспечивает корректность предзагрузки в цикле. Обычно после использования предзагруженного регистра запись в него разрешена. В цикле это не так: если загрузка константы в регистр происходит вне цикла или на большем уровне вложенности, то есть опасность порчи содержимого перед началом следующей итерации. В операторе loop уровни вложенности для предзагрузки контролируются.

5. Заключение

Использование перебора классов инструкций при генерации случайных тестов позволяет оценивать покрытие по комбинациям сочетаниям классов инструкций. Объем тестового кода получается гораздо меньшим, чем при переборе комбинаций самих инструкций. Это дает возможность компенсировать нарушения последовательности классов за счет многократного повторения перебора. Генератор тестов позволяет избежать предопределенных комбинаций, таких как загрузка регистра непосредственно перед переходом по этому регистру.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения. 0580-2022-0004», Рег № 122041100063-6.

Random Tests with Enumeration of Instruction Classes

A.S. Koutsaev

Abstract. The advantage of random test generators is ease of use and the ability to issue large amounts of code. Stochastic testing reveals both ordinary and specific errors (rare combinations and specification flaws). At the same time, the increase in test coverage does not directly depend on the generator control parameters. Enumeration of placements of n instruction classes by k allows to obtain coverage for the corresponding combinations of instructions with acceptable code size.

Keywords: random tests, coverage, verification, enumeration of instruction classes

Литература

1. Bergeron J. Writing Testbenches using SystemVerilog. Springer, 2006.

2. Хисамбеев И.Ш. Роль стохастического тестирования в функциональной верификации микропроцессоров. Программные продукты и системы 2012, №3, 107-112.

3. И.В. Грибков, А.В. Захаров и др. Стохастическое тестирование в системе INTEG. Программные продукты и системы. 2007, № 2, 22–26.

4. И.В. Грибков, А.В. Захаров и др. Развитие системы стохастического тестирования INTEG. Программные продукты и системы 2010, №2, 14-22.

5. А.С. Куцаев. Развитие генератора случайных тестов tergen. Труды НИИСИ РАН 2018, Том 8, № 1, 19-26.

Заместительная гормональная терапия и здоровье женщин в постменопаузе

С.Ю. Лукашенко[1], Т.И. Рубченко[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, s_lukashenko@mail.ru;

[2]tatiana-maxim@mail.ru

Аннотация. Проведен анализ данных о 745 пациентках с естественной и хирургической постменопаузой. Обсуждается характер влияния заместительной гормональной терапии (ЗГТ) на здоровье женщин. Для группы из 91 пациентки, принимавших ЗГТ на протяжении 7 лет, исследована динамика минеральной плотности костной ткани (МПКТ) шейки бедра и поясничных позвонков методом двуэнергетической рентгеновской абсорбциометрии (ДРА). Изучались характер изменений МПКТ на фоне приема ЗГТ в зависимости от ее исходного состояния, а также динамика оценочного заключения о наличии остеопороза и остеопении у пациентки. Исследование позволяет сделать вывод, что ЗГТ, как минимум, на 7 лет «естественным способом» тормозит старение костной ткани в проблемных зонах и должна быть основной при лечении постменопаузального остеопороза.

Ключевые слова: анализ данных, медицинская информатика, статистическая проверка гипотез, менопауза, постменопауза, заместительная гормональная терапия, минеральная плотность костной ткани, остеопороз, остеопения, денситометрия

1. Введение

Статья является результатом длительной совместной работы математика и врача и основана на методах медицинской информатики и инженерии знаний, разработанных академиком И.М.Гельфандом и группой его сотрудников-математиков, работавших с талантливыми и любознательными врачами, специализирующимися в самых разных областях медицины [1].

Вопросы, касающиеся здоровья и качества жизни женщин постменопаузального возраста являются крайне важными. Средний возраст наступления естественной менопаузы (ЕМ) у российских женщин по данным разных исследователей – от 48 лет до 51 года [2]. Таким образом, около трети своей жизни среднестатистическая россиянка проводит в состоянии постменопаузы, то есть в периоде жизни, когда репродуктивная система женщины перестает функционировать естественным образом. При этом от женщин, вступивших в этот период, все еще ожидается высокий уровень социальной активности и предполагается, что женщина еще длительное время сохраняет свою трудоспособность, и, в свете новых социальных реалий, ожидается, что это время сохранения трудоспособности будет удлиняться. Скажем несколько слов о терминологии. Менопауза – это дата, от которой начинается отсчет времени, время жизни женщины после наступления менопаузы называется постменопаузальным или просто менопаузальным периодом, или, еще упрощая, может быть использованы термины типа: длительность менопаузы 5 лет, то есть остановка менструального цикла у женщины произошла 5 лет назад.

У женщин с хирургической менопаузой (ХМ), т.е. после гистерэктомии (удаления матки) в репродуктивном возрасте, датой менопаузы считается дата операции, и, таким образом, период постменопаузы, в среднем, начинается раньше и, соответственно длится дольше, чем у женщин с естественной менопаузой.

У женщин с естественной менопаузой ее наличие подтверждается ретроспективно, только после года наблюдения, то есть отсутствием менструации в течение года. Нельзя сказать, что длительность менопаузального периода пациентки 10 или 11 месяцев, пока длительность отсутствия менструации менее года – женщине диагностируется период перименопаузы, то есть менопауза (дата остановки менструального цикла) диагностируется «задним числом». Перименопауза - промежуток времени, предшествующий менопаузе, когда уже наблюдаются клинические проявления перестройки гормональной системы организма женщины, связанной с угасанием продукции гормонов прогестерона и эстрадиола яичниками. Началом клинических проявлений перименопаузы является нарушение регулярности менструального цикла и изменения его характеристик. При исходной дисфункции яичников ориентируются на клинико-лабораторные параметры.

Диагностируется два основных варианта течения перименопаузы: по типу атрезии фолликул и по типу их персистенции. Эти два варианта связаны с тем, каким является ответ организма

на реакцию «стража и регулятора» эндокринной системы – гипофиза, который, по мере угасания функции яичников, начинает продуцировать гормоны, стимулирующие выработку половых стероидов, например фолликуло-стимулирующий гормон (ФСГ), активизирующий выработку эстрадиола.

Если реакция организма на повышение ФСГ слабая и уровень эстрадиола не высок, то мы наблюдаем перименопаузу по типу атрезии фолликул с постепенным затуханием менструального цикла и, у большинства женщин, с появлением и нарастанием так называемого климактерического синдрома, обусловленного эстрогендефицитным состоянием: горячие приливы, повышенная потливость, расстройство сна, ухудшение памяти, снижение работоспособности, головокружения, боли в позвоночнике, суставах, мышцах; повышенная тревожность и нервозность. Не обязательно присутствует весь набор этих неблагоприятных признаков, их сочетания могут быть различными.

Если же реакция эндокринной системы на повышение гипофизом выработки стимулирующих гормонов «успешна», и наблюдается рост уровня эстрадиола на фоне повышения ФСГ вплоть до очень высоких уровней, тогда мы наблюдаем, как промежутки эстрогендефицитного состояния с климактерическим синдромом перемежаются с вроде бы «светлыми» промежутками, когда вдруг исчезают все эти неприятные климактерические состояния. Но, эти, казалось бы, «светлые» промежутки времени могут оказаться опасными не только для здоровья, но и для жизни женщины, так как высокий уровень эстрадиола на фоне низкого уровня прогестерона может привести к гиперплазии эндометрия и маточному кровотечению, требующему экстренного инвазивного вмешательства – выскабливанию эндометрия матки. Пациентки с перименопаузой по типу персистенции фолликул нуждаются в тщательном наблюдении и своевременном лечении препаратами прогестерона (гестагенами), которые формируют псевдоменструацию и не допускают развития гиперплазии эндометрия. В результате такого лечения перименопауза по типу персистенции фолликул постепенно трансформируется в перименопаузу по типу атрезии фолликул, иногда на это уходит несколько лет. Препараты ЗГТ в перименопаузе по типу персистенции фолликул не назначаются.

Следует сказать, что сейчас заместительную гормональную терапию (ЗГТ) все чаще стали называть постменопаузальной гормональной терапией (ПГТ), что уже подразумевает, что это терапия, которую не назначают в перимено-паузе, а только через год после остановки менструальной функции, когда уже есть полная уверенность, что ее назначение не приведет к гиперэстрогении. Однако, при очень тяжелом течении климактерического синдрома и перименопаузе по типу атрезии фолликул могут назначаться некоторые препараты ЗГТ при условии регулярного ультразвукового и гормонального обследования.

Перименопауза по типу атрезии фолликул, как правило, при естественном развитии событий, начинается несколько раньше и быстрее переходит в постменопаузу, чем пери-менопауза по типу персистенции фолликул, которая обычно длится несколько лет.

К моменту диагностирования естественной постменопаузы тяжесть климактерического синдрома достигает своего максимума. При отсутствии лечения, со временем острота ситуации снижается и соответствующие проявления эстрогендефицитного состояния постепенно ослабевают. Временной период жизни женщины, включающий перименопаузу и ту часть постменопаузы, которая протетакает на фоне манифестации эстрогендефицитного состояния, называется климаксом, а сами клинические проявления, связанные с этим состоянием – климактерическим синдромом или синдромом менопаузы. Продолжительность этого периода может существенно различаться как по интенсивности проявления, так и по длительности – от нескольких месяцев до 10-12 лет, у большинства женщин он длится от 2 до 5 лет. Примерно 40% женщин с ЕМ тяжело переживают этот период и нуждаются в назначении ЗГТ. Напомним, что от женщины этого возраста (45-55 лет) ожидается высокий уровень социальной активности. На деле мы нередко наблюдаем, что женщины, не получая адекватной медикаментозной коррекции своего состояния, «уходят в болезнь», а соответствующие психоневрологические отклонения могут приводить к осложнениям в отношениях даже с родными и близкими, а не только с социумом в целом.

Нормой наступления естественной менопаузы считается возраст 45-55 лет, в противном случае говорят о ранней (<45 лет) или, соответственно, о поздней (>55 лет) менопаузе. Обычная длительность назначения ЗГТ женщинам с ЕМ – 5-7 лет, препараты не назначаются женщинам (как с ЕМ, так и с ХМ) достигшим 60 лет. Длительное сохранение гормонального фона, соответствующего репродуктивному периоду нежелательно для возрастных женщин, так как считается, что повышается риск развития рака молочных желез. Однако сейчас появляются препараты с пониженным содержанием эстрадиола и прогестерона - 0,5 обычной дозировки, которые в настоящее время назначаются после 2-

3 лет приема обычных препаратов при ЕМ. В последние несколько лет появились препараты mini - с 0,25 обычной дозировки для возрастных женщин.

2. Результаты исследований

2.1. О здоровье женщины в постменопаузе

Эта статья написана по результатам совместной работы врача Рубченко Татьяны Ивановны (гинеколог-эндокринолог, д.м.н.) и математика Лукашенко Светланы Юрьевны (к.ф.м.н.), которая длилась около 15 лет. Работа посвящена изучению состояния здоровья и оптимизации ведения пациенток с хирургической и естественной постменопаузой.

Нами собраны сведения о 745 пациентках: I группа - 216 пациенток с ХМ (после гистэрэктомии с придатками); II группа - 153 пациентки после гистэрэктомии, с наличием яичниковой ткани (оставлены яичники или их часть); III группа - 376 пациенток с ЕМ.

Пациенткам проводилось исследование гормонов - половых стероидов и тиреотропных гормонов, биохимических параметров крови (в том числе липидного профиля), исследовалась центральная гемодинамика - кровоток в магистральных артериях, проводилось ультразвуковое исследование органов малого таза и гинекологический осмотр, собраны клинико-анамнестические данные. Определялась минеральная плотность костной ткани (МПКТ), плотность внутренней (трабекулярной) костной ткани 2-4 поясничных позвонков на компьютерном томографе, или(и) иследования суммарной трабекулярной и кортикальной костной ткани 1-4 поясничных позвонков и костной ткани в пояснично-крестцовой области скелета (5 параметров, в том числе и в шейке бедра) методом двуэнергетической рентгеновской абсорбциометрии (ДРА).

Данные собраны с помошью разработанной иформационной карты и введены в компьютерную базу данных. Анализ данных проводился с использованием оригинального пакета программ, разработанным сотрудником группы И.М.Гельфанда в ИПМ РАН доктором физ.-мат. наук Ю.Б.Котовым, применялись непараметрические методы прикладной статис-тики, критерии Манна-Уитни и Хи-квадрат, различия считали достоверными при P<0,05.

Части обследуемых пациенток были назначены препараты ЗГТ, части - препараты кальция для лечения остеопороза, часть пациенток не лечилась, но обследовалась, им были даны рекомендации по питанию и оздоровительным процедурам.

У пациенток I группы менопауза – дата операции. У большинства женщин этой группы исходно была сочетанная гинекологическая патология, основные показания к операции: доброкачественные заболевания матки или придатков - миома матки, кисты или кистомы яичников (41% наблюдений, медиана возраста операции 47 лет); эндометриоз (часто тяжелый, с прорастанием в соседние органы) (36%, 44 года): гнойно-воспалительные процессы малого таза (23%, 36 лет). На рис. 1 показана динамика изменения эстрадиола и ФСГ в зависимости от давности операции.

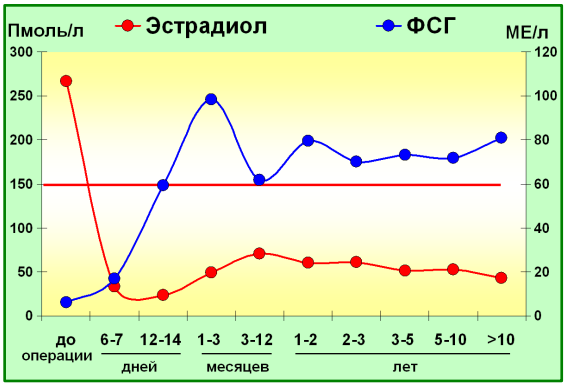

Рис. 1. Уровни эстрадиола и ФСГ у пациенток I группы (без ЗГТ)

Можно сказать, что женщины I группы оказываются в более сложном положении, чем женщины с естественной менопаузой, так как у них полностью удалены яичники. У женщин в состоянии естественной постменопаузы ткань яичников еще длительное время (около 10 лет) продуцирует мужские гормоны, которые на фоне эстрогендефицитного состояния могут конвертироваться в женские. И лишь, примерно, через 10 лет (продолжительность у разных женщин может различаться), когда возможности яичниковой ткани истощаются, женщина вступает в период адренопаузы, полного истощения функции яичников. Женщины I группы после операции сразу же оказываются в состоянии адренопаузы.

Большинство пациенток I группы (более 90%) нуждаются в назначении ЗГТ, однако не все. У некоторых пациенток происходит компенсация гипоэстрогенного состояния за счет внутренних резервов, можно предполо-жить, что это происходит за счет активизации выработки андрогенов надпочечниками и активной конверсии андрогенов в эстрогены, в процесс конверсии включена жировая ткань, как депо эстрогенов и андрогенов.

Процесс конверсии имеет место всегда, и при хирургической и при естественной постменопаузе. Крайне редко наблюдаются значения близкие к нулевым, но обычными являются значения

эстрадиола в 2-5 раз ниже нижней границы нормы репродуктивного возраста (150 пмоль/л). Однако и среди пациенток, перенесших полную кастрацию есть пациентки не нуждающиеся в ЗГТ, их менее 10%, но они есть и сама по себе пангистерэктомия в анамнезе не является абсолютным показанием к назначению ЗГТ!

У пациенток II-ой группы при достаточном объеме оставленной яичниковой ткани (более 3,5 см^3) довольно быстро восстанавливаются нормальные уровни половых стероидов (рис. 2).

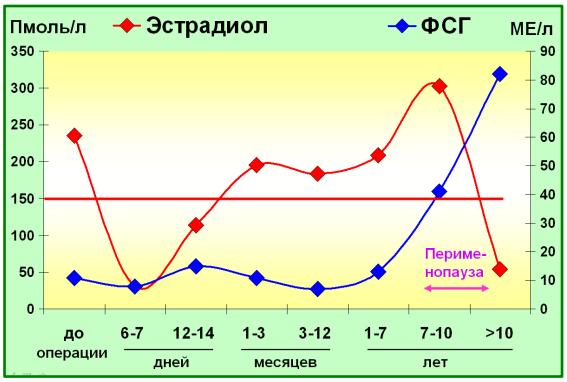

Рис. 2. Динамика содержания эстрадиола и ФСГ в периферической крови у женщин в ближайшее и отдаленное время после гистерэктомии с сохране-нием яичниковой ткани (медианы параметров)

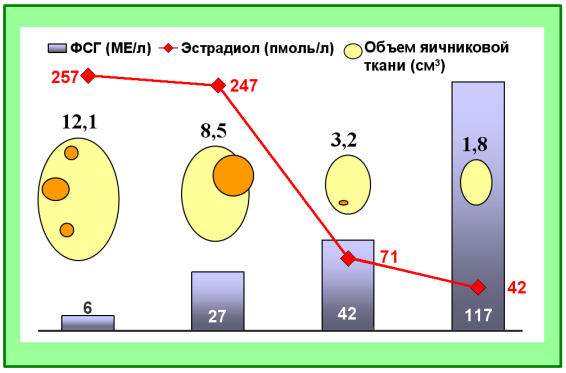

Рис. 3 Соотношение между объемом и структурой яичников и уровнями эстрадиола и ФСГ у женщин II-ой группы (медианы параметров)

В дальнейшем эти пациентки по своему состоянию ничем не отличаются от пациенток не переносивших гистерэктомию, кроме отсутствия менструации. Самым простым способом определения наличия и времени овуляции у этих пациенток является метод измерения ректальной температуры [3]. Во время овуляции температура тела повышается. Зная время овуляции и, таким образом «день цикла», можно провести более точное обследование пациенток, определяя уровни половых стероидов и проводя УЗИ яичников.

Наступление климакса у женщин второй группы, как правило, бывает в соответствии с возрастом и о нем можно судить по возникающим клиническим проявлениям, а также по состоянию яичников по данным УЗИ (рис. 3). Как правило, в дальнейшем пациентки этой группы приходят к врачу для обследования при появлении приливов и или других климактерических расстройств и наблюдаются также, как и пациентки с естественной постменопаузой [4].

Считается, что при естественной постменопаузе ЗГТ показана примерно 40% женщин, у которых климактерический синдром протекает в тяжелой форме и у которых нет противопоказаний к назначению этих препаратов. Основная задача при назначении ЗГТ – устранение климактерического синдрома, который резко ухудшает качество жизни женщины: горячие приливы, повышенная потливость, расстройство сна, ухудшение памяти, снижение работоспособности, головокружения, боли в позвоночнике, суставах, мышцах, повышенная тревожность и нервозность. Эта задача успешно решается назначением любого из современных препаратов ЗГТ. Однако спектр воздействия ЗГТ на здоровье женщины очень широк. Эти препараты влияют на липидный профиль крови, на параметры центральной гемодинамики, на биохимические показатели крови, на изменение плотности костной ткани, позволяют предотвратить такое грозное осложнение эстрогендефицитного состояния, как быстрое развитие постменопаузального остеопороза (ПО) – разжижения костной ткани до степени высокого риска переломов.

Плотность костной ткани после рождения человека и в процессе его роста увеличивается вплоть до возраста 28-30 лет, затем начинается процесс старения позвоночника и других костных структур и происходит постепенное снижение плотности на протяжении всей жизни, это снижение у женщин ускоряется в период пери- и постменопаузы на фоне дефицита эстрогенов.

Высокая распространенность постменопаузального остеопороза, частота и тяжесть его осложнений, стоимость лечения остеопоротических переломов в нашей стране явно недооценены. Начало развития ПО совпадает с менопаузой, и после 50 лет по данным литературы его выявляют у 30-40% женщин. Последствия его – атравматичные или малотравматичные переломы позвонков появляются, как правило, после 60 лет, а самые тяжелые осложнения, переломы шейки бедра, еще в более позднем возрасте. Следовательно, между началом развития ПО и его осложнениями проходит несколько лет, дающих возможность диагностики его и лечения, что по тем же литературным данным снижает на 50-60% частоту переломов. Переломы лечат травматологи, а вот уникальную возможность диа-

гностировать вовремя ПО и применить общепризнанный метод лечения его – ЗГТ, имеют только гинекологи.

Когда врачи стали изучать изменение плотности костной ткани на фоне ЗГТ, то была надежда, что скорость снижения плотности костной ткани замедлится. Оказалось, что эффект лечения превзошел ожидания и у части больных на фоне ЗГТ был отмечен рост плотности костной ткани. В результате этих исследований ЗГТ стали рассматривать и как метод лечения остеопороза, а не только климактерического синдрома.

Однако, после опубликования в 2002 году первых отчетов исследования WHI (Women's Health Initiative - Инициатива во имя здоровья женщин, США), когда было заявлено о связи ЗГТ с повышением риска развития рака молочной железы и отсутствием положительного влияния ЗГТ на сердечно-сосудистую систему. На основании этих публикаций резко снизилось число назначающих и принимающих ЗГТ. При более тщательном анализе этого исследования выяснилось, что назначения ЗГТ в корне противоречило реальной клинической практике [5]. ЗГТ назначали пациенткам 60-70 лет, с большой длительностью постменопаузы (от 5 до 20 лет) и осложнения от приема ЗГТ наблюдались именно у женщин этой возрастной группы. В 2012 году в Дании [6] были опубликованы результаты аналогичного 10-летнего рандомизированного наблюдения за пациентками, которым в первые годы постменопаузы назначалась ЗГТ на 5 лет. В этом исследовании показано как отсутствие отрицательного воздействия ЗГТ на повышение частоты рака груди, так и положительное влияние ЗГТ, такое, например, как снижение смертности от сердечно-сосудистых заболеваний в два раза, в сравнении с контрольной группой. В результате, в этом же 2012 году, после пересмотра данных ранее проведенных исследований, международные эксперты (в том числе и WHI) заявили, что вероятность развития осложнений на фоне приема препаратов ЗГТ крайне низкая и соответствующим регулирующим структурам следует обновить рекомендации по применению ЗГТ.

Однако эти десять лет оказали крайне негативное влияние на использование ЗГТ в клинической практике во всем мире. В России это время совпало с уходом части производителей ЗГТ с российского рынка, что еще усугубило ситуацию. Сейчас – время реабилитации ЗГТ.

2.2. Изменения плотности костной ткани у пациенток длительно принимавших ЗГТ

Костная система человека примерно на 20% состоит из трабекулярной (губчатой) костной ткани, а на 80% из кортикальной (компактной). Постменопаузальный остеопороз (ПО), в большей степени связывают со снижением минеральной плотности трабекулярной костной ткани на фоне возникшего дефицита эстрогенов в постменопаузе. Доля трабекулярной ткани в поясничных позвонках L1-L4 оценивается разными авторами от 50 до 66%, в шейке бедра от 25 до 40%, то есть существенно ниже [7].

МПКТ в нашем исследовании определялась методом двуэнергетической рентгеновской абсорбциометрии (ДРА). При этом методе исследуется суммарная плотность костной ткани – трабекулярной и кортикальной.

Исследовалась динамика значений T- и Z-критерия среднего значения МПКТ L1-L4, а также шейки бедра (рис. 4)

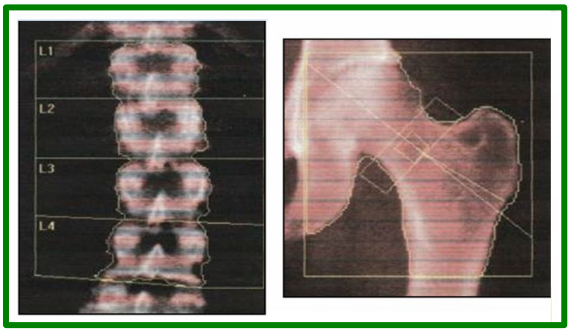

Рис. 4. Локализация исследования МПКТ методом ДРА, поясничные позвонки L1-L4 и шейка бедра.

Исследование проводилось рентгеновским денситометром фирмы «Hologic». T-критерий – оценивает степень отклонения от пиковой плотности костной ткани (т.е. от МПКТ здоровой 30-летней женщины) в количестве стандартных отклонений. Редкий случай в медицине – МПКТ имеет нормальное распределение! При значении T-критерия менее (–2,5) диагностируется остеопороз и риск перелома. При значении T-критерия менее (-1), но более (-2,5) МПКТ считается сниженной, диагностируется остеопения, но риск перелома не велик. Если T-критерий>-1, считаем, что МПКТ не снижена.

Z-критерий оценивает степень отклонения от возрастной нормы, остеопения по Z-критерию диагностируется в диапазоне (-1, -2), остеопороз при значениях (–2) и менее.

Для лечебных целей важнее определение T-критерия, так как при очень пожилом возрасте нормальное значение Z-критерия (то есть среднее популяционное значение) может оказаться в зоне высокого риска перелома.

В течение 7 лет ЗГТ получала 91 женщина, из них 48 с ЕМ и 43 - с ХМ: первые - комбинированные препараты, циклические или монофазные, вторые, в основном, принимали «чистые»

эстрогены, но нескольким пациенткам, оперированным по поводу распространенного эндометриоза, тоже были назначены комбинированные препараты.

Средний возраст женщин был 47 лет (от 27 до 59, квартили 43-51 лет), длительность менопаузы – 24 месяца (от 0 до 134 месяцев, квартили 12-65 месяцев).

Приведем два клинических наблюдения с довольно характерными изменением МПКТ у пациенток с естественной (рис. 5) и хирургической менопаузой (рис. 6). На рисунках приведена динамика оценки Т- и Z-критерия состояния плотности костной ткани.

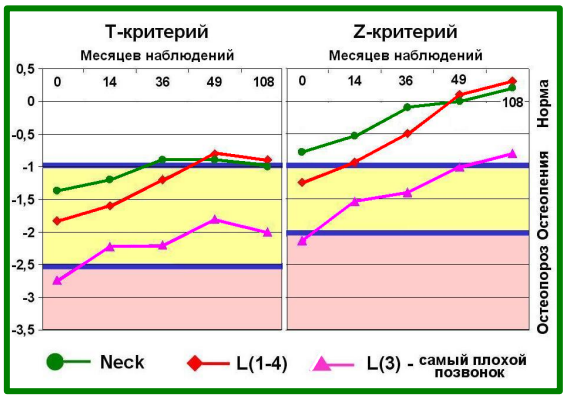

Рис. 5. Пациентка Л, ЕМ, 9 лет наблюдения, Климонорм, последние 5 лет – Фемостон 1/5

Клиническое наблюдение 1. Женщина с ЕМ, наступившей в возрасте 46 лет. Перимено-пауза протекала по типу персистенции фолликул, назначались гестагены. В 47 лет был назначен препарат Климонорм, восстанавлива-ющий менструальную функцию и профилакти-рующий развитие гиперплазии эндометрия. После 3-х лет наблюдения назначен монофазный препарат Фемостон 1/5 с уменьшенным содержанием эстрадиола и гестагена. Масса пациентки почти не менялась, оставаясь в диапазоне 82-84 кг. На рис. 5 приведены изменения МПКТ шейки бедра, среднего значения МПКТ по 4-м позвонкам L1-L4 и значения МПКТ в худшем (3-ем) поясничном позвонке L3. У пациентки исходно диагностированы (по Т-критерию) остеопения L1-L4 и шейки бедра, однако при оценке каждого из позвонков выявлен остеопороз 3-го поясничного. По Т-критерию мы видим выход оценки МПКТ L1-L4 после 3 лет и шейки бедра после 4 лет лечения на уровень нижней границы диапазона нормы. В 3-ем поясничном позвонке после года приема ЗГТ наблюдаем выход из опасной зоны остеопороза в диапазон остеопении, где все последующее время оценка и оставалась с небольшой положительной динамикой во времени. Оценивая степень соответствия своим возрастным нормативам (Z-критерий), мы видим неуклонный рост оценки МПКТ. Для шейки бедра этот рост наблюдается в диапазоне нормы, для L1-L4 - от остеопении к норме, в L3 - от остеопороза к нижней границе своей возрастной нормы.

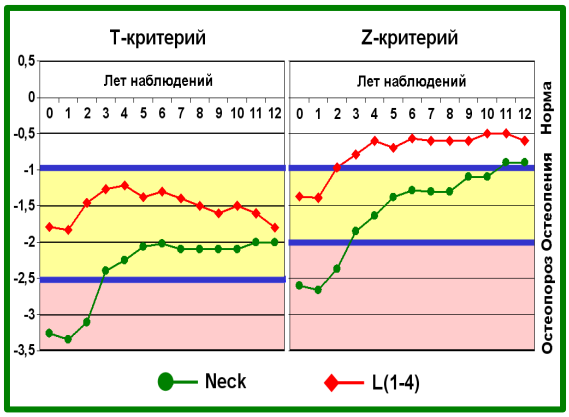

Рис. 6. Пациентка К, ХМ, 13 денситометрий - 12 лет наблюдения, Эстрофем

Клиническое наблюдение 2. Пациентка была прооперирована в возрасте 42 лет, ей была сделана пангистерэктомия (удалены матка и придатки). Женщина имела «субтильное» телосложение, масса тела была в начале наблюдения 44 кг, на протяжении всех 12 лет наблюдения масса тела почти не менялась, колебания были в диапазоне 44-46 кг. Назначенный препарат – эстрофем, содержит эстрадиол и не содержит гестагены. У пациентки исходно диагностированы (и по Т- и по Z-критерию) остеопения в поясничных позвонках и остеопороз шейки бедра. По Т-критерию остеопения в позвонках оставалась на протяжении всех 13 лет наблюдения, а МПКТ шейки бедра уже через 3 года лечения вышла из зоны высокого риска перелома (остеопороза), после пяти лет лечения достигла наилучшего результата и на этом уровне стабилизировалась. Оценивая степень соответствия своим возрастным нормативам (Z-критерий), мы видим, что после трех лет лечения МПКТ позвонков достигла возрастной нормы и оставалась в нормативном диапазоне на протяжении всего времени наблюдения. Возрастная оценка МПКТ шейки бедра улучшалось на протяжении всех 13 лет наблюдения и в результате лечения из зоны остеопороза вышла к нижней границе своей возрастной нормы. Отметим небольшое снижение МПКТ L1-L4 после 4 лет наблюдения. Оно связано с уходом производителя препарата с российского рынка. Женщина отказалась менять препарат, так как прекрасно себя чувствовала, принимая его, и несколько месяцев была без лечения, прежде чем смогла найти возможность принимать его в

дальнейшем.

Мы сравнили влияние длительного приема ЗГТ на состояние минеральной плотности костной ткани (МПКТ) у пациенток с ее различным исходным уровнем. Важным является изучение влияния МГТ в тех локусах, переломы в которых приводят к инвалидизации пациенток. Исследовалась динамика значений Т-критерия среднего значения МПКТ L1-L4 и шейки бедра (рис. 7).

Остеопороз в L1-L4 выявлен у 34(37,3%) женщин, остеопения у 36(39,6%), нормальная МПКТ была у 21(23,1%) пациенток. Медианы значений Т-критерия до лечения были, соответственно, (-2,65), (-1,69) и (-0,04).

Остеопороз шейки бедра был выявлен у 22 (24,1%) женщин, остеопения у 28 (30,8%), нормальная МПКТ была у 41 (45,1%) пациенток. Медианы значений Т-критерия до лечения были - (-2,75), (-1,85) и (-0,56).

Изменения в L1-L4 и группах с исходной остеопенией и остеопорозом имеют одинаковый характер. В первые 3-4 года лечения мы наблюдаем достоверный рост МПКТ, причем в группе остеопороза уже через год медиана выходит в зону остеопении, то есть выходит из зоны высокого риска переломов.

При более длительном лечении наблюдается тенденция относительного снижения МПКТ. У пациенток с исходной нормальной МПКТ в первые 1-2 года лечения происходит снижение МПКТ, то есть организм «не замечает» лечения, после 2 лет лечения наблюдается тенденция замедления потери костной ткани и стабилизации МПКТ.

В шейке бедра в группе с нормальной МПКТ организм «не замечает» лечения, в группе с исходной остеопенией наблюдается стабилизация значения Т-критерия на протяжении всех 7 лет наблюдения, что можно расценивать как неплохой эффект лечения. В группе с исходным остеопорозом наблюдается постоянный рост МПКТ на фоне лечения, медиана Т-критерия выходит в зону остеопении после 3 лет лечения, наилучшие результаты мы наблюдаем после 7 лет лечения.

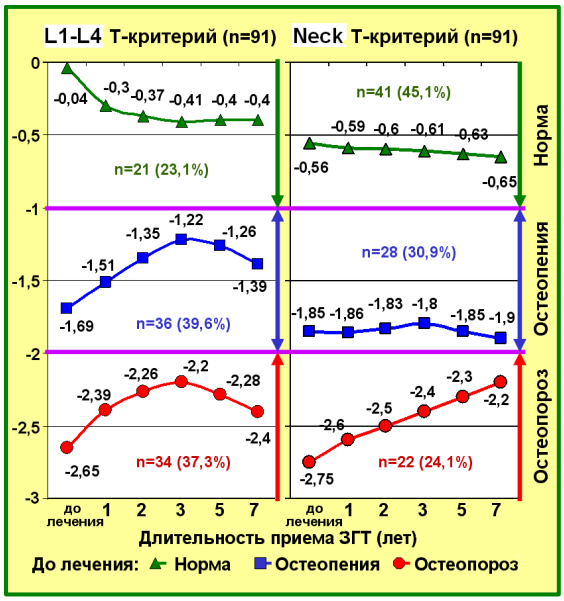

Рис. 7. Изменение Т-критерия МПКТ на фоне ЗГТ в зависимости от ее исходного состояния (до лечения) в позвоночнике и в шейке бедра.

Исследование показало, что чем больше выражено снижение МПКТ, тем эффективнее лечение ПО препаратами ЗГТ. Воздействие ЗГТ на патологические изменения в шейке бедра, которая состоит в основном из кортикальной ткани, носят более «инерционный характер», чем в L1-L4, для достижения хорошего эффекта требуется более длительное время.

Обратим внимание на различие в распределении частоты выявления остеопороза в L1-L4 – 37,3% и в шейке бедра – 24,1%. Соответственно, частота нормальной МПКТ в L1-L4 – 23,1% и в шейке бедра – 45,1%. Эти распределения хорошо согласуются с представлениями о том, что постменопаузальный остеопороз в большей степени является остеопорозом трабекулярной, а не кортикальной костной ткани.

Можно сказать об общей тенденции, которую мы наблюдаем при изучении воздействия ЗГТ на различные параметры, связанные с наличием гипоэстрогении в организме: чем более были выражены неблагоприятные последствия этого состояния, тем более выраженный эффект лечения мы наблюдаем, тем ярче выражена реакция благодарности организма за оказанную ему помощь. Эти тенденции мы наблюдали при изучении кровотока во внутренней сонной артерии [8], при изучении изменений коэффициента атерогенности, характеризуещего липидный профиль крови [9] и при изменении МПКТ. У пациенток с исходно нормальным значением параметра организм «не замечает» лечения. При длительном наблюдении мы выявили, что «плохие» параметры улучшаются, но, как правило, не до

«идеальной нормы» а до относительно благоприятных значений – значений «выхода» из зоны опасности для выживания, после этого включаются нормальные механизмы старения и параметры начинают ухудшаться с увеличением возраста, но в несколько более медленном темпе, чем в популяции (коэффициент атерогенности растет, плотность костной ткани снижается в абсолютных значениях, но скорость этого снижения ниже популяционной.

При назначении лечения остеопороза врачом-клиницистом важным является само наличие остеопороза, как системного заболевания, а не его локализация. Итоговое заключение о наличии остеопороза у пациентки делается при его наличии хотя бы в одной из изучаемых локализаций, то есть либо в шейке бедра либо в одном из 4-х поясничных позвонков. К норме относят пациенток у которых МПКТ для всех изучаемых локализаций находится в нормативном диапазоне. Таким образом, считают, что у пациентки остеопения, если у нее ни в одной из локализаций не выявлен остеопороз, но хотя бы в одной локализации имеется остеопения.

Мы исследовали изменения общих оценочных заключений о состоянии минеральной плотности костной ткани (МПКТ) на фоне длительного приема ЗГТ по Т-критерию (рис. 8) и Z-критерию (рис. 9).

По Т-критерию мы наблюдаем устойчивый рост частоты остеопении с 50,5 % до 78,6% (достоверно), в основном за счет снижения частоты остеопороза, который был выявлен у 39,6% женщин, через 3-5 лет лечения его частота - 16,7%-17,6%, после 7 лет - 21,4%, а также за счет плавного уменьшения частоты нормальной МПКТ с 9,9% до 0.

Анализ Z-критерия показал, что частота соответствия своей возрастной норме после 3-5 лет лечения увеличилось с 25,3% до 50,1-47,1% (достоверно) в основном за счет снижения частоты остеопороза по Z-критерию.

И по Т-критерию, и по Z-критерию распределения состояния МПКТ после 7 лет лечения хуже, чем после 3-5 лет лечения. Тем не менее, эти распределения более благоприятны, чем те, которые мы наблюдали перед назначением ЗГТ. Так, частота остеопороза по Т-критерию снизилась с 39,6% до 21,4%, а частота соответствия популяционной норме (норма по Z-критерию) увеличилась с 25,3% до 35,7%.

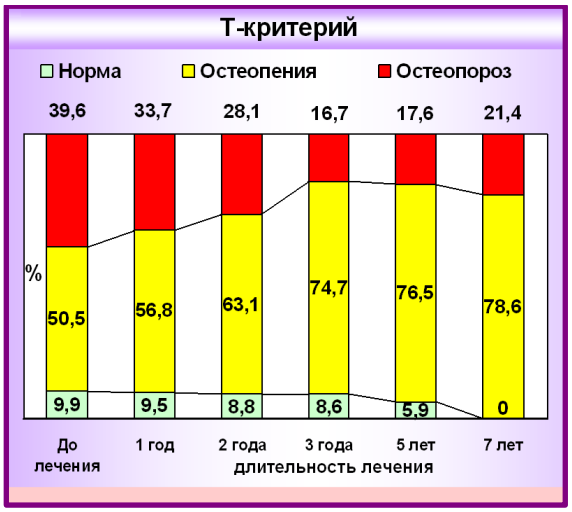

Рис. 8. Динамика частоты (%) остеопороза и остеопении в исследуемой области (ДРА, Т-критерий) на фоне лечения ЗГТ (итоговое заключение)

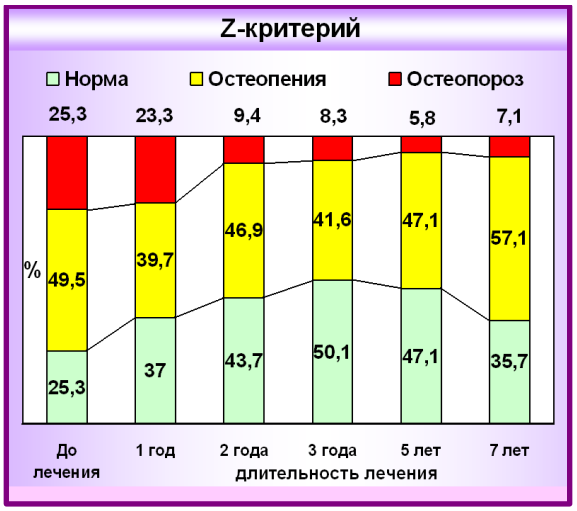

Рис. 9. Динамика частоты (%) остеопороза и остеопении в исследуемой области (ДРА, Z-крите-рий) на фоне лечения ЗГТ (итоговое заключение)

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0007 «Развитие методов математического моделирования распределённых систем и соответствующих методов вычисления. 0580-2022-0007»

3. Заключение

Решение о назначении ЗГТ должно приниматься строго индивидуально на основе комплексного клинико-лабораторного обследования пациентки.

Наилучший результат воздействия ЗГТ на МПКТ мы наблюдаем после 3-5 лет лечения. По Т-критерию частота ПО, основного фактора риска переломов, снижается с 39,6% до 16,7-

17,6% (достоверно). Частота остеопении увеличилась за счет повышения МПКТ у пациенток с исходным ПО и уменьшения МПКТ у женщин с исходно нормальной МПКТ, но снижение МПКТ у женщин с исходно нормальной МПКТ не является результатом неэффективности ЗГТ. Нормальная МПКТ на фоне терапии климактерического синдрома половыми стероидами не повышается. Из этого следует, что профилактику остеопороза при нормальной МПКТ проводить не надо.

Hormone Replacement Therapy and Health of Postmenopausal Women

S.Yu.Lukashenko, T.I.Roubtchenko

Abstract. Data on 745 patients with natural and surgical postmenopause has been analyzed. An effect of hormone replacement therapy (HRT) on women's health is discussed. For a group of 91 patients who took HRT for 7 years, the dynamics of bone mineral density (BMD) of the femoral neck and lumbar vertebrae was studied by the method of dual-energy X-ray absorptiometry (DRA). The nature of changes in BMD against the background of HRT intake was studied depending on its initial state as well as the dynamics of the evaluation conclusion about the presence of osteoporosis and osteopenia in the patient. The study allows us to conclude that HRT "naturally" slows down the aging of bone tissue in problematic areas for at least 7 years and should be a basis in the treatment of postmenopausal osteoporosis.

Keywords: data analysis, medical informatics, statistical testing of hypothesis, menopause, postmenopause, hormone replacement therapy, bone mineral density, osteoporosis, osteopenia, densitometry

Литература

1. Гельфанд И.М., Розенфельд Б.И., Шифрин М.А. Очерки о совместной работе математиков и врачей. Москва, Наука, 1989, 279 с.

2. Сметник В.П., Тумилович Л.Г. Неоперативная гинекология. Москва, МИА, 1998, 214 с.

3. Кватер Е.И., Гормональная диагностика и терапия в акушерстве и гинекологии, Москва, Медицина, 1967, 359 с.

4. Рубченко Т.И., Лукашенко С.Ю. Функция яичников после гистерэктомии. «Проблемы репродукции». 2002, Том 8, № 1, 6-11

5. James H. Clark, Critique of Women’s Health Initiative Studies (2002-2006) Nucl Recept Signal. 2006; 4: e023. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1630688/.

6. Schierbeck LL, Rejnmark L, Tofteng CL, et al. Effect of hormone replacement therapy on cardiovascular events in recently postmenopausal women: Randomised trial. BMJ 2012. http://www.bmj.com

7. Волков А.А., Белосельский Н.Н., Прибытков Ю.Н. Абсорбциометрическая оценка некоторых количественных и качественных характеристик костной ткани при снижении минеральной плотности кости. «Остеопороз и остеопатии». 2015, том 18, № 2, 3-5.

8. Рубченко Т.И., Лукашенко С.Ю., Допплерографические показатели кровотока во внутренней сонной артерии у женщин в постменопаузе и влияние на эти показатели заместительной гормональной терапии. «Проблемы репродукции». 2006, Том 12, № 4, 98-103.

9. Рубченко Т.И., Лукашенко С.Ю., Влияние заместительной гормональной терапии на уровни атерогенных фракций липидов в крови постменопаузальных женщин. Материалы XVIII Всероссийского научного форума «Мать и дитя», Москва, 27-29 сентября 2017, 140.

Подготовка к изданию журнала «Труды НИИСИ РАН»

А.А. Асонов[1], А.Н. Годунов[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, asonow@niisi.ras.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nkag@niisi.ras.ru

Аннотация. В статье описывается процесс подготовки номеров журнала к печати от сбора статей до создания оригинал-макета для передачи в типографию; указывается последовательность действий при подготовке журнала, описываются используемые средства и даются рекомендации по их применению, приводятся требования к оформлению страниц, колонтитулов, титульных листов и содержания, даются обзор и рекомендации по использованию издательской системы для журнала Труды НИИСИ РАН. Статья предназначена для лиц, осуществляющих подготовку журнала к изданию, а также будет полезна авторам статей, ознакомив их с процессом формирования и выпуска журнала.

Ключевые слова: подготовка журнала, НИИСИ РАН, оригинал-макет, научный редактор, издательская система

1. Введение

Журнал «Труды НИИСИ» является периодическим изданием, в котором публикуются оригинальные статьи по тематике работ ФГУ ФНЦ НИИСИ РАН.

В статье указаны действия, которые необходимо выполнить при выпуске номера журнала. Ниже используются следующие понятия:

Оригинал-макет – файл, содержащий оформленный номер журнала (кроме обложки), готовый к передаче в типографию.

Аннотация – файл, содержащий пронумерованные названия статей журнала, список авторов и данные о них, а также аннотации и ключевые слова этих статей.

Рабочий каталог – каталог, в который помещаются все файлы, используемые при подготовке журнала.

Издательская система – программа, позволяющая автоматизировать формирование оригинал-макета и аннотацию, а также создание и печать рецензий и экспертных заключений.

Раздел «Порядок выполнения работ» кратко описывает последовательность всех действий по подготовке журнала.

В разделе «Общие правила оформления» приводятся требования к оформлению: параметры и нумерация страниц, настройка колонтитулов, используемые шрифты, форматирование абзацев и др.

Правилам оформления титульных листов посвящен отдельный раздел.

В разделе «Формирование журнала» описываются действия по включению статей в журнал, формированию содержаний (на русском и английском языках), даются советы по использованию текстового редактора Microsoft Word и Adobe Acrobat Pro.

В разделе «Формирование аннотации» описываются правила создания аннотации к журналу, дается последовательность действий по ее созданию.

В разделе «Формирование рецензий и экспертных заключений» описываются прототипы данных документов, указывается, какие изменения следует в них внести.

В конце статьи даются советы по решению некоторых проблем, которые могут возникнуть при подготовке номера к изданию.

Помимо настоящего руководства лицу, осуществляющему подготовку журнала к изданию, передается архив, содержащий структуру рабочего каталога, правила оформления статьи, MS Word шаблоны статьи и оригинал-макета, а также прототипы файлов (рецензии, экспертных заключений) и примеры титульных листов, необходимых для подготовки номера. Также предоставляется издательская система.

В приложении 1 приводятся структура рабочего каталога (рис. 2) и примеры именования содержащихся в нем файлов (таблица 1). В приложении 2 дается описание архива.

2. Порядок выполнения работ

Подготовка журнала к изданию состоит из сбора статей, создания оригинал-макета и аннотации к журналу, а также подготовки рецензий и экспертных заключений о возможности опубликования статей и номера целиком.

2.1. Сбор статей

Как правило, научный редактор получает от авторов статьи по электронной почте в виде файла формата .doc или .docx. Следует проверить соответствие статьи требованиям, описанным в [1].

Статья должна быть создана с применением шаблона. Для ее проверки достаточно убедиться, что во всех ее частях применены корректные стили («_НАЗВАНИЕ СТАТЬИ_», «_СПИСОК АВТОРОВ_», «_АННОТАЦИЯ_», «_ОСНОВНОЙ ТЕКСТ_» и др.). Для этого нужно перейти на закладку «Главная» и в блоке «Стили» нажать на кнопку в нижнем правом углу. В результате в правой части окна редактора откроется окно «Стили», в котором используемые в шаблоне стили отобразятся первыми в списке. При переводе текстового курсора в любую часть статьи в этом окне соответствующий стиль будет выделен прямоугольной рамкой. Чтобы убедиться, что в статье используются только стили, определенные в шаблоне, нужно в окне «Стили» нажать на кнопку «Параметры…» и во всплывающем окне выбрать опцию «Отображаемые стили: Используемые». Для проверки статьи также можно воспользоваться специальной программой.

Статьи, созданные без применения шаблона, а также с применением устаревших версий редактора MS Word (2010 и более старых версий), приниматься не должны.

Также статьи необходимо проверить на отсутствие плагиата. Проверка производится с помощью веб-ресурса «Антиплагиат». Для этого в личном кабинете на сайте https://niisi.antiplagiat.ru нужно нажать кнопку «Добавить документ» и выбрать файл со статьей для проверки. В открывшемся окне выбрать тип документа «Статья», отметить все модули проверки и нажать кнопку «Продолжить». В результате этих действий запустится процесс проверки документа на оригинальность, который может занять некоторое время. После окончания процесса проверки надо нажать кнопку «Посмотреть результаты», а затем кнопку «Полный отчет».

Статья считается прошедшей проверку на оригинальность, если сумма показателей «Цитирования» и «Оригинальность» превышает 75%. В закладке «Исходный вид» можно найти более подробную информацию по тексту статьи.

Если есть замечания к оформлению или содержанию статьи, то их нужно направить автору (одному из авторов) по электронной почте.

2.2. Создание оригинал-макета

Когда статьи собраны, можно приступить к формированию оригинал-макета. Вначале формируется рабочий каталог на основе поставляемого архива, затем в него помещаются публикуемые статьи и создаются файлы с рецензиями и экспертными заключениями из имеющихся прототипов.

Структура рабочего каталога с указанием названий подкаталогов и файлов приведена в таблице 1 и на рис. 1.

Научный редактор определяет основные разделы номера, распределяет статьи по разделам, добавляет необходимую информацию в титульные листы и помещает статьи в оригинал-макет. Наименования разделов определяются научным редактором с учетом имеющегося опыта создания предыдущих номеров журнала.

Всем статьям, вошедшим в номер, назначаются рецензенты (один на статью) и эксперты (три на статью). На основе прототипов готовятся рецензии, а также экспертные заключения о возможности опубликования статей и номера целиком. В рецензию на статью помещаются имя рецензента, имена авторов статьи и ее название.

В экспертное заключение о возможности опубликования материала (статьи) помещаются имена экспертов, имена авторов статьи, ее название, а также количество страниц, рисунков и таблиц в этой статье. В экспертное заключение о возможности опубликования номера помещаются имена экспертов, том и номер журнала, а также количество рисунков, таблиц и страниц в этом номере.

Перед отправкой в типографию оригинал-макет нужно преобразовать из формата .docx в формат .pdf.

Подробное описание процедуры создания оригинал-макета приводится в дальнейших разделах данной статьи.

3. Общие правила оформления

Для подготовки оригинал-макета следует использовать текстовый редактор Microsoft Word 2016 или более новой версии.

Поставляемый шаблон оригинал-макета содержит:

- прототипы начальных и концевого титульных листов, а также «содержаний» (на русском и английском языках);
- прототип добавленной в журнал статьи.

В шаблоне оригинал-макета (в виде стилей) указаны все правила форматирования абзацев, также указаны требуемые параметры страниц (размеры полей и колонтитулов), где необходимо, добавлены разрывы разделов. Текст, который должен быть модифицирован, набран красным цветом и приведен в угловых скобках.

Ниже приводятся правила оформления в случае подготовки номера без применения данного

шаблона.

Для всего текста оригинал-макета используется шрифт Times New Roman.

При формировании оригинал-макета необходимо указать размеры полей и определить колонтитулы, содержащие номера страниц.

Размер всех *полей* (верхнее, нижнее, внутри (левое), снаружи (правое)) равен 3 см. Для задания размеров полей необходимо на вкладке «Макет» нажать кнопку «Параметры страницы» и в появившемся окне указать необходимые размеры полей.

Верхний колонтитул состоит из одного абзаца с нижней границей в виде сплошной линии и номером страницы. Для четных страниц их номера указываются слева (форматирование абзаца по левому краю), для нечетных – справа (форматирование абзаца по правому краю).

Нижний колонтитул всех страниц оригинал-макета, а также верхние колонтитулы титульных листов, «содержаний» и первых страниц статей оставляются пустыми (не содержат никакого текста).

Размер *верхнего* и *нижнего колонтитулов* – 1,25 см. Для добавления колонтитулов и задания их размеров необходимо на вкладке «Вставка» нажать кнопку «Верхний (Нижний) колонтитул», во всплывающем меню выбрать пункт «Изменить верхний (нижний) колонтитул» и на появившейся вкладке «Конструктор» указать необходимые размеры колонтитулов.

Для добавления нижней границы абзаца (сплошной линии) колонтитулов необходимо на вкладке «Главная» нажать кнопку «Границы», во всплывающем меню выбрать пункт «Нижняя граница».

Для добавления номера страницы необходимо на вкладке «Вставка» нажать кнопку «Номер страницы», во всплывающем меню выбрать пункт «Текущее положение | Простой номер». Номера страниц набираются 12 кеглем.

Для создания разных верхних колонтитулов на четных и нечетных страницах необходимо двойным щелчком мыши в области верхнего колонтитула перейти к его редактированию, на открывшейся вкладке «Конструктор» нажать кнопку «Параметры», во всплывающем меню проставить галочки в пунктах «Особый колонтитул для первой страницы», «Разные колонтитулы для четных и нечетных страниц». Для создания «пустого» верхнего колонтитула, где это требуется, необходимо в конце текста предыдущей страницы вставить разрыв раздела. Для этого нужно перейти во вкладку «Макет», нажать кнопку «Разрывы», во всплывающем меню выбрать пункт «Следующая страница». В результате колонтитул на странице, следующей за разрывом раздела, будет совпадать с «пустым» колонтитулом первой страницы оригинал-макета.

4. Титульные листы

Титульные листы включают в себя два начальных листа и один концевой.

Примеры всех титульных листов предоставляются в виде двух файлов: 0_Начальные.docx и Z_Конечный.docx.

4.1. Первый лист

На первом листе указываются

- полное и краткое (в скобках) название организации;
- название журнала;
- том и номер *(указывается научным редактором)*;
- название темы, которой посвящен журнал, набранное прописными буквами;
- город, место издания;
- год издания *(указывается научным редактором)*.

4.2. Второй лист

На втором листе указываются

- список членов редакционного совета ФГУ ФНЦ НИИСИ РАН;
- главный редактор журнала;
- научный редактор номера *(указывается научным редактором)*;
- тематика номера *(указывается научным редактором)*;
- краткая информация о журнале;
- тематика номера на английском языке *(указывается научным редактором)*;
- краткая информация о журнале на английском языке;
- заведующий редакцией;
- издатель (наименование и адрес);
- правообладатель *(год указывается научным редактором)*.

4.3. Концевой лист

На концевом листе указываются

- дата, когда номер был подписан в печать;
- формат;
- тип печати и количество печатных листов;
- тираж и номер заказа;
- название и адрес типографии.

5. Формирование журнала

5.1. Подготовительный этап

Для организации рабочего каталога предоставляется zip-архив «Рабочий каталог журнала Труды НИИСИ РАН.zip», содержащий каталоги и прототипы файлов, используемых для создания оригинал-макета. Содержимое архива описано в приложении 2. Архив нужно распаковать в каталог с именами ТОМ_<X>_N<Y>, где <X>

– номер тома, а <Y> – номер журнала, например, TOM_9_N3. После распаковки в этом каталоге появятся два подкаталога с именами «Статьи_Акты_Рецензии» и «Типография». Помимо этих подкаталогов, рабочий каталог содержит шаблон оригинал-макета, а также файлы с правилами оформления статьи и шаблоном статьи, прототипами рецензии на статью и экспертных заключений на статью и на весь номер.

В каталог «Статьи_Акты_Рецензии» помещаются полученные от авторов статьи, а также относящиеся к ним рецензии и экспертные заключения. При копировании файлов со статьями следует их переименовать. Новые имена файлов определяются научным редактором номера. Рекомендуется, чтобы имя файла со статьей содержало ее порядковый номер, слово «Статья», фамилии авторов (не более двух), например, «1_Статья_Онин_и_др.docx», также в имя статьи можно добавить основные параметры статьи (количество страниц, рисунков и таблиц). Имена файлов с рецензией и экспертным заключением следует именовать на основе имени соответствующего файла со статьей заменой слова «Статья» на слово «Рецензия» или «Акт».

В процессе переименования статей формируются «содержания» (на русском и английском языках), рецензии и экспертные заключения.

Рецензии создаются из прототипа (файл «Прототип рецензии на статью.docx») путем вписывания в него имени, должности и ученой степени рецензента, определяемого научным редактором номера, а также имен авторов и названия статьи.

Экспертные заключения на статью создаются из прототипа (файл «Прототип экспертного заключения на статью.docx») путем вписывания в него имен, должностей и ученых степеней экспертов, авторов и названия статьи, а также данных о количестве страниц, рисунков и таблиц в статье.

Экспертное заключение на весь номер создается из прототипа (файл «Прототип экспертного заключения на весь номер.docx») путем вписывания в него имен, должностей и ученых степеней экспертов, тома и номера, а также данных о количестве страниц, рисунков и таблиц в журнале.

Чтобы упростить формирование «содержаний», рецензий и экспертных заключений, рекомендуется это делать при переименовании статей. Для этого следует открыть файл оригинал-макета, статью и относящиеся к данной статье файлы рецензии и экспертного заключения. Далее можно приступить к первому этапу формирования содержаний на русском и английском языках, вписав в него авторов и название статьи, а вместо номера страницы количество рисунков и таблиц и порядковый номер статьи, одновременно добавляя требуемые данные по этой статье в рецензию и экспертное заключение. После обработки всех статей нужно подсчитать общее количество таблиц и рисунков в номере и вместе с количеством страниц в выпускаемом журнале занести их в экспертное заключение о возможности опубликования номера целиком.

5.2. Формирование содержания

В оригинал-макет добавляются содержания на русском и английском языках

«Содержание» состоит из

- заголовка «СОДЕРЖАНИЕ» («CONTENT» для английской версии содержания), набранного прописными буквами, шрифт полужирный, 12 кегль, выравнивание абзаца по центру;

- названий разделов журнала, пронумерованных римскими цифрами и набранных прописными буквами, шрифт полужирный, 12 кегль, выравнивание абзаца по центру;

- списка статей, распределенных по разделам, состоящего из имен авторов, набранных курсивом, названия статьи, набранного обычным шрифтом, номера первой страницы статьи, отделенного от названия статьи символами табуляции до правого края строки. Размер шрифта – 12 кегль, выравнивание абзаца по ширине.

Первый этап формирования содержания см. выше.

На втором этапе – в «содержание» необходимо добавить наименования разделов и переупорядочить по ним уже указанные в «содержании» названия статей.

В зависимости от длины названий статей и их количества, «содержание» может занимать одну или несколько страниц. Поэтому номера страниц рекомендуется указывать на заключительном (третьем) этапе формирования «содержания».

Те же действия необходимо произвести для формирования содержания на английском языке.

Завершить формирование «содержания» можно лишь после того, как в оригинал-макет помещены все статьи.

5.3. Формирование оригинал-макета

Оригинал-макет состоит из двух начальных титульных листов, «содержания» (на русском и английском языках), статей и концевого титульного листа.

В качестве исходного файла для формирования оригинал-макета берется шаблон оригинал-макета «Шаблон оригинал-макета журнала Труды НИИСИ РАН.dotx». При открытии шаблона будет создан новый документ, связанный с этим шаблоном. В этом документе необходимо:

- модифицировать начальные и концевые титульные листы (см. раздел 4);

- сформировать «содержание» (на русском и английском языках) (см. 5.1, 5.2);

- добавить в оригинал-макет статьи в порядке, указанном в «содержании».

Для сохранения исходного форматирования статей рекомендуется добавлять их текст в формируемый оригинал-макет в виде объекта редактора Word, называемого «Текст из файла». Для этого в оригинал-макете необходимо на вкладке «Вставка» нажать кнопку «объект», во всплывающем меню выбрать пункт «Текст из файла» и выбрать требуемый файл статьи или концевого титульного листа.

Титульные листы, «содержания» и каждая статья начинаются с новой страницы. В этих целях в конце текста данных разделов добавляются разрывы с переходом на следующую страницу. Для добавления такого разрыва необходимо на вкладке «Макет» нажать кнопку «Разрывы» и во всплывающем меню выбрать пункт «Следующая страница». В шаблоне оригинал-макета разрывы между титульными листами, «содержанием» и примером статьи уже присутствуют.

5.3. Преобразование в формат PDF

Для преобразования оригинал-макета в формат PDF рекомендуется использовать программу Adobe Acrobat Pro X или более поздней версии. В этом случае сделать преобразование оригинал-макета можно непосредственно из редактора MS Word. Для этого необходимо открыть закладку «ACROBAT» (которая должна появиться после установки программы Adobe Acrobat Pro на компьютер), нажать на кнопку «Create PDF» и указать место, куда будет сохранен PDF-файл.

6. Формирование аннотации

Аннотация состоит из

- заголовка, содержащего название журнала, тома, номера журнала и года выпуска (например, «Труды НИИСИ РАН, Т. 2, №5, 2022». Размер шрифта – 16 кегль, полужирный, выравнивание абзаца по центру;

- краткой информации по статьям, входящим в журнал.

Краткая информация по статье включает в себя

- название статьи с порядковым номером публикации статьи в оригинал-макете. Размер шрифта – 14 кегль, полужирный, выравнивание абзаца по центру;

- список авторов с номерными сносками на каждого автора. Размер шрифта – 12 кегль, полужирный, выравнивание абзаца по центру;

- сноски с информацией о каждом авторе статьи, включающей в себя место работы, город, страну и адрес электронной почты. Размер шрифта – 12 кегль, выравнивание абзаца по центру;

- аннотация. Размер шрифта – 12 кегль, выравнивание абзаца по ширине, «Аннотация.» выделено полужирным;

- ключевые слова. Размер шрифта – 12 кегль, выравнивание абзаца по ширине, «Ключевые слова:» выделено полужирным.

7. Формирование рецензий и экспертных заключений

Рецензии и экспертные заключения следует составлять на основе прототипов, поставляемых с рабочим каталогом (см. Приложение 2).

В прототип экспертного заключения необходимо внести вид и название материала, фамилии, и., о. авторов, количество страниц, рисунков, таблиц, а также заполнить степени, должности, фамилии, и., о. трех выбранных экспертов.

В прототип рецензии на статью необходимо внести и., о., фамилии. авторов статьи, название статьи, а также степень, звание, должность, и., о., фамилию рецензента.

После формирования оригинал-макета необходимо заполнить прототип экспертного заключения на журнал, внеся в прототип том, номер, год выпуска журнала, количество страниц, общее количество рисунков и таблиц, содержащихся в данном выпуске, а также степени, должности и фамилии, и., о. членов экспертной комиссии.

8. Устранение возможных проблем

В некоторых случаях при добавлении статьи в оригинал-макет может нарушиться ее исходное форматирование. Такое возможно, если статья не была оформлена в соответствии с требованиями, приведенными в [1]. В этой ситуации рекомендуется привести статью в соответствие с шаблоном и повторно добавить ее в оригинал-макет.

При добавлении статьи, набранной в редакторе MS Word версии ниже 2010, возможна подмена разрывов раздела с началом следующего раздела на текущей странице на разрыв раздела со следующей страницы. В этой ситуации следует поабзацно перенести статью в шаблон, используя актуальные версии MS Word.

9. Заключение

В статье описана последовательность всех действий, необходимых для формирования оригинал-макета номера журнала «Труды НИИСИ РАН».

В соответствии с описанной технологией верстаются все выпуски, начиная с 3-го номера 9-го тома журнала «Труды НИИСИ РАН».

Авторы выражают надежду, что использование данной статьи в качестве руководства по подготовке журнала к изданию поможет сократить время подготовки и повысить качество выпускаемого номера. Просьба присылать авторам замечания и предложения по улучшению методики подготовки журнала к изданию.

Приложение 1 – Структура рабочего каталога

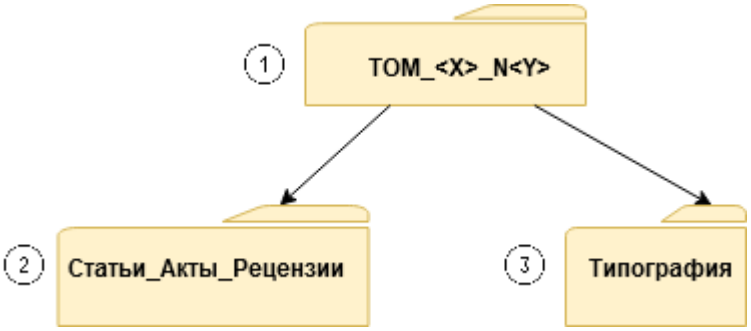

Рис. 1. Структура рабочего каталога

Таблица 1. Структура рабочего каталога

Файл / каталог	Название	Примеры
Каталоги		
1. Рабочий каталог	TOM_<X>_N<Y>	TOM_9_N3
2. Каталог для статей, экспертных заключений, рецензий и титульных листов журнала (находится в каталоге 1)	Статьи_Акты_Рецензии	
3. Каталог оригинал-макета (находится в каталоге 1)	Типография	
Файлы		
Файл с последней версией шаблона оригинал-макета (находится в каталоге 1)	Шаблон оригинал-макета (версия <номер версии> от <дата выпуска версии>).docx	Шаблон оригинал-макета (версия 1.00 от 24.12.2019).docx
Файл оригинал-макета в формате .docx (находится в каталоге 3)	Труды_НИИСИ_РАН_Т<X>_N<Y>.docx	Труды_НИИСИ_РАН_Т9_N3.docx
Файл оригинал-макета в формате .pdf (находится в каталоге 3)	Труды_НИИСИ_РАН_Т<X>_N<Y>.pdf	Труды_НИИСИ_РАН_Т9_N3.pdf
Файл с примером первых титульных листов (находится в каталоге 2)	Начальные_ТЛ.docx	
Файл с примером концевого титульного листа (находится в каталоге 2)	Концевой_ТЛ.docx	
Файлы авторских статей (находится в каталоге 2)	<N>_Статья_<Фамилия первого автора>[_<Фамилия второго автора (если есть) или слова «и_др», если статья имеет больше двух авторов>].doc[x]	3_Статья_Бурцев.doc 4_Статья_Грингауз_Онин.doc 8_Статья_Тимохин_и_др.docx
Файлы экспертных заключений на статью (находится в каталоге 2)	<N>_Акт_<Фамилия первого автора>[_<Фамилия второго автора (если есть) или слова «и_др», если статья имеет больше двух авто-ров>].doc	1_Акт_Онин_и_др.doc 7_Акт_Мальцев_Вожегов.doc 6_Акт_Левченкова.doc
Файлы рецензий на статью (находится в каталоге 2)	<N>_Рецензия_<Фамилия первого автора>[_<Фамилия второго автора (если есть) или слова «и_др», если статья имеет больше двух авторов>].doc	3_Рецензия_Бурцев.doc 4_Рецензия_Грингауз_Онин.doc 8_Рецензия_Тимохин_и_др.doc

Файл с прототипом экспертного заключения на статью (находится в каталоге 1)	Прототип экспертного заключения на статью.docx	
Файл с прототипом экспертного заключения на весь номер (находится в каталоге 1)	Прототип экспертного заключения на весь номер.docx	
Файл с прототипом рецензии на статью (находится в каталоге 1)	Прототип рецензии на статью.docx	
Файл с прототипом аннотации (находится в каталоге 3)	Прототип аннотации.docx	
Файл с последней версией правил оформления статей для журнала (находится в каталоге 1)	Подготовка статей для журнала Труды НИИСИ РАН (версия <номер версии> от <дата выпуска версии>).docx	Подготовка статей для журнала Труды НИИСИ РАН (версия 1.05 от 24.12.2019).docx
Файл с последней версией шаблона статьи для журнала (находится в каталоге 1)	Шаблон статьи для сборника трудов НИИСИ РАН (версия <номер версии шаблона> от <дата выпуска версии шаблона>).dotx	Шаблон статьи для сборника трудов НИИСИ РАН (версия 1.05 от 24.12.2019).dotx
Файл с последней версией руководства по подготовке к изданию журнала «Труды НИИСИ РАН» (данная статья) (находится в каталоге 1)	Подготовка к изданию журнала Труды НИИСИ РАН (версия <номер версии> от <дата выпуска версии>).docx	Подготовка к изданию журнала Труды НИИСИ РАН (версия 1.00 от 24.12.2019).docx
Условные обозначения: <X> – номер тома <Y> – номер журнала <N> – номер статьи <Z> – количество статей в номере + 1 *Примечание:* Нумерация статей определяется научным редактором номера.		

Приложение 2 – Файлы архива рабочего каталога

Корневой каталог:

- «Шаблон оригинал-макета (версия 1.00 от 24.12.2019).docx» – последняя версия шаблона оригинал-макета;
- «Правила оформления статей в журнал Труды НИИСИ РАН (версия 1.05 от 24.12.2019).docx» – последняя версия правил оформления статей для журнала;
- «Шаблон статьи для сборника трудов НИИСИ РАН (версия 1.05 от 24.12.2019).dotx» – последняя версия MS Word шаблона статьи для журнала;
- «Прототип экспертного заключения на статью.docx» – прототип экспертного заключения на статью;

«Прототип экспертного заключения на весь номер.docx» – прототип экспертного заключения на статью;

- «Прототип рецензии на статью.docx» – прототип рецензии на статью;
- «Прототип аннотации.docx» - прототип аннотации;
- «Подготовка к изданию журнала Труды НИИСИ РАН (версия 2.06 от 21.07.2022).docx» – последняя версия руководства по подготовке к изданию журнала «Труды НИИСИ РАН» (данная статья).

Подкаталог «Статьи_Акты_Рецензии»:

- «Начальные_ТЛ.docx» – пример первых титульных листов;
- «Концевой_ТЛ.docx» – пример концевого титульного листа.

Preparatory Works for the Publication of the “Proceedings of SRISA RAS” Journal

Alexander Asonov, Alexander Godunov

Abstract. The article describes the process of preparing journal for publishing – from collecting articles to creating a layout original for a printing office. The sequence of actions in the preparation of the journal is indicated. The tools used are described, and recommendations are given for their use. The requirements for the design of pages, headers and footers, title pages, and content are given. An overview and recommendations are considered on the use of the Publishing System for the “Proceedings of SRISA RAS” journal. The article is intended for those preparing the journal for publication, and will also be useful to the authors of the articles, acquainting them with the process of creation and release of the journal.

Keywords: preparation of the journal, SRISA RAS, layout original, scientific editor, publishing system

Литература

1. А.А. Асонов, А.Н. Годунов. Подготовка статей для журнала «Труды НИИСИ РАН». «Труды НИИСИ РАН», Т. 9 (2019), № 5, 130–156.

Подписано в печать 23.08.2022 г.
Формат 60x90/8
Печать цифровая. Печатных листов 6
Тираж 100 экз. Заказ № 460

Отпечатано в ФГБУ «Издательство «Наука»
(Типография «Наука»)
121099, Москва, Шубинский пер., 6